

# Azure para Arquitetos

Terceira edição

Crie aplicações seguras, escalonáveis e de alta disponibilidade na nuvem



Ritesh Modi, Jack Lee e Rithin Skaria

Packt

[www.packt.com](http://www.packt.com)

# Azure para Arquitetos

## *Terceira edição*

Crie aplicações seguras, escalonáveis e de alta disponibilidade na nuvem

Ritesh Modi, Jack Lee e Rithin Skaria

**Packt**

## Azure para Arquitetos – Terceira edição

Copyright © 2021 Packt Publishing

Todos os direitos reservados. Nenhuma parte deste livro pode ser reproduzida, armazenada em um sistema de recuperação ou transmitida sob qualquer forma ou por qualquer meio sem a permissão prévia por escrito da editora, exceto no caso de breves citações incorporadas em artigos críticos ou comentários.

Não se poupou esforços na preparação deste livro para garantir a precisão das informações apresentadas. No entanto, as informações contidas neste livro são vendidas sem garantia, expressa ou implícita. Os autores, a Packt Publishing e seus revendedores e distribuidores não serão responsabilizados por quaisquer danos causados ou supostamente causados de forma direta ou indireta por este livro.

A Packt Publishing empenha-se em fornecer informações de marca registrada sobre todas as empresas e produtos mencionados neste livro pelo uso adequado de capitais. No entanto, a Packt Publishing não garante a precisão dessas informações.

Autores: Ritesh Modi, Jack Lee e Rithin Skaria

Revisores técnicos: Melony Qin e Sanjeev Kumar

Editores-gerentes: Aditya Datar e Afzal Shaikh

Editor de aquisições: Shrilekha Inani

Editores de produção: Ganesh Bhadwalkar e Deepak Chavan

Conselho editorial: Vishal Bodwani, Ben Renow-Clarke, Edward Doxey, Joanne Lovell, Arijit Sarkar e Dominic Shakeshaft

Primeira edição: outubro de 2017

Segunda edição: janeiro de 2019

Terceira edição: junho de 2020

Referência de produção: 3260620

ISBN: 978-1-83921-586-5

Publicado pela Packt Publishing Ltd.

Livery Place, 35 Livery Street

Birmingham B3 2PB, UK





# startHere(Azure);

**Adquira experiência  
prática na nuvem**

---

Aprenda a usar o Azure. Experimente mais de 100 serviços.

- Serviços gratuitos
- Crédito de USD 200
- Treinamento gratuito

[Experimente o Azure grátis >](#)

Obtenha ajuda com seu projeto.  
[Fale com um especialista em vendas >](#)

# Sumário

## Prefácio

---

i

<b>Capítulo 1: Introdução ao Azure</b>	<b>1</b>
Computação na nuvem .....	2
Vantagens da computação na nuvem .....	3
Por que adotar a computação na nuvem? .....	3
Paradigmas de implantação no Azure .....	5
Noções básicas sobre o Azure .....	6
O Azure como nuvem inteligente .....	8
Azure Resource Manager .....	8
Arquitetura do ARM .....	9
Por que usar o ARM? .....	9
Vantagens do ARM .....	10
Conceitos do ARM .....	11
Virtualização .....	14
Contêineres .....	15
Docker .....	17
Interação com a nuvem inteligente .....	17
O portal do Azure .....	17
PowerShell .....	18
A CLI do Azure .....	18
A API REST do Azure .....	19
Modelos do ARM .....	19
Resumo .....	20

<b>Capítulo 2: Disponibilidade, escalabilidade e monitoramento da solução do Azure</b>	<b>23</b>
<hr/>	
Alta disponibilidade .....	24
Alta disponibilidade do Azure .....	25
Conceitos .....	26
Balanceamento de carga .....	29
Alta disponibilidade de VMs .....	30
Alta disponibilidade de computação .....	30
Plataformas de alta disponibilidade .....	32
Balanceadores de carga no Azure .....	32
O Gateway de Aplicação do Azure .....	35
Gerenciador de Tráfego do Azure .....	36
Azure Front Door .....	38
Considerações sobre arquitetura para alta disponibilidade .....	38
Alta disponibilidade em regiões do Azure .....	39
Alta disponibilidade entre regiões do Azure .....	40
Escalabilidade .....	42
Escalabilidade versus performance .....	43
Escalabilidade do Azure .....	44
Escalabilidade da PaaS .....	46
Escalabilidade da IaaS .....	49
Conjuntos de escalas de VM .....	50
Arquitetura de VMSS .....	51
Escala VMSS .....	51

Atualizações e manutenção .....	54
Atualizações de aplicações .....	56
Atualizações de convidados .....	56
Atualizações de imagens .....	56
Práticas recomendadas de dimensionamento para VMSSs .....	57
Monitoramento .....	58
Monitoramento do Azure .....	59
Logs de atividades do Azure .....	59
Logs de diagnóstico do Azure .....	60
Logs de aplicações do Azure .....	60
Logs de sistema operacional convidado e do host .....	60
Azure Monitor .....	61
Azure Application Insights .....	61
Azure Log Analytics .....	61
Soluções .....	62
Alertas .....	63
Resumo .....	67
<b>Capítulo 3: Padrão de design – redes, armazenamento, mensagens e eventos</b>	<b>69</b>
<hr/>	
Zonas e regiões de disponibilidade do Azure .....	70
Disponibilidade dos recursos .....	70
Conformidade de dados e privacidade .....	70
Performance de aplicações .....	71
Custo da execução das aplicações .....	71

Redes virtuais .....	71
Considerações sobre a arquitetura de redes virtuais .....	72
Benefícios das redes virtuais .....	76
Design da rede virtual .....	76
Conexão com recursos dentro da mesma região e assinatura .....	77
Conexão com recursos dentro da mesma região em outra assinatura .....	77
Conexão com recursos em diferentes regiões em outra assinatura .....	79
Conectando-se a datacenters na infraestrutura local .....	80
Armazenamento .....	84
Categorias de armazenamento .....	84
Tipos de armazenamento .....	84
Recursos de armazenamento .....	85
Considerações sobre arquitetura para contas de armazenamento .....	86
Padrões de design de nuvem .....	88
Padrões de mensagens .....	89
Padrões de performance e escalabilidade .....	93
Resumo .....	101
<b>Capítulo 4: Automatizar a arquitetura no Azure</b>	<b>103</b>
Automação .....	104
Automação do Azure .....	105
Arquitetura da Automação do Azure .....	105
Automação de processos .....	107
Gerenciamento de configuração .....	108
Gerenciamento de atualizações .....	109

Conceitos relacionados à Automação do Azure .....	109
Runbook .....	109
Contas de execução .....	110
Trabalhos .....	111
Ativos .....	112
Credenciais .....	112
Certificados .....	113
Criar uma entidade de serviço usando credenciais de certificado .....	115
Conexões .....	116
Criação e execução do runbook .....	118
Runbooks pais e filhos .....	119
Criar um runbook .....	120
Usar módulos Az .....	122
Webhooks .....	125
Invocar um webhook .....	127
Invocar um runbook do Azure Monitor .....	129
Hybrid Workers .....	134
Configuração de estado da Automação do Azure .....	136
Preços da Automação do Azure .....	141
Comparação com a automação sem servidor .....	141
Resumo .....	142

<b>Capítulo 5: Criar políticas, bloqueios e marcas para implantações do Azure</b>	<b>145</b>
<b>Grupos de gerenciamento do Azure .....</b>	<b>146</b>
<b>Marcas do Azure .....</b>	<b>147</b>
<b>Marcas com o PowerShell .....</b>	<b>150</b>
<b>Marcas com modelos do Azure Resource Manager .....</b>	<b>150</b>
<b>Marcar grupos de recursos versus recursos .....</b>	<b>151</b>
<b>Azure Policy .....</b>	<b>152</b>
<b>Políticas internas .....</b>	<b>153</b>
<b>Linguagem de políticas .....</b>	<b>153</b>
<b>Campos permitidos .....</b>	<b>156</b>
<b>Bloqueios do Azure .....</b>	<b>156</b>
<b>Azure RBAC .....</b>	<b>158</b>
<b>Funções personalizadas .....</b>	<b>161</b>
<b>Como os bloqueios são diferentes do RBAC? .....</b>	<b>162</b>
<b>Azure Blueprints .....</b>	<b>162</b>
<b>Um exemplo de implementação de recursos de governança do Azure .....</b>	<b>163</b>
<b>Contexto .....</b>	<b>163</b>
<b>RBAC para Company Inc .....</b>	<b>163</b>
<b>Azure Policy .....</b>	<b>164</b>
<b>Bloqueios do Azure .....</b>	<b>165</b>
<b>Resumo .....</b>	<b>165</b>

## **Capítulo 6: Gerenciamento de custos para soluções do Azure 167**

---

Detalhes das ofertas do Azure .....	168
Noções básicas de cobrança .....	169
Faturamento .....	176
A experiência do Comércio Moderno .....	177
Uso e cotas .....	179
Provedores de recursos e tipos de recursos .....	180
APIs de Uso e Cobrança .....	182
APIs de Cobrança Empresarial do Azure .....	182
APIs de Consumo do Azure .....	183
APIs de Gerenciamento de Custos do Azure .....	184
Calculadora de preços do Azure .....	184
Práticas recomendadas .....	187
Governança do Azure .....	187
Práticas recomendadas de computação .....	188
Práticas recomendadas de armazenamento .....	189
Práticas recomendadas de PaaS .....	190
Práticas recomendadas gerais .....	191
Resumo .....	191

---

## **Capítulo 7: Soluções OLTP do Azure** 193

---

<b>Aplicações OLTP .....</b>	<b>194</b>
Bancos de dados relacionais .....	195
<b>Serviços de nuvem do Azure .....</b>	<b>195</b>
<b>Modelos de implantação .....</b>	<b>196</b>
Bancos de dados em Máquinas Virtuais do Azure .....	197
Bancos de dados hospedados como serviços gerenciados .....	198
<b>Banco de Dados SQL do Azure .....</b>	<b>198</b>
Recursos da aplicação .....	199
Segurança .....	204
<b>Instância Única .....</b>	<b>210</b>
<b>Pools elásticos .....</b>	<b>211</b>
<b>Instância Gerenciada .....</b>	<b>213</b>
<b>Preços do banco de dados SQL .....</b>	<b>215</b>
Preços baseados em DTU .....	215
Preços baseados em vCPU .....	217
Como escolher o modelo de preços apropriado .....	218
<b>Azure Cosmos DB .....</b>	<b>219</b>
Recursos .....	221
Cenários de uso .....	222
<b>Resumo .....</b>	<b>222</b>

<b>Capítulo 8: Arquitetar aplicações seguras no Azure</b>	<b>225</b>
<b>Segurança .....</b>	<b>226</b>
Ciclo de vida da segurança .....	228
Segurança do Azure .....	230
<b>Segurança de IaaS .....</b>	<b>231</b>
Grupos de segurança de rede .....	231
Firewalls .....	234
Grupos de segurança de aplicações .....	235
Firewall do Azure .....	236
Reducir a área de superfície de ataque .....	237
Implementar servidores de salto .....	238
Azure Bastion .....	239
<b>Segurança de aplicações .....</b>	<b>239</b>
SSL/TLS .....	239
Identidades gerenciadas .....	240
<b>Azure Sentinel .....</b>	<b>244</b>
<b>Segurança de PaaS .....</b>	<b>245</b>
Link Privado do Azure .....	245
Gateway de Aplicativo do Azure .....	245
Azure Front Door .....	246
Ambiente do Serviço de Aplicativo do Azure .....	247
Log Analytics .....	247

Armazenamento do Azure .....	248
SQL do Azure .....	252
Azure Key Vault .....	256
Autenticação e autorização usando OAuth .....	257
Monitoramento e auditoria de segurança .....	265
Azure Monitor .....	265
Central de Segurança do Azure .....	267
Resumo .....	268
<b>Capítulo 9: Soluções de Big Data do Azure</b>	<b>271</b>
<hr/>	
Big data .....	272
Processo de big data .....	273
Ferramentas de big data .....	274
Azure Data Factory .....	274
Azure Data Lake Storage .....	274
Hadoop .....	275
Apache Spark .....	276
Databricks .....	276
Integração de dados .....	276
ETL .....	277
Uma cartilha sobre o Azure Data Factory .....	278
Uma cartilha sobre o Azure Data Lake Storage .....	279

Migrar dados do Armazenamento do Azure para o Data Lake Storage Gen2 .....	280
Preparar a conta de armazenamento de origem .....	280
Provisionar um novo grupo de recursos .....	280
Provisionar uma conta de armazenamento .....	281
Provisionar o serviço Data Lake Storage Gen2 .....	283
Provisionar o Azure Data Factory .....	284
Configurações do repositório .....	285
Conjuntos de dados do Data Factory .....	287
Criar o segundo conjunto de dados .....	289
Criar um terceiro conjunto de dados .....	289
Criar um pipeline .....	291
Adicionar mais uma atividade Copiar Dados .....	293
Criar uma solução usando o Databricks .....	294
Carregar dados .....	297
Resumo .....	303
<b>Capítulo 10: Sem servidor no Azure – Trabalhar com o Azure Functions</b>	<b>305</b>
Sem servidor .....	306
As vantagens do Azure Functions .....	306
FaaS .....	308
O tempo de execução do Azure Functions .....	308
Associação e gatilhos do Azure Functions .....	309
Configuração do Azure Functions .....	312
Planos de custo do Azure Functions .....	314
Hosts de destino do Azure Functions .....	316
Casos de uso do Azure Functions .....	316
Tipos de funções do Azure .....	318

Criar uma função orientada a eventos .....	318
Function Proxies .....	321
Durable Functions .....	322
Etapas para criar uma função durável usando o Visual Studio .....	324
Criar uma arquitetura conectada com funções .....	329
Grade de Eventos do Azure .....	332
Grade de Eventos .....	333
Eventos de recursos .....	335
Eventos personalizados .....	340
Resumo .....	343
<b>Capítulo 11: Soluções do Azure usando Aplicativos Lógicos do Azure, Grade de Eventos e Functions</b>	<b>345</b>
<hr/>	
Aplicativos Lógicos do Azure .....	346
Atividades .....	346
Conectores .....	346
O funcionamento de um Aplicativo Lógico .....	347
Criação de uma solução de ponta a ponta usando tecnologias sem servidor .....	355
A declaração do problema .....	355
Solução .....	355
Arquitetura .....	356
Pré-requisitos .....	357
Implementação .....	357
Testes .....	385
Resumo .....	386

<b>Capítulo 12: Soluções de eventos de big data do Azure</b>	<b>389</b>
<b>Introdução a eventos .....</b>	<b>390</b>
Streaming de eventos .....	391
Hubs de Eventos .....	392
<b>Arquitetura dos Hubs de Eventos .....</b>	<b>395</b>
Grupos de consumidores .....	402
Taxa de transferência .....	403
<b>Uma cartilha sobre o Stream Analytics .....</b>	<b>403</b>
Ambiente de hospedagem .....	407
Unidades de streaming .....	408
<b>Uma aplicação de exemplo usando os Hubs de Eventos e o Stream Analytics .....</b>	<b>408</b>
Provisionar um novo grupo de recursos .....	408
Criando um namespace dos Hubs de Eventos .....	409
Criar um hub de eventos .....	410
Provisionar um aplicativo lógico .....	411
Provisionar a conta de armazenamento .....	413
Criar um contêiner de armazenamento .....	413
Criar trabalhos do Stream Analytics .....	414
Executar a aplicação .....	416
Resumo .....	418

<b>Capítulo 13: Integrar o DevOps do Azure</b>	<b>421</b>
DevOps .....	422
A essência do DevOps .....	425
Práticas de DevOps .....	427
Gerenciamento de configuração .....	428
Ferramentas de gerenciamento de configuração .....	429
Integração contínua .....	430
Implantação contínua .....	433
Entrega contínua .....	435
Aprendizado contínuo .....	435
Azure DevOps .....	436
TFVC .....	439
Git .....	439
Preparar para o DevOps .....	440
Organizações do Azure DevOps .....	441
Provisionar o Azure Key Vault .....	442
Provisionar um servidor/serviço de gerenciamento de configuração .....	442
Log Analytics .....	443
Contas de armazenamento do Azure .....	443
Imagens do Docker e do sistema operacional .....	443
Ferramentas de gerenciamento .....	443
DevOps para soluções de PaaS .....	444
Serviço de Aplicativo do Azure .....	445
Slots de implantação .....	445
SQL do Azure .....	446
Os pipelines de build e lançamento .....	446

<b>DevOps para IaaS .....</b>	<b>458</b>
Máquinas virtuais do Azure .....	458
Balanceadores de carga públicos do Azure .....	459
O pipeline de build .....	459
O pipeline de lançamento .....	460
<b>DevOps com contêineres .....</b>	<b>462</b>
Contêineres .....	462
O pipeline de build .....	463
O pipeline de lançamento .....	463
<b>Azure DevOps e Jenkins .....</b>	<b>464</b>
<b>Automação do Azure .....</b>	<b>466</b>
Provisionar uma conta da Automação do Azure .....	467
Criar configuração do DSC .....	468
Importar a configuração do DSC .....	469
Compilar a configuração do DSC .....	470
Atribuir configurações a nós .....	470
Validação .....	471
<b>Ferramentas para DevOps .....</b>	<b>471</b>
<b>Resumo .....</b>	<b>473</b>
<hr/> <b>Capítulo 14: Arquitetar soluções de Kubernetes do Azure</b>	<b>475</b>
<b>Introdução aos contêineres .....</b>	<b>476</b>
<b>Fundamentos do Kubernetes .....</b>	<b>477</b>
<b>Arquitetura do Kubernetes .....</b>	<b>479</b>
Clusters do Kubernetes .....	480
Componentes do Kubernetes .....	481

Primitivos do Kubernetes .....	484
Pod .....	485
Serviços .....	486
Implantações .....	488
Controlador de replicação e ReplicaSet .....	490
ConfigMaps e segredos .....	491
Arquitetura do AKS .....	492
Implantar um cluster do AKS .....	493
Criar um cluster do AKS .....	493
Kubectl .....	495
Conectar ao cluster .....	495
Rede do AKS .....	500
Kubenet .....	501
CNI do Azure (rede avançada) .....	503
Acesso e identidade para o AKS .....	504
Kubelet virtual .....	505
Nós virtuais .....	506
Resumo .....	507
<b>Capítulo 15: Implantações entre assinaturas usando modelos do ARM</b> .....	<b>509</b>
Modelos do ARM .....	510
Implantar grupos de recursos com modelos do ARM .....	513
Implantar modelos do ARM .....	515
Implantação de modelos usando a CLI do Azure .....	516
Implantar recursos em assinaturas e grupos de recursos .....	517
Outro exemplo de implantações entre assinaturas e grupos de recursos .....	519

Implantar entre assinaturas e grupos de recursos usando modelos vinculados .....	522
Soluções de máquina virtual usando modelos do ARM .....	526
Soluções PaaS usando modelos do ARM .....	532
Soluções relacionadas a dados usando modelos do ARM .....	534
Criar uma solução IaaS no Azure com o Active Directory e o DNS .....	541
Resumo .....	545
<b>Capítulo 16: Design modular e implementação de modelos do ARM</b>	<b>547</b>
Problemas com a abordagem de um único modelo .....	548
Flexibilidade reduzida na alteração de modelos .....	548
Solucionar problemas com modelos grandes .....	548
Abuso de dependência .....	549
Agilidade reduzida .....	549
Sem reutilização .....	549
Entender o princípio da responsabilidade única .....	550
Solução de problemas e depuração mais rápidas .....	550
Modelos modulares .....	550
Recursos de implantação .....	551
Modelos vinculados .....	552
Modelos aninhados .....	554
Configurações de fluxo livre .....	556
Configurações conhecidas .....	556
Entender copy e copyIndex .....	567
Proteger modelos do ARM .....	569
Usar saídas entre modelos do ARM .....	570
Resumo .....	573

---

## Capítulo 17: Projetar soluções IoT 575

---

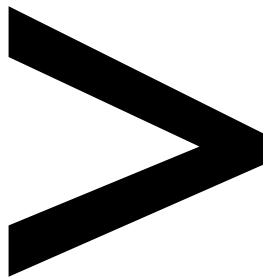
IoT .....	576
Arquitetura de IoT .....	577
Conectividade .....	579
Identidade .....	581
Captura .....	581
Ingestão .....	581
Armazenamento .....	582
Transformação .....	582
Análise .....	582
Apresentação .....	583
Azure IoT .....	584
Conectividade .....	584
Identidade .....	585
Captura .....	585
Ingestão .....	585
Armazenamento .....	586
Transformação e análise .....	586
Apresentação .....	587
Hub IoT do Azure .....	588
Protocolos .....	589
Registro do dispositivo .....	589
Gerenciamento de mensagens .....	590
Segurança .....	593
Escalabilidade .....	594
Azure IoT Edge .....	596
Alta disponibilidade .....	596

Azure IoT Central .....	597
Resumo .....	598
<b>Capítulo 18: Azure Synapse Analytics para arquitetos</b>	<b>601</b>
Azure Synapse Analytics .....	602
Um cenário comum para arquitetos .....	603
Uma visão geral do Azure Synapse Analytics .....	603
O que é isolamento de workloads? .....	604
Introdução aos espaços de trabalho do Synapse e ao Synapse Studio .....	605
Apache Spark para Synapse .....	607
Synapse SQL .....	608
Pipelines do Synapse .....	609
Azure Synapse Link para Cosmos DB .....	610
Migrar de sistemas herdados existentes para o Azure Synapse Analytics .....	611
Por que você deve migrar seu data warehouse herdado para o Azure Synapse Analytics? .....	611
O processo de migração em três etapas .....	613
Os dois tipos de estratégias de migração .....	614
Reducir a complexidade do seu data warehouse herdado existente antes de migrar .....	615
Converter data marts físicos em data marts virtuais .....	615
Migrar esquemas de data warehouse existentes para o Azure Synapse Analytics .....	616
Migrar dados históricos e processos de ETL de seu data warehouse herdado para o Azure Synapse Analytics .....	619
Migrar os processos de ETL existentes para o Azure Synapse Analytics .....	621
Redesenvolver processos de ETL escaláveis usando o ADF .....	622
Recomendações para migração de consultas, relatórios de BI, painéis e outras visualizações .....	622
Problemas comuns de migração e resoluções .....	623

Incompatibilidades do SQL comuns e resoluções .....	625
Diferenças e resoluções de DDL SQL .....	626
Diferenças e resoluções de DML SQL .....	627
Diferenças e resoluções de DCL SQL .....	627
Diferenças do SQL estendido e soluções alternativas .....	631
Considerações sobre segurança .....	632
Criptografia de dados em repouso .....	632
Dados em movimento .....	632
Ferramentas para ajudar a migrar para o Azure Synapse Analytics .....	633
ADF .....	633
Azure Data Warehouse Migration Utility .....	634
Serviços da Microsoft para transferência de dados físicos .....	634
Serviços da Microsoft para ingestão de dados .....	635
Resumo .....	636
<b>Capítulo 19: Arquitetar soluções inteligentes</b>	<b>639</b>
A evolução da IA .....	640
Processos de IA do Azure .....	641
Ingestão de dados .....	641
Transformação de dados .....	641
Análise .....	641
Modelagem de dados .....	642
Validar o modelo .....	642
Implantação .....	642
Monitoramento .....	642

Serviços Cognitivos do Azure .....	643
Visão .....	644
Pesquisa .....	644
Linguagem .....	644
Fala .....	644
Decisão .....	644
Entender os Serviços Cognitivos .....	645
Consumir Serviços Cognitivos .....	646
Criar um serviço de OCR .....	646
Usar o Powershell .....	649
Usar C# .....	650
O processo de desenvolvimento .....	652
Criar um serviço de recursos visuais usando o SDK .NET de Pesquisa Cognitiva .....	655
Usar o Powershell .....	655
Usar .NET .....	656
Proteger a chave dos Serviços Cognitivos .....	658
Usar Proxies do Azure Functions .....	658
Consumir Serviços Cognitivos .....	659
Resumo .....	659
<b>Índice</b>	<b>661</b>





# Prefácio

## Sobre

Esta seção apresenta brevemente os autores, a cobertura deste livro, as habilidades técnicas de que você precisará para começar e os requisitos de hardware e software para arquitetar soluções usando o Azure.

## Azure para Arquitetos – Terceira edição

Graças ao seu suporte à alta disponibilidade, escalabilidade, segurança, performance e Disaster Recovery, o Azure é amplamente adotado para criar e implantar diferentes tipos de aplicação com facilidade. Atualizado de acordo com os desenvolvimentos mais recentes, esta terceira edição de *Azure para arquitetos* ajuda você a se familiarizar com os principais conceitos de design de arquitetura sem servidor, incluindo contêineres, implantações do Kubernetes e soluções de big data. Você aprenderá a arquitetar soluções como funções sem servidor, descobrirá padrões de implantação de contêineres e Kubernetes e explorará o processamento de big data de grande escala usando o Spark e o Databricks. À medida que você avança, implementará o DevOps usando o Azure DevOps, trabalhará com soluções inteligentes usando os Serviços Cognitivos do Azure e integrará segurança, alta disponibilidade e escalabilidade a cada solução. Por fim, você se aprofundará em conceitos de segurança do Azure, como OAuth, OpenConnect e identidades gerenciadas.

No final deste livro, você terá ganhado a confiança para projetar soluções inteligentes do Azure com base em contêineres e funções sem servidor.

### Sobre os autores

**Ritesh Modi** foi evangelista tecnológico sênior da Microsoft. Ele foi reconhecido como diretor regional da Microsoft por suas contribuições para produtos, serviços e comunidades da Microsoft. Ele é arquiteto de nuvem, autor publicado, palestrante e um líder popular por suas contribuições para datacenters, Azure, Kubernetes, blockchain, serviços cognitivos, DevOps, inteligência artificial e automação. Ele é o autor de oito livros.

Ritesh deu palestras em inúmeras conferências nacionais e internacionais e é um autor publicado da MSDN Magazine. Ele tem mais de uma década de experiência na criação e implantação de soluções empresariais para clientes e mais de 25 certificações técnicas. Seus interesses e hobbies são escrever livros, brincar com sua filha, assistir a filmes e aprender sobre novas tecnologias. Ele atualmente mora em Hyderabad, na Índia. Você pode segui-lo no Twitter em [@automationnext](#).

**Jack Lee** é consultor certificado sênior do Azure e líder de prática do Azure com uma paixão por desenvolvimento de software, nuvem e inovações de DevOps. Jack foi reconhecido como MVP da Microsoft por suas contribuições para a comunidade tecnológica. Ele se apresentou em vários grupos de usuários e conferências, incluindo o Global Azure Bootcamp na Microsoft Canada. Jack é um mentor experiente e juiz em hackathons e também é o presidente de um grupo de usuários que se concentra em Azure, DevOps e desenvolvimento de software. Ele é o coautor de *Análise de Nuvem com Microsoft Azure*, publicado pela Packt Publishing. Você pode seguir Jack no Twitter em [@jlee\\_consulting](#).

**Rithin Skaria** é um difusor de Open Source com mais de 7 anos de experiência no gerenciamento de workloads de Open Source no Azure, na AWS e no OpenStack. Atualmente, ele trabalha para a Microsoft e faz parte de várias atividades da comunidade de Open Source realizadas na empresa. Ele é instrutor certificado da Microsoft, engenheiro e administrador da Linux Foundation, administrador e desenvolvedor de aplicações no Kubernetes e administrador certificado do OpenStack. Quando se trata do Azure, ele tem quatro certificações (arquitetura de solução, administração do Azure, DevOps e segurança), além de ser certificado em administração do Office 365. Ele desempenhou uma função vital em várias implantações de Open Source e na administração e migração desses workloads para a nuvem. Ele também foi coautor de *Administração do Linux no Azure*, publicado pela Packt Publishing. Conecte-se com ele no LinkedIn em [@rithin-Skaria](#).

## Sobre os revisores

**Melony Qin** é uma mulher em STEM. Atualmente trabalhando como gerente de programas na Microsoft, ela é membro da **Association for Computing Machinery (ACM)** e do **Project Management Institute (PMI)**. Ela contribuiu para computação sem servidor, processamento de big data, DevOps, inteligência artificial, aprendizado de máquina e IoT com o Microsoft Azure. Ela possui todas as certificações do Azure (ambas as faixas Aplicativos e infraestrutura e Dados e IA), bem como **Certified Kubernetes Administrator (CKA)** e **Certified Kubernetes Application Developer (CKAD)**, e está trabalhando principalmente em suas contribuições para **software Open Source (OSS)**, DevOps, Kubernetes, sem servidor, análise de big data e IoT no Microsoft Azure para a comunidade. Ela é autora e coautora de dois livros, *Infraestrutura do Microsoft Azure* e *O workshop do Kubernetes*, ambos publicados pela Packt Publishing. É possível entrar contato via Twitter em [@MelonyQ](#).

**Sanjeev Kumar** é arquiteto de soluções de nuvem para SAP no Azure na Microsoft. Atualmente, mora em Zurique, na Suíça. Ele trabalha com a tecnologia SAP há mais de 19 anos e com tecnologias de nuvem pública há cerca de 8 anos, sendo os últimos 2 anos com foco no Microsoft Azure.

Em sua função de consultor de arquitetura de nuvem SAP no Azure, Sanjeev Kumar trabalhou com vários dos principais serviços financeiros e empresas de manufatura do mundo. Suas áreas de foco incluem arquitetura e design de nuvem para ajudar os clientes a migrar os sistemas SAP para o Azure e adotar as práticas recomendadas do Azure para implantações SAP, principalmente implementando Infraestrutura como Código e DevOps. Ele também trabalhou nas áreas de conteinerização e microsserviços usando o serviço Docker e Serviço de Kubernetes do Azure, processamento de dados de streaming usando o Apache Kafka e o desenvolvimento completo de aplicações usando o Node.js. Trabalhou em várias iniciativas de desenvolvimento de produtos que abrangem IaaS, PaaS e SaaS. Ele também se interessa nos tópicos emergentes de inteligência artificial, aprendizado de máquina e análise e processamento de dados em grande escala. Ele escreve sobre tópicos relacionados a SAP no Azure, DevOps e Infraestrutura como Código no LinkedIn, onde você pode encontrá-lo em [@sanjeevkumarprofile](#).

## Objetivos de aprendizagem

Até o final deste livro, você será capaz de:

- Entender os componentes da plataforma de nuvem do Azure
- Usar padrões de design de nuvem
- Usar as diretrizes de segurança empresarial para sua implantação do Azure
- Criar e implementar soluções de integração e sem servidor
- Criar soluções de dados eficientes no Azure
- Entender os serviços de contêiner no Azure

## Público-alvo

Se você for um arquiteto de nuvem, engenheiro de DevOps ou desenvolvedor que busca aprender sobre os principais aspectos arquitetônicos da plataforma de nuvem do Azure, este livro é destinado a você. Uma compreensão básica da plataforma de nuvem do Azure ajudará você a entender os conceitos abordados neste livro de forma mais eficaz.

## Abordagem

Este livro aborda cada tópico com explicações passo a passo de conceitos essenciais, exemplos práticos e perguntas de autoavaliação. Ao oferecer um equilíbrio entre teoria e experiência prática com projetos envolventes, este livro vai ajudá-lo a entender como os arquitetos trabalham no mundo real.

## Requisitos de hardware

Para oferecer a experiência ideal, recomendamos a seguinte configuração:

- Mínimo de 4 GB de RAM
- Memória mínima gratuita de 32 GB

## Requisitos de software

- Visual Studio 2019
- Versão mais recente do Docker para Windows
- Módulo AZ do PowerShell, 1.7 e posterior
- Versão mais recente da CLI do Azure
- Assinatura do Azure
- Windows Server 2016/2019
- Versão mais recente do Windows 10 de 64 bits

## Convenções

Palavras de código em textos, nomes de tabelas de bancos de dados, nomes de pastas, nomes de arquivos, extensões de arquivos, nomes de caminhos, simulações de URLs e entrada do usuário são mostrados da seguinte maneira:

A configuração do DSC ainda não é conhecida pela Automação do Azure. Ela está disponível em alguns computadores locais. Ela deve ser carregada nas Configurações do DSC da Automação do Azure. A Automação do Azure fornece o cmdlet **Import-AzureRmAutomationDscConfiguration** para importar a configuração:

```
Import-AzureRmAutomationDscConfiguration -SourcePath "C:\DSC\AA\DSCfiles\  
ConfigureSiteOnIIS.ps1" -ResourceGroupName "omsauto" -AutomationAccountName  
"datacenterautomation" -Published -Verbose
```

## Fazer o download dos recursos

O pacote de código para este livro também está hospedado no GitHub em:  
<https://github.com/PacktPublishing/Azure-for-Architects-Third-Edition>.

Também temos outros pacotes de código em nosso rico catálogo de livros e vídeos disponíveis em <https://github.com/PacktPublishing> Confira!



# 1

## Introdução ao Azure

De tempos em tempos, vemos uma inovação tecnológica que muda permanentemente todo o cenário e o ecossistema ao redor dela. Se voltarmos no tempo, veremos que as décadas de 1970 e 1980 foram o tempo dos mainframes. Esses mainframes eram enormes, frequentemente ocupavam grandes salas e eram os únicos responsáveis por quase todo o trabalho de computação. Era difícil adquirir tecnologia, e seu uso era demorado. Por isso, muitas empresas costumavam fazer pedidos de mainframes com um mês de antecedência, antes de configurar um mainframe operacional.

Então, no início da década de 1990 houve um aumento na demanda por computação pessoal e Internet. Como resultado, os computadores se tornaram muito menores e fáceis de serem adquiridos pelo público em geral. As inovações contínuas na computação pessoal e nas frentes da Internet mudaram toda a indústria de computadores. Muitas pessoas tinham desktops capazes de executar vários programas e se conectar à Internet. A ascensão da Internet também propagou a ascensão das implantações de cliente-servidor. Era possível ter servidores centralizados que hospedavam aplicações, e os serviços podiam ser acessados por qualquer pessoa que tivesse uma conexão com a Internet em qualquer lugar do mundo. Este foi também um momento em que a tecnologia de servidores ganhou destaque. O Windows NT foi lançado nessa época e logo foi seguido pelo Windows 2000 e o Windows 2003, na virada do século.

A inovação mais notável dos anos 2000 foi o surgimento e a adoção de dispositivos portáteis, especialmente os smartphones, e com eles vieram uma infinidade de aplicativos. Os aplicativos podiam se conectar a servidores centralizados na Internet e conduzir os negócios normalmente. Os usuários não eram mais dependentes de navegadores para fazer esse trabalho; todos os servidores eram auto-hospedados ou hospedados usando um provedor de serviços, como um **Provedor de Serviços de Internet (ISP)**.

Os usuários não tinham muito controle sobre seus servidores. Vários clientes e suas implantações faziam parte do mesmo servidor, mesmo sem os clientes saberem disso.

No entanto, algo mais acontecia em meados e nas últimas partes da primeira década dos anos 2000. Era o surgimento da computação na nuvem e, novamente, isso transformou todo o cenário da Indústria de TI. Inicialmente, a adoção foi lenta e as pessoas abordaram com cautela, seja porque a nuvem estava em fase inicial e ainda tinha muito a evoluir ou porque as pessoas tinham noções negativas variadas a respeito dela.

Para compreender melhor a tecnologia revolucionária, abordaremos os seguintes tópicos neste capítulo:

- Computação na nuvem
- **Infraestrutura como serviço (IaaS), Plataforma como serviço (PaaS) e Software como serviço (SaaS)**
- Noções básicas sobre o Azure
- **Azure Resource Manager (ARM)**
- Virtualização, contêineres e Docker
- Intereração com a nuvem inteligente

## Computação na nuvem

Hoje, a computação na nuvem é uma das tecnologias do futuro mais promissoras. Ela está sendo adotada por empresas de todos os portes como parte de sua estratégia de TI. Nos dias de hoje, é difícil ter uma conversa significativa sobre uma estratégia de TI sem incluir a computação na nuvem nas discussões sobre soluções em geral.

A computação na nuvem, ou simplesmente a nuvem em termos gerais, refere-se à disponibilidade de recursos na internet. Esses recursos são disponibilizados para os usuários na internet como serviços. Por exemplo, o armazenamento é disponibilizado sob demanda pela Internet para que os usuários possam armazenar seus arquivos, documentos e muito mais. Nesse caso, o armazenamento é um serviço oferecido por um provedor de nuvem.

O provedor de nuvem é uma empresa ou um consórcio de empresas que fornecem serviços de nuvem para outras empresas e consumidores. Eles hospedam e gerenciam esses serviços em nome do usuário. Ele é responsável para viabilizar e manter a integridade dos serviços. Há grandes datacenters em todo o mundo que foram abertos por provedores de nuvem para atender às demandas de TI dos usuários.

Os recursos de nuvem consistem em hospedar serviços em infraestruturas sob demanda, como de computação, redes e instalações de armazenamento. Essa variante da nuvem é conhecida como IaaS.

## Vantagens da computação na nuvem

A adoção da nuvem está em alta e crescendo devido a diversas vantagens como estas:

- **Modelo de pagamento conforme o uso:** os clientes não precisam comprar hardware e software para recursos de nuvem. Não há despesa de capital para usar um recurso de nuvem; os clientes pagam apenas pelo tempo que usam ou reservam um recurso.
- **Acesso global:** os recursos de nuvem estão disponíveis globalmente pela Internet. Os clientes podem acessar os recursos sob demanda de qualquer lugar.
- **Recursos ilimitados:** o recurso de escala da tecnologia de nuvem é ilimitada; os clientes podem provisionar quantos recursos quiserem, sem restrições. Isso também é conhecido como escalabilidade ilimitada.
- **Serviços gerenciados:** o provedor de nuvem fornece diversos serviços que são gerenciados por eles para os clientes. Isso elimina todas as responsabilidades técnicas e financeiras do cliente.

## Por que adotar a computação na nuvem?

Para entender a necessidade da computação na nuvem, devemos entender a perspectiva da indústria.

## Flexibilidade e agilidade

Em vez de criar uma grande aplicação monolítica usando uma metodologia de implantação de abordagem Big-Bang, hoje as aplicações compreendem serviços menores usando o paradigma de microsserviços. Os microsserviços ajudam a criar serviços de forma independente e autônoma. Eles podem ser evoluídos isoladamente sem reduzir toda a aplicação. Eles proporcionam grande flexibilidade e agilidade para trazer mudanças à produção de maneira mais rápida e melhor. Muitos microsserviços são convergidos para criar uma aplicação e fornecer soluções integradas aos clientes. Eles devem ser detectáveis e ter pontos de extremidade bem definidos para a integração. O número de integrações com a abordagem de microsserviços é muito alto em comparação com as aplicações monolíticas tradicionais. Essas integrações agregam complexidade ao desenvolvimento e à implantação das aplicações.

## Velocidade, padronização e consistência

Portanto, a metodologia das implantações também deve sofrer alterações para se adaptar às necessidades desses serviços, ou seja, mudanças e implantações frequentes. Para mudanças e implantações frequentes, é importante usar processos que ajudem a promover essas alterações de maneira previsível e consistente. Os processos ágeis automatizados devem ser usados para que as alterações menores possam ser implantadas e testadas isoladamente.

## Manutenção da relevância

Por fim, os destinos de implantação devem ser redefinidos. Os destinos de implantação devem ser criados facilmente em poucos segundos, e o ambiente desenvolvido deve ser consistente em todas as versões, com binários, tempos de execução, estruturas e configuração apropriados. As máquinas virtuais foram usadas com aplicações monolíticas, mas os microsserviços precisam de mais agilidade, flexibilidade e uma opção mais leve do que as máquinas virtuais. A tecnologia de contêineres é o mecanismo preferido para destinos de implantação desses serviços, e falaremos mais sobre isso mais adiante neste capítulo.

## Escalabilidade

Alguns princípios importantes do uso dos microsserviços: eles têm uma capacidade de escala ilimitada isoladamente, alta disponibilidade global, Disaster Recovery com um ponto de recuperação quase zero e objetivos de tempo. Essas qualidades dos microsserviços exigem uma infraestrutura que pode ser escalada de forma ilimitada. Não deve haver restrições de recursos. Embora esse seja o caso, também é importante que uma organização não pague pelos recursos com antecedência quando eles não forem utilizados.

## Economia

O princípio fundamental da computação na nuvem é pagar pelos recursos que estão sendo consumidos e usá-los de maneira ideal, aumentando e diminuindo o número de recursos e a capacidade automaticamente. Esses requisitos de aplicações emergentes exigem a nuvem como a plataforma preferida para facilidade de escala, alta disponibilidade, resistência a desastres, implantação de mudanças com facilidade e obtenção de implantações automatizadas previsíveis e consistentes de maneira econômica.

## Paradigmas de implantação no Azure

Há três padrões de implantação diferentes disponíveis no Azure. São eles:

- IaaS
- PaaS
- SaaS

A diferença entre esses três padrões de implantação é o nível de controle que é exercido pelos clientes por meio do Azure. A Figura 1.1 exibe os diferentes níveis de controle dentro de cada um desses padrões de implantação:

IaaS	PaaS	SaaS
Aplicações	Aplicações	Aplicações
Dados	Dados	Dados
Tempo de execução	Tempo de execução	Tempo de execução
Middleware	Middleware	Middleware
Sistema operacional	Sistema operacional	Sistema operacional
Virtualização	Virtualização	Virtualização
Servidores	Servidores	Servidores
Armazenamento	Armazenamento	Armazenamento
Rede	Rede	Rede
<b>Gerenciado pelo cliente</b>		<b>Gerenciado pelo fornecedor</b>

Figura 1.1: serviços de nuvem — IaaS, PaaS e SaaS

Fica claro na Figura 1.1 que os clientes têm mais controle ao usar implantações de IaaS e que esse nível de controle diminui continuamente à medida que progredimos das implantações de PaaS para SaaS.

### IaaS

IaaS é um tipo de modelo de implantação que permite aos clientes provisionar sua própria infraestrutura no Azure. O Azure fornece vários recursos de infraestrutura, e os clientes podem provisioná-los sob demanda. Os clientes são responsáveis por manter e governar sua própria infraestrutura. O Azure garantirá a manutenção da infraestrutura física em que esses recursos de infraestrutura virtual estão hospedados. Com base nessa abordagem, os clientes exigem operações e gerenciamento ativo no ambiente do Azure.

### PaaS

A PaaS usa a implantação de infraestrutura e o controle do cliente. Esta é uma abstração de nível mais alto em comparação com a IaaS. Nesta abordagem, os clientes trazem a própria aplicação, código e dados e os implantam na plataforma fornecida pelo Azure. Essas plataformas são gerenciadas e controladas pelo Azure, e os clientes são os únicos responsáveis por suas aplicações. Os clientes executam apenas atividades relacionadas à implantação de suas aplicações. Esse modelo fornece opções mais rápidas e fáceis para a implantação de aplicações em comparação com a IaaS.

### SaaS

A SaaS é uma abstração de nível mais alto em comparação com a PaaS. Nessa abordagem, o software e seus serviços estão disponíveis para o consumo do cliente. Os clientes só trazem seus dados para esses serviços. Eles não têm nenhum controle sobre esses serviços. Agora que temos uma compreensão básica dos tipos de serviço no Azure, vamos detalhar o Azure e entendê-lo do zero.

## Noções básicas sobre o Azure

O Azure fornece todos os benefícios da nuvem, permanecendo aberto e flexível. O Azure é compatível com uma ampla variedade de sistemas operacionais, linguagens, ferramentas, plataformas, utilitários e estruturas. Por exemplo, ele oferece suporte a Linux e Windows, SQL Server, MySQL e PostgreSQL. Ele oferece suporte à maioria das linguagens de programação, incluindo C#, Python, Java, Node.js e Bash. Ele oferece suporte a bancos de dados NoSQL, como MongoDB e Cosmos DB, e também a ferramentas de integração contínua, como Jenkins e Azure DevOps Services (anteriormente **Visual Studio Team Services - VSTS**). A ideia por trás desse ecossistema é permitir que os clientes tenham liberdade de escolha em termos de linguagem, plataforma, sistema operacional, banco de dados, armazenamento, ferramentas e utilitários. Os clientes não devem ficar limitados pela perspectiva da tecnologia, mas devem sim ser capazes de criar e focar em sua solução de negócios, e o Azure fornece a eles um acervo tecnológico de primeira linha para uso.

O Azure é compatível com a escolha de acervo tecnológico do cliente. Por exemplo, o Azure é compatível com todos os ambientes de banco de dados populares (open source e comerciais). O Azure fornece os serviços PaaS pelo SQL do Azure, MySQL e Postgres. Ele fornece um ecossistema Hadoop e oferece o HDInsight, um serviço PaaS totalmente baseado no Apache Hadoop. Ele também fornece o Hadoop na implementação de **máquinas virtuais (VM)** Linux para clientes que preferem a abordagem IaaS. O Azure também fornece o serviço de Cache Redis e oferece suporte a outros ambientes de banco de dados populares, como Cassandra, Couchbase e Oracle como uma implementação IaaS.

O número de serviços está aumentando a cada dia no Azure, e a lista mais atualizada de serviços pode ser encontrada em <https://azure.microsoft.com/services>.

O Azure também fornece um paradigma de computação na nuvem original conhecido como nuvem híbrida. A nuvem híbrida refere-se a uma estratégia de implantação em que um subconjunto de serviços é implantado em uma nuvem pública, enquanto outros serviços são implantados em uma nuvem privada ou em um datacenter na infraestrutura local. Há uma conexão de **Rede Virtual Privada (VPN)** entre a nuvem pública e a privada. O Azure oferece aos clientes a flexibilidade de dividir e implantar seu workload na nuvem pública e em um datacenter na infraestrutura local.

O Azure tem datacenters em todo o mundo, combinados em regiões. Cada região tem vários datacenters para garantir que a Disaster Recovery seja rápida e eficiente. Até o momento da redação deste documento, há 58 regiões espalhadas pelo mundo. Desse modo, os clientes têm a flexibilidade para implantar os serviços no local que desejarem. Eles também podem combinar essas regiões para implantar uma solução resistente a desastres e próxima de sua base de clientes.

### Observação

Na China e na Alemanha, os Serviços de Nuvem do Azure são separados para uso geral e para uso governamental. Isso significa que os serviços de nuvem são mantidos em datacenters separados.

## O Azure como nuvem inteligente

O Azure fornece infraestrutura e serviços para ingerir bilhões de transações usando o processamento de hiperescala. Ele fornece petabytes de armazenamento para dados, além de uma série de serviços interconectados que podem transferir dados entre si. Com esses recursos implantados, os dados podem ser processados para gerar conhecimento e insights significativos. Há vários tipos de insights que podem ser gerados por meio da análise de dados, que são os insights:

- **Descritiva:** esse tipo de análise fornece detalhes sobre o que está acontecendo ou aconteceu no passado.
- **Preditiva:** esse tipo de análise fornece detalhes sobre o que vai acontecer no futuro.
- **Prescritiva:** esse tipo de análise fornece detalhes sobre o que deve ser feito para melhorar ou evitar eventos atuais ou futuros.
- **Cognitiva:** esse tipo de análise executa as ações determinadas pela análise prescritiva de maneira automatizada.

Obter insights de dados é bom, mas é igualmente importante tomar providências em relação a eles. O Azure fornece uma plataforma avançada para ingerir grandes volumes de dados, processá-los e transformá-los, armazenar e gerar insights a partir deles e exibi-los em painéis em tempo real. Também é possível tomar medidas em relação a esses insights automaticamente. Esses serviços estão disponíveis para todos os clientes do Azure e fornecem um amplo ecossistema no qual eles podem criar soluções. As empresas estão criando diversas aplicações e serviços que estão transformando completamente as indústrias devido à fácil disponibilidade desses serviços inteligentes do Azure, que são combinados para agregar um valor significativo para os clientes finais. O Azure garante que serviços cuja implementação era comercialmente inviável por pequenas e médias empresas agora possam ser consumidos e implantados prontamente em poucos minutos.

## Azure Resource Manager

**Azure Resource Manager (ARM)** é o serviço de orquestração e plataforma tecnológica da Microsoft que engloba todos os componentes abordados anteriormente. Ele reúne os provedores de recursos, os recursos e os grupos de recursos do Azure para formar uma plataforma de nuvem coesa. Ele disponibiliza os Serviços do Azure para assinaturas, tipos de recursos para os grupos de recursos, recursos e APIs de recursos para o portal e outros clientes, além de autenticar o acesso a esses recursos. Ele também habilita recursos como identificação, autenticação, **controle de acesso baseado em função (RBAC)**, bloqueio de recursos e aplicação de políticas para as assinaturas e seus grupos de recursos. Ele também fornece recursos de implantação e gerenciamento usando o portal do Azure, o Azure PowerShell e as ferramentas de **interface de linha de comando (CLI)**.

## Arquitetura do ARM

A arquitetura do ARM e seus componentes são mostrados na Figura 1.2. Como podemos ver, a **Assinatura do Azure** é composta por vários grupos de recursos. Cada grupo contém instâncias de recursos que são criadas a partir dos tipos de recursos disponíveis no provedor de recursos:

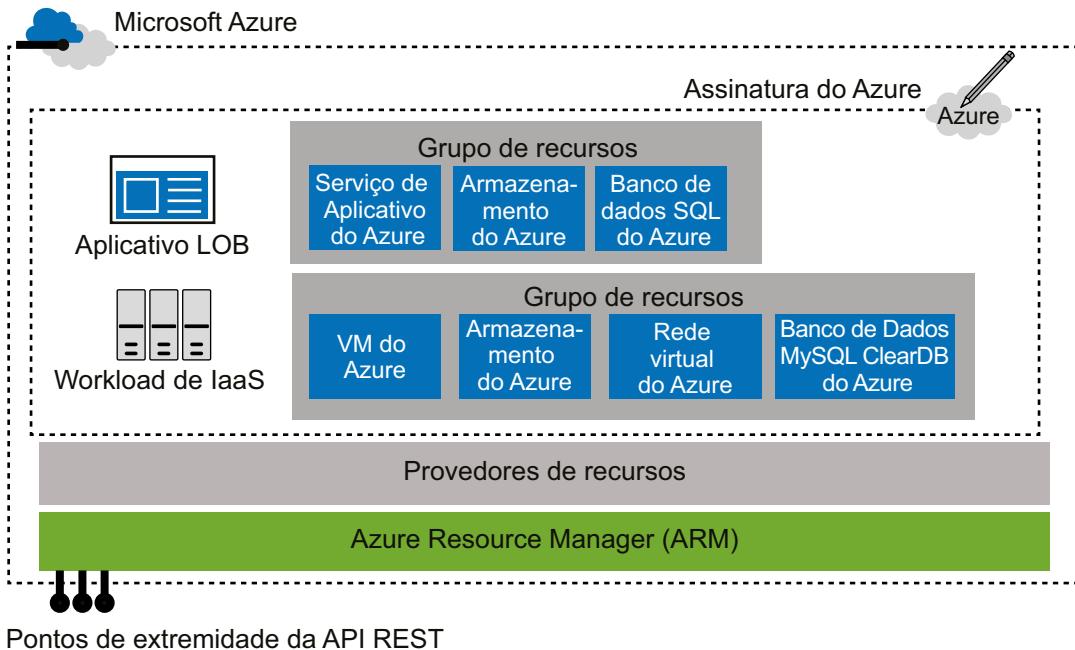


Figura 1.2: arquitetura do ARM

## Por que usar o ARM?

Antes do ARM, a estrutura usada pelo Azure era conhecida como **Gerenciador de Serviços do Azure (ASM)**. É importante apresentar uma pequena introdução para termos uma compreensão clara do surgimento do ARM e da substituição lenta e constante do ASM.

## Limitações do ASM

O ASM tem restrições inerentes. Por exemplo, as implantações do ASM são lentas e causam bloqueio. As operações serão bloqueadas se uma operação anterior já estiver em andamento. Algumas das limitações do ASM são:

- **Paralelismo:** o paralelismo é um desafio no ASM. Não é possível executar várias transações em paralelo com êxito. As operações no ASM são lineares e executadas uma após a outra. Se várias transações forem executadas ao mesmo tempo, haverá erros de operações paralelas ou as transações serão bloqueadas.
- **Recursos:** os recursos no ASM são provisionados e gerenciados isoladamente uns dos outros. Não existe relação entre os recursos do ASM. Não é possível agrupar serviços e recursos nem os configurar em conjunto.
- **Serviços de nuvem:** os serviços de nuvem são as unidades de implantação no ASM. Eles dependem de grupos de afinidade e não são escaláveis devido ao seu design e arquitetura.

Funções e permissões granulares e discretas não podem ser atribuídas aos recursos no ASM. Os clientes são administradores de serviços ou coadministradores na assinatura. Eles têm controle total sobre os recursos ou não têm acesso a eles. O ASM não fornece suporte para implantação. As duas implantações são feitas manualmente. Caso contrário, precisaremos recorrer à escrita de scripts de procedimentos no .NET ou PowerShell. As APIs do ASM não são consistentes entre os recursos.

## Vantagens do ARM

O ARM fornece as seguintes vantagens e benefícios distintos em relação ao ASM:

- **Agrupamento:** o ARM permite o agrupamento de recursos em um contêiner lógico. Esses recursos podem ser gerenciados em conjunto e passar por um ciclo de vida comum, como um grupo. Isso facilita a identificação de recursos relacionados e dependentes.
- **Ciclos de vida comuns:** os recursos de um grupo têm o mesmo ciclo de vida. Esses recursos podem evoluir e ser gerenciados em conjunto, como uma unidade.
- **RBAC:** funções e permissões granulares podem ser atribuídas a recursos, proporcionando acesso discreto para os clientes. Os clientes também podem ter somente os direitos que lhe são atribuídos.

- **Supporte para implantação:** o ARM fornece suporte para implantação em termos de modelos, habilitando o DevOps e a **infraestrutura como código (IaC)**. Essas implantações são mais rápidas, consistentes e previsíveis.
- **Tecnologia superior:** o custo e a cobrança dos recursos podem ser gerenciados como uma unidade. Cada grupo de recursos pode fornecer suas informações de uso e custo.
- **Capacidade de gerenciamento:** o ARM fornece recursos avançados, como segurança, monitoramento, auditoria e marcação, para melhor capacidade de gerenciamento de recursos. Os recursos podem ser consultados com base em marcas. As marcas também fornecem informações de custo e cobrança para recursos marcados de maneira semelhante.
- **Migração:** mais facilidade de migração e atualização de recursos internos e entre grupos de recursos.

## Conceitos do ARM

Com o ARM, tudo no Azure é um recurso. Exemplos de recursos são VMs, interfaces de rede, endereços IP públicos, contas de armazenamento e redes virtuais. O ARM é baseado em conceitos relacionados a provedores de recursos e a consumidores de recursos. O Azure fornece recursos e serviços por meio de vários provedores de recursos que são consumidos e implantados em grupos.

### Provedores de recursos

Esses são os serviços responsáveis por fornecer tipos de recursos por meio do ARM. O conceito de nível superior do ARM engloba o provedor de recursos. Esses provedores são contêineres de tipos de recursos. Os tipos de recursos são agrupados em provedores de recursos. Eles são responsáveis por implantar e gerenciar os recursos. Por exemplo, um tipo de recurso de VM é fornecido por um provedor de recursos chamado **Microsoft.Compute/virtualMachines**. As operações de APIs **Representational State Transfer (REST)** têm versões para distinguir entre elas. A nomenclatura das versões é baseada nas datas em que elas são lançadas pela Microsoft. É necessário que um provedor de recursos relacionado esteja disponível para uma assinatura para implantar um recurso. Nem todos os provedores de recursos estão disponíveis para uma assinatura de imediato. Se um recurso não estiver disponível para uma assinatura, precisaremos verificar se o provedor de recursos necessário está disponível em cada região. Se estiver disponível, o cliente poderá inscrever-se explicitamente na assinatura.

## Tipos de recursos

Os tipos de recursos são uma especificação real de recursos que define sua interface e implementação de API pública. Eles implementam o funcionamento e as operações com suporte do recurso. Semelhantes aos provedores de recursos, os tipos de recursos também evoluem ao longo do tempo, no que se refere à sua implementação interna, e têm várias versões de seus esquemas e das interfaces de API pública. Os nomes das versões são baseados nas datas em que elas são lançadas pela Microsoft como uma versão prévia ou **disponibilidade geral (GA)**. Os tipos de recursos tornam-se disponíveis como uma assinatura depois que um provedor de recursos é registrado nela. Além disso, nem todos os tipos de recursos estão disponíveis em todas as regiões do Azure. A disponibilidade de um recurso depende da disponibilidade e do registro de um provedor de recursos em uma região do Azure e deve dar suporte à versão da API necessária para o provisionamento dele.

## Grupos de recursos

Os grupos de recursos são unidades de implantação no ARM. Eles são contêineres que agrupam várias instâncias de recursos em um limite de gerenciamento e segurança. Um grupo de recursos recebe um nome exclusivo em uma assinatura. Os recursos podem ser provisionados em diferentes regiões do Azure e ainda pertencerem ao mesmo grupo de recursos. Os grupos de recursos fornecem serviços adicionais a todos os recursos deles. Os grupos de recursos fornecem serviços de metadados, como marcação, o que permite a classificação dos recursos, o gerenciamento de recursos com base em políticas, o RBAC, a proteção de recursos contra exclusão ou atualizações acidentais e muito mais. Como mencionado anteriormente, eles têm um limite de segurança, e os usuários que não têm acesso a um grupo de recursos não podem acessar os recursos contidos nele. Cada instância de recurso precisa fazer parte de um grupo de recursos. Caso contrário, ela não poderá ser implantada.

## Recursos e instâncias de recursos

Os recursos são criados a partir de tipos de recursos e são uma instância de um tipo de recurso. Uma instância pode ser globalmente exclusiva ou estar em um nível de grupo de recursos. A exclusividade é definida pelo nome e tipo do recurso. Se compararmos isso com construções de programação orientadas a objetos, as instâncias de recursos podem ser vistas como objetos, e os tipos de recursos podem ser vistos como classes. Os serviços são consumidos por meio das operações com suporte e implementados por instâncias de recursos. O tipo de recurso define propriedades, e cada instância deve configurar propriedades obrigatórias durante o provisionamento de uma instância. Algumas propriedades são obrigatórias, enquanto outras são opcionais. Elas herdam a configuração de segurança e acesso de seu grupo de recursos pai. Essas atribuições de funções e permissões herdadas podem ser substituídas para cada recurso. Um recurso pode ser bloqueado de forma que algumas de suas operações possam ser bloqueadas e disponibilizadas para funções, usuários e grupos, mesmo que eles tenham acesso ao recurso. Os recursos podem ser marcados para facilitar a detecção e o gerenciamento.

## Recursos do ARM

Veja alguns dos principais recursos fornecidos pelo ARM:

- **RBAC: Azure Active Directory (Azure AD)** autentica os usuários para fornecer acesso a assinaturas, grupos de recursos e recursos. O ARM implementa OAuth e RBAC na plataforma, permitindo a autorização e o controle de acesso a recursos, grupos de recursos e assinaturas com base nas funções atribuídas a um usuário ou grupo. Uma permissão define o acesso às operações em um recurso. Essas permissões podem permitir ou negar acesso ao recurso. Uma definição de função é uma coleção dessas permissões. As funções mapeiam usuários e grupos do Azure AD para permissões específicas. As funções são atribuídas posteriormente a um escopo, que pode ser um indivíduo, uma coleção de recursos, um grupo de recursos ou a assinatura. As identidades do Azure AD (usuários, grupos e princípios de serviço) adicionadas a uma função ganham acesso ao recurso de acordo com as permissões definidas na função. O ARM fornece várias funções prontas para uso. Ele fornece funções de sistema, como o **proprietário**, o **colaborador** e o **leitor**. Também fornece funções baseadas em recursos, como colaborador de BD SQL e colaborador de VM. O ARM também permite a criação de funções personalizadas.
- **Marcas:** as marcas são pares de nome–valor que adicionam mais informações e metadados aos recursos. Recursos e grupos de recursos podem ter várias marcas. As marcas ajudam na categorização dos recursos para melhorar a capacidade de descoberta e gerenciamento. Os recursos podem ser pesquisados rapidamente e identificados de forma fácil. Informações de cobrança e custos também podem ser buscadas para os recursos que têm as mesmas marcas. Embora esse recurso seja fornecido pelo ARM, um administrador de TI define seu uso e taxonomia em matéria de recursos e grupos de recursos. A taxonomia e as marcas, por exemplo, podem estar relacionadas a departamentos, uso de recursos, localização, projetos ou quaisquer outros critérios considerados adequados do ponto de vista de custo, uso, cobrança ou pesquisa. Essas marcas podem, então, ser aplicadas aos recursos. As marcas definidas no nível de grupo de recursos não são herdadas por seus recursos.
- **Políticas:** outro recurso de segurança fornecido pelo ARM são as políticas personalizadas. As políticas personalizadas podem ser criadas para controlar o acesso aos recursos. As políticas são definidas como regras e convenções e devem ser respeitadas durante a interação com os recursos e grupos de recursos. A definição de política contém uma negação explícita de ações nos recursos ou acesso aos recursos. Por padrão, todo acesso é permitido quando não é mencionado na definição de política. Essas definições de política são atribuídas ao escopo de recursos, grupos de recursos e assinaturas. É importante notar que essas políticas não substituem o RBAC (controle de acesso baseado em função). Na verdade, elas complementam e trabalham em conjunto com o RBAC. As políticas

são avaliadas depois que um usuário é autenticado pelo Azure AD e autorizado pelo serviço RBAC. O ARM fornece uma linguagem de definição de política baseada em JSON para definir políticas. Alguns exemplos de definições de política são que uma política deve marcar todos os recursos provisionados, e os recursos podem ser provisionados somente para regiões específicas do Azure.

- **Bloqueios:** assinaturas, grupos de recursos e recursos podem ser bloqueados para evitar exclusões ou atualizações acidentais por um usuário autenticado. Os bloqueios aplicados em níveis mais altos se propagam aos recursos filho. Como alternativa, os bloqueios aplicados no nível de assinatura bloqueiam todos os grupos de recursos e os recursos contidos neles.
- **Várias regiões:** o Azure fornece várias regiões para o provisionamento e a hospedagem de recursos. O ARM permite que recursos sejam provisionados em locais diferentes enquanto ainda residam no mesmo grupo de recursos. Um grupo pode conter recursos de regiões diferentes.
- **Idempotente:** esse recurso assegura a previsibilidade, padronização e consistência na implantação de recursos, garantindo que cada implantação resulte no mesmo estado de recursos e sua configuração, não importa quantas vezes o processo seja executado.
- **Extensível:** o ARM fornece uma arquitetura extensível para permitir a criação e conexão de novos provedores de recursos e tipos de recursos na plataforma.

## Virtualização

A virtualização foi uma inovação revolucionária que mudou completamente a maneira como os servidores físicos eram vistos. Refere-se à abstração de um objeto físico em um objeto lógico.

A virtualização de servidores físicos levou a servidores virtuais conhecidos como VMs. Essas VMs consomem e compartilham a CPU física, memória, armazenamento e outros componentes de hardware do servidor físico em que são hospedadas. Isso permite um provisionamento mais rápido e fácil de ambientes de aplicações sob demanda, fornecendo alta disponibilidade e escalabilidade com custo reduzido. Um servidor físico é suficiente para hospedar várias VMs, com cada VM contendo o próprio sistema operacional e serviços de hospedagem.

Não havia mais necessidade de comprar servidores físicos adicionais para implantar novas aplicações e serviços. Os servidores físicos existentes eram suficientes para hospedar mais VMs. Além disso, como parte da racionalização, o número de servidores físicos foi reduzido com a ajuda da virtualização.

Cada VM contém todo o sistema operacional e é completamente isolada de outras VMs, incluindo os hosts físicos. Embora uma VM use o hardware fornecido pelo servidor host físico, ela tem controle total sobre seus recursos atribuídos e seu ambiente. Essas VMs podem ser hospedadas em uma rede, como um servidor físico com sua própria identidade.

O Azure pode criar VMs do Linux e do Windows em alguns minutos. A Microsoft fornece suas próprias imagens, juntamente com imagens de seus parceiros e da comunidade; os usuários também podem fornecer suas próprias imagens. As VMs são criadas usando essas imagens.

## Contêineres

Contêineres também são uma tecnologia de virtualização. No entanto, eles não virtualizam um servidor. Na verdade, um contêiner é uma virtualização no nível do sistema operacional. Isso significa que os contêineres compartilham o kernel do sistema operacional (que é fornecido pelo host) entre si, juntamente com o host. Vários contêineres em execução em um host (físico ou virtual) compartilham o kernel do sistema operacional do host. Os contêineres garantem que eles reutilizem o kernel do host em vez de cada um ter um kernel dedicado a eles mesmos.

Os contêineres são completamente isolados de seu host ou de outros contêineres em execução no host. Os contêineres do Windows usam drivers de filtro de armazenamento e isolamento de sessão do Windows para isolar serviços do sistema operacional, como sistema de arquivos, registro, processos e redes. O mesmo acontece para contêineres do Linux em execução em hosts Linux. Os contêineres do Linux usam o namespace do Linux, os grupos de controle e o sistema de arquivos da união para virtualizar o sistema operacional do host.

O contêiner aparece como se tivesse um sistema operacional e recursos completamente novos e intocados. Esse arranjo fornece muitos benefícios. São eles:

- O provisionamento dos contêineres é rápido e leva menos tempo em comparação com as máquinas virtuais. A maioria dos serviços de sistemas operacionais em um contêiner é fornecida pelo sistema operacional do host.
- Os contêineres são leves e exigem menos recursos de computação do que as VMs. A sobrecarga de recursos do sistema operacional não é mais necessária com os contêineres.
- Os contêineres são muito menores do que as VMs.
- Os contêineres podem ajudar a resolver problemas relacionados ao gerenciamento de várias dependências de aplicações de maneira intuitiva, automatizada e simples.
- Os contêineres fornecem a infraestrutura para definir todas as dependências de aplicações em um único lugar.

Os contêineres são um recurso inerente do Windows Server 2016 e do Windows 10. No entanto, eles são gerenciados e acessados por meio de um cliente do Docker e um daemon do Docker. Os contêineres podem ser criados no Azure com uma SKU do Windows Server 2016 como imagem. Cada contêiner tem um único processo principal que deve estar em execução para o contêiner existir. Um contêiner será interrompido quando esse processo terminar. Além disso, um contêiner pode funcionar no modo interativo ou no modo desconectado como um serviço:



**Figura 1.3: arquitetura de contêiner**

A Figura 1.3 mostra todas as camadas técnicas que habilitam os contêineres. A camada inferior fornece a infraestrutura básica em termos de rede, armazenamento,平衡adores de carga e placas de rede. Acima da infraestrutura está a camada de computação, que consiste em um servidor físico ou em servidores físicos e virtuais acima de um servidor físico. Essa camada contém o sistema operacional com a capacidade de hospedar contêineres. O sistema operacional fornece o driver de execução que as camadas acima usam para chamar o código do kernel e os objetos para executar os contêineres. A Microsoft criou o **Host Container System Shim (HCSShim)** para gerenciar e criar contêineres e usa drivers de filtro de armazenamento do Windows para gerenciamento de imagens e arquivos.

O isolamento do ambiente de contêineres é habilitado para a sessão do Windows. O Windows Server 2016 e o Nano Server fornecem o sistema operacional, habilitam os recursos do contêiner e executam o cliente do Docker e o mecanismo do Docker no nível do usuário. O mecanismo do Docker usa os serviços do HCSShim, os drivers de filtro de armazenamento e as sessões para gerar vários contêineres no servidor, com cada um contendo um serviço, aplicação ou banco de dados.

## Docker

O Docker fornece recursos de gerenciamento a contêineres do Windows. Ele é composto pelos dois executáveis a seguir:

- Daemon do Docker
- Cliente do Docker

O daemon do Docker é útil para o gerenciamento de contêineres. É um serviço do Windows responsável por gerenciar todas as atividades no host relacionadas a contêineres. O cliente do Docker interage com o daemon do Docker e é responsável pela captura de entradas e pelo seu envio ao daemon do Docker. O daemon do Docker fornece tempo de execução, bibliotecas, drivers de gráficos e o mecanismo para criar, gerenciar e monitorar contêineres e imagens no servidor host. Ele também tem a capacidade de criar imagens personalizadas que são usadas para criar e enviar aplicações para vários ambientes.

## Interação com a nuvem inteligente

O Azure fornece várias maneiras de conectar, automatizar e interagir com a nuvem inteligente. Todos esses métodos exigem que usuários sejam autenticados com credenciais válidas para que possam ser usados. As diferentes maneiras de se conectar ao Azure são:

- O portal do Azure
- PowerShell
- A CLI do Azure
- A API REST do Azure

## O portal do Azure

O portal do Azure é um ótimo lugar para começar. Com o portal do Azure, os usuários podem fazer logon e começar a criar e gerenciar recursos do Azure manualmente. O portal fornece uma interface de usuário intuitiva e fácil de usar por meio do navegador. O portal do Azure fornece uma maneira fácil de navegar nos recursos usando **folhas**. As folhas exibem todas as propriedades de um recurso, incluindo seus logs, custo, relação com outros recursos, marcas, opções de segurança e muito mais. Uma implantação inteira da nuvem pode ser gerenciada por meio do portal.

## PowerShell

O PowerShell é um shell de linha de comando baseado em objeto e linguagem de script usado para a administração, a configuração e o gerenciamento de infraestrutura e ambientes. Ele é baseado no .NET Framework e fornece recursos de automação.

O PowerShell realmente se tornou um cidadão de primeira classe entre administradores de TI e desenvolvedores de automação para gerenciar e controlar o ambiente Windows. Hoje, quase todos os ambientes Windows e muitos ambientes Linux podem ser gerenciados pelo PowerShell. Na verdade, quase todos os aspectos do Azure também podem ser gerenciados pelo PowerShell. O Azure fornece suporte avançado para o PowerShell. Ele fornece um módulo do PowerShell para cada provedor de recursos contendo centenas de cmdlets. Os usuários podem usar esses cmdlets em seus scripts para automatizar a interação com o Azure. O módulo do Azure PowerShell está disponível por meio do instalador da plataforma Web e pela **Galeria do PowerShell**.

O Windows Server 2016 e o Windows 10 fornecem gerenciamento de pacotes e módulos **PowerShellGet** para downloads e instalações rápidos e fáceis de módulos do PowerShell da Galeria do PowerShell. O módulo **PowerShellGet** fornece o cmdlet **Install-Module** para fazer o download e instalar módulos no sistema.

A instalação de um módulo é o simples ato de copiar os arquivos do módulo em locais de módulos bem definidos, o que pode ser feito da seguinte forma:

```
Import-module PowerShellGet  
Install-Module -Name az -verbose
```

O comando **Import-module** importa um módulo e suas funções relacionadas dentro do atual escopo de execução, e o **Install-Module** ajuda na instalação de módulos.

## A CLI do Azure

O Azure também fornece o Azure CLI 2.0, que pode ser implantado em sistemas operacionais Linux, Windows e macOS. O Azure CLI 2.0 é o novo utilitário de linha de comando do Azure para gerenciar os recursos do Azure. O Azure CLI 2.0 é otimizado para gerenciar e administrar recursos do Azure a partir da linha de comando e para criar scripts de automação que funcionam com o ARM. A CLI pode ser usada para executar comandos usando o shell do Bash ou a linha de comando do Windows.

A CLI do Azure é muito famosa entre usuários que não são do Windows, pois permite interagir com o Azure no Linux e no macOS. As etapas para instalação do Azure CLI 2.0 estão disponíveis em <https://docs.microsoft.com/cli/azure/install-azure-cli?view=azure-cli-latest>.

## A API REST do Azure

Todos os recursos do Azure são expostos aos usuários por meio de pontos de extremidade REST. As APIs REST são pontos de extremidade de serviço que implementam operações (ou métodos) HTTP, fornecendo acesso de **criação, recuperação, atualização ou exclusão (CRUD)** aos recursos do serviço. Os usuários podem consumir estas APIs para criar e gerenciar recursos. Na verdade, a CLI e os mecanismos PowerShell usam essas APIs REST internamente para interagir com recursos no Azure.

## Modelos do ARM

Em uma seção anterior, analisamos os recursos de implantação, como vários serviços, várias regiões, extensível e idempotente, fornecidos pelo ARM. Os modelos do ARM são os principais meios de provisionamento de recursos no ARM. Os modelos do ARM fornecem suporte de implementação para os recursos de implantação do ARM.

Eles fornecem um modelo declarativo por meio do qual recursos, suas configurações, scripts e extensões são especificados. Os modelos do ARM se baseiam no formato **JavaScript Object Notation (JSON)**. Eles usam as convenções e a sintaxe JSON para declarar e configurar recursos. Arquivos JSON são baseados em texto, fáceis de usar e de ler.

Eles podem ser armazenados em um repositório de código-fonte e submetidos ao controle de versão. Eles também servem para representar a IaC que pode ser usada para provisionar recursos em um grupo de recursos do Azure de forma constante, previsível e uniforme. Um modelo precisa de um grupo de recursos para implantação. Ele só pode ser implantado em um grupo de recursos, que deve existir antes de executar a implantação de um modelo. Um modelo não é capaz de criar um grupo de recursos.

Os modelos fornecem a flexibilidade de serem genéricos e modulares no seu design e na sua implementação. Eles fornecem a capacidade de aceitar parâmetros de usuários, declarar variáveis internas, definir dependências entre recursos, vincular recursos dentro do mesmo grupo de recursos ou em grupos de recursos diferentes, além de executar outros modelos. Eles também fornecem expressões e funções do tipo de linguagem de script que os tornam dinâmicos e personalizáveis no tempo de execução.

## Implantações

O PowerShell permite os dois modos a seguir para a implantação de modelos:

- **Incremental:** a implantação incremental adiciona recursos declarados no modelo que não existem em um grupo de recursos e deixa os recursos inalterados em um grupo que não faz parte de uma definição de modelo e em um grupo existente no modelo e no grupo com o mesmo estado de configuração.
- **Completa:** a implantação completa, por outro lado, adiciona recursos declarados em um modelo ao grupo de recursos, exclui recursos que não existem no modelo do grupo e deixa inalterados os recursos que existem no grupo e no modelo com o mesmo estado de configuração.

## Resumo

A nuvem é um paradigma relativamente novo e ainda está em fase incipiente.

Haverá muita inovação e recursos adicionados ao longo do tempo. O Azure é um dos principais provedores de nuvem do momento e fornece recursos avançados por meio de implantações de IaaS, PaaS, SaaS e híbridas. Na verdade, o Azure Stack, que é uma implementação da nuvem privada da Microsoft, será lançado em breve. Ele terá os mesmos recursos que estão disponíveis em uma nuvem privada na nuvem pública. Ambas poderão, de fato, se conectar e funcionar em conjunto de forma transparente e integrada.

É muito fácil começar a usar o Azure, mas os desenvolvedores e arquitetos também podem cair em uma armadilha se não projetarem e arquitetarem suas soluções adequadamente. Este livro é uma tentativa de fornecer orientações e instruções para arquitetar soluções da maneira certa, usando os serviços e recursos adequados. Cada serviço no Azure é um recurso. É importante compreender como esses recursos são organizados e gerenciados no Azure. Este capítulo apresentou o contexto do ARM e dos grupos, que são as estruturas básicas que fornecem os elementos básicos dos recursos. O ARM oferece um conjunto de serviços para recursos que ajudam a fornecer uniformidade, padronização e consistência no gerenciamento deles. Os serviços, como o RBAC, marcas, políticas e bloqueios, estão disponíveis para todos os recursos e provedores de recursos. O Azure também fornece funcionalidades avançadas de automação para automatizar e interagir com recursos. Ferramentas como o PowerShell, modelos do ARM e a CLI do Azure podem ser incorporadas como parte dos pipelines de lançamento e da implantação e entrega contínuas. Os usuários podem se conectar ao Azure a partir de ambientes heterogêneos usando essas ferramentas de automação.

O próximo capítulo abordará algumas das importantes preocupações de arquitetura que ajudam a resolver problemas comuns da implantação baseada na nuvem e a garantir que as aplicações sejam seguras, disponíveis, escaláveis e de fácil manutenção no longo prazo.





# 2

## Disponibilidade, escalabilidade e monitoramento da solução do Azure

As preocupações com arquitetura, como alta disponibilidade e escalabilidade, são alguns dos itens de maior prioridade para qualquer arquiteto. Isso é comum em muitos projetos e soluções. No entanto, isso se torna ainda mais importante ao implantar aplicações na nuvem devido à complexidade envolvida. Na maioria das vezes, a complexidade não vem da aplicação, mas das opções disponíveis em termos de recursos semelhantes na nuvem. O outro problema complexo que surge da nuvem é a disponibilidade constante de novos recursos. Esses novos recursos podem tornar as decisões de um arquiteto completamente redundantes em retrospectiva.

Neste capítulo, analisaremos a perspectiva de um arquiteto em termos de implantação de aplicações altamente disponíveis e escalonáveis no Azure.

O Azure é uma plataforma madura que fornece várias opções para implementar alta disponibilidade e escalabilidade em vários níveis. Para um arquiteto, é essencial conhecê-las, incluindo as diferenças entre elas e os custos envolvidos, e estar em uma posição de escolher uma solução adequada que atenda aos melhores requisitos. Não há uma única solução para tudo, mas há uma boa solução para cada projeto.

A execução de aplicações e sistemas disponíveis para os usuários para consumo sempre que eles precisam é uma das principais prioridades das organizações. Eles querem que suas aplicações sejam operacionais e funcionais e continuem disponíveis para os clientes mesmo que eventos desagradáveis aconteçam. A alta disponibilidade é o tema principal deste capítulo. Manter as luzes acesas é a metáfora comum usada para alta disponibilidade. Alcançar a alta disponibilidade para aplicações não é uma tarefa fácil, e as organizações precisam gastar tempo, energia, recursos e dinheiro consideráveis para isso. Além disso, ainda há o risco de a implementação de uma organização não produzir os resultados desejados. O Azure fornece muitos recursos de alta disponibilidade para **máquinas virtuais (VMs)** e o serviço de **Plataforma como Serviço (PaaS)**. Neste capítulo, veremos os recursos de arquitetura e design fornecidos pelo Azure para garantir a alta disponibilidade para executar aplicações e serviços.

Neste capítulo, abordaremos os seguintes tópicos:

- Alta disponibilidade
- Alta disponibilidade do Azure
- Considerações sobre arquitetura para alta disponibilidade
- Escalabilidade
- Atualizações e manutenção

## Alta disponibilidade

A alta disponibilidade é um dos requisitos técnicos não funcionais essenciais para qualquer serviço crítico para os negócios e sua implantação. A alta disponibilidade refere-se ao recurso de um serviço ou aplicação que os mantém operacionais continuamente. Para fazer isso, ela atende ou supera o **acordo de nível de serviço (SLA)** prometido. Os usuários recebem a promessa de determinado SLA com base no tipo de serviço. O serviço deve estar disponível para utilização com base em seu respectivo SLA. Por exemplo, um SLA pode definir 99% de disponibilidade de uma aplicação para o ano inteiro. Isso significa que ele deve estar disponível para os usuários 361,35 dias por ano. Se ele não permanecer disponível para esse período, isso constitui uma violação do SLA. A maioria das aplicações de missão crítica define seu SLA de alta disponibilidade como 99,999% por um ano. Isso significa que a aplicação deve estar ativa, em execução e disponível o ano todo, mas só pode ficar inativa e indisponível por 5,2 horas. Se o tempo de inatividade for além disso, você estará qualificado para crédito, que será calculado com base na porcentagem de tempo de atividade total.

É importante observar que a alta disponibilidade é definida em termos de tempo (anual, mensal, semanal ou uma combinação desses períodos).

Uma aplicação ou serviço possui vários componentes, que são implantados em camadas e níveis separados. Além disso, um serviço ou aplicação é implantado em um **sistema operacional (SO)** e hospedado em um computador físico ou VM. Ele consome serviços de rede e armazenamento para diversas finalidades. Tal aplicação ou serviço pode até mesmo depender de sistemas externos. Para que esses serviços ou aplicações sejam altamente disponíveis, é importante que as redes, o armazenamento, os sistemas operacionais, os computadores físicos ou as VMs e todos os componentes da aplicação sejam desenvolvidos considerando o SLA e a alta disponibilidade. Um processo de ciclo de vida da aplicação definido é usado para garantir que a alta disponibilidade tenha respaldo desde o início do planejamento da aplicação até sua introdução às operações. Isso também envolve a introdução da redundância. Os recursos redundantes devem ser incluídos na arquitetura geral da implantação e da aplicação para garantir que, se um recurso ficar inativo, o outro assumirá e atenderá às solicitações do cliente.

Alguns dos principais fatores que afetam a alta disponibilidade de uma aplicação são:

- Manutenção planejada
- Manutenção não planejada
- Arquitetura de implantação de aplicação

Veremos cada um desses fatores nas seções a seguir. Vamos dar uma olhada mais de perto em como a alta disponibilidade é assegurada para implantações no Azure.

## Alta disponibilidade do Azure

É difícil obter alta disponibilidade e atender aos altos requisitos do SLA. O Azure fornece muitos recursos que permitem alta disponibilidade de aplicações, do sistema operacional host e convidado para aplicações que usam PaaS. Os arquitetos podem usar esses recursos para obter alta disponibilidade em suas aplicações usando a configuração, em vez de criar esses recursos do zero ou depender de ferramentas de terceiros.

Nesta seção, veremos as funcionalidades e os recursos fornecidos pelo Azure para tornar as aplicações altamente disponíveis. Antes de entrarmos em detalhes de arquitetura e configuração, é importante compreender os conceitos relacionados à alta disponibilidade do Azure.

## Conceitos

Os conceitos fundamentais fornecidos pelo Azure para obter alta disponibilidade são:

- Conjuntos de disponibilidade
- O domínio de falha
- O domínio de atualização
- Zonas de disponibilidade

Como você sabe, é muito importante criar soluções altamente disponíveis. Os workloads podem ser de missão crítica e exigem arquitetura altamente disponível. Vamos conferir mais de perto cada um dos conceitos de alta disponibilidade no Azure agora. Vamos começar com os conjuntos de disponibilidade.

### Conjuntos de disponibilidade

No Azure, a alta disponibilidade é obtida principalmente por meio de redundância. Redundância significa que há mais de uma instância do recurso do mesmo tipo que assume o controle em caso de falha do recurso principal. No entanto, só o fato de ter mais recursos semelhantes não os torna altamente disponíveis. Por exemplo, pode haver várias VMs provisionadas dentro de uma assinatura, mas esse fato por si só não as torna altamente disponíveis. O Azure fornece um recurso conhecido como conjunto de disponibilidade, e ter várias VMs associadas a ele o torna altamente disponível. Para que o conjunto de disponibilidade se torne altamente disponível, é necessário hospedar no mínimo duas VMs nele. Todas as VMs no conjunto de disponibilidade se tornam altamente disponíveis porque são colocadas em racks físicos separados no datacenter do Azure. Durante as atualizações, essas VMs são atualizadas uma por vez, e não todas ao mesmo tempo. Para isso, os conjuntos de disponibilidade fornecem domínios de falha e de atualização. Falaremos mais sobre isso na próxima seção. Resumindo, os conjuntos de disponibilidade fornecem redundância no nível do datacenter, semelhante ao armazenamento localmente redundante.

É importante observar que os conjuntos de disponibilidade fornecem alta disponibilidade dentro de um datacenter. Se um datacenter inteiro estiver indisponível, a disponibilidade da aplicação será afetada. Para garantir que as aplicações continuem disponíveis mesmo quando um datacenter ficar indisponível, o Azure introduziu um novo recurso conhecido como zonas de disponibilidade, que conhceremos em breve.

Se você se lembrar da lista de conceitos fundamentais, o próximo na lista será o domínio de falha. O domínio de falha é muitas vezes denotado pelo acrônimo FD. Na próxima seção, vamos discutir o que é o FD e como ele é relevante ao criar soluções altamente disponíveis.

## O domínio de falha

**Os domínios de falha (FDs)** representam um grupo de VMs que compartilham a mesma fonte de energia e comutador de rede. Quando uma VM é provisionada e atribuída a um conjunto de disponibilidade, ela é hospedada em um FD. Cada conjunto de disponibilidade tem dois ou três FDs, dependendo da região do Azure. Algumas regiões fornecem dois, enquanto outras fornecem três FDs em um conjunto de disponibilidade. Os FDs não podem ser configurados pelos usuários.

Quando várias VMs são criadas, elas são colocadas em FDs separados. Se o número de VMs for maior do que o de FDs, as VMs adicionais serão colocadas em FDs existentes. Por exemplo, se houver cinco VMs, haverá FDs hospedados em mais de uma VM.

Os FDs estão relacionados aos racks físicos no datacenter do Azure. Os FDs fornecem alta disponibilidade em caso de tempo de inatividade não planejado devido a falhas de hardware, energia e rede. Como cada VM é colocada em um rack diferente com hardware, fonte de alimentação e rede diferentes, outras VMs continuarão sendo executadas se um rack for desligado.

O próximo na lista é o domínio de atualização.

## O domínio de atualização

Um FD cuida do tempo de inatividade não planejado. Já um domínio de atualização gerencia o tempo de inatividade da manutenção planejada. Cada VM também é atribuída a um domínio de atualização, e todas as VMs dentro desse domínio de atualização serão reinicializadas juntas. Pode haver mais de 20 domínios de atualização em um único conjunto de disponibilidade. Os domínios de atualização não podem ser configurados pelos usuários. Quando várias VMs são criadas, elas são colocadas em domínios de atualização separados. Se mais de 20 VMs forem provisionadas em um conjunto de disponibilidade, elas serão colocadas em um modo de distribuição alternada nesses domínios de atualização. Os domínios de atualização cuidam da manutenção planejada. Em **Integridade do serviço** no portal do Azure, você pode verificar os detalhes da manutenção planejada e definir alertas.

Na próxima seção, falaremos sobre as zonas de disponibilidade.

## Zonas de disponibilidade

Esse é um conceito relativamente novo introduzido pelo Azure. Ele é muito semelhante à redundância de zonas de contas de armazenamento. As zonas de disponibilidade fornecem alta disponibilidade em uma região, colocando instâncias de VM em datacenters separados dentro da região. As zonas de disponibilidade são aplicáveis a muitos recursos no Azure, incluindo VMs, discos gerenciados, conjuntos de escalas de VM e平衡adores de carga. Veja a lista completa de recursos compatíveis com a zona de disponibilidade em <https://docs.microsoft.com/azure/availability-zones/az-overview#services-that-support-availability-zones>. A impossibilidade de configurar a disponibilidade entre zonas foi uma falha no Azure por muito tempo. Isso foi corrigido com a introdução das zonas de disponibilidade.

Cada região do Azure comprehende vários datacenters equipados com energia, resfriamento e rede independentes. Algumas regiões têm mais datacenters, enquanto outras têm menos. Esses datacenters dentro da região são conhecidos como zonas. Para garantir a resiliência, há um mínimo de três zonas separadas em todas as regiões habilitadas. Implantar VMs em uma zona de disponibilidade garante que essas VMs estejam em datacenters, racks e redes diferentes. Esses datacenters em uma região se relacionam com redes de alta velocidade e não há atraso na comunicação entre essas VMs. A Figura 2.1 mostra como as zonas de disponibilidade são configuradas em uma região:

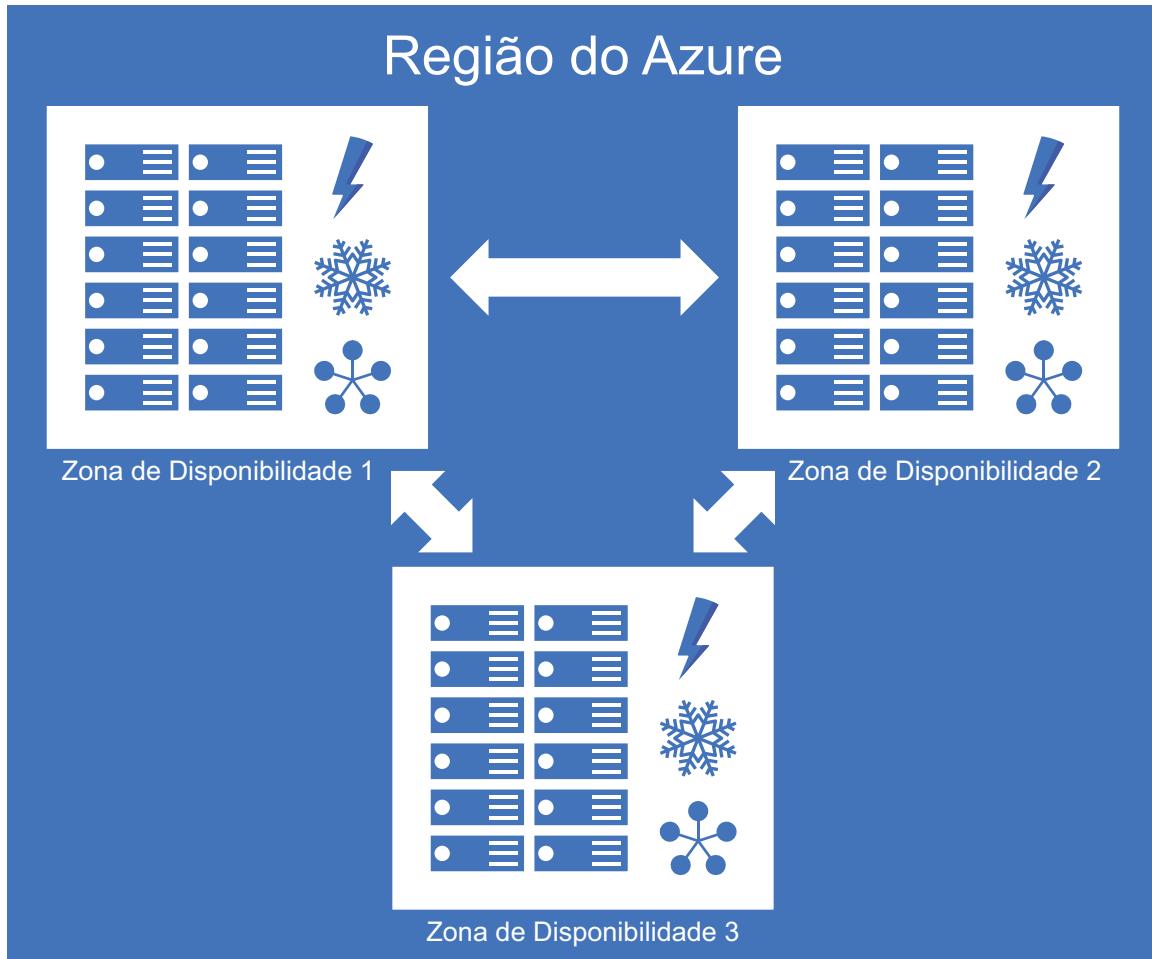


Figura 2.1: zonas de disponibilidade em uma região

Você pode encontrar mais informações sobre as zonas de disponibilidade em <https://docs.microsoft.com/azure/availability-zones/az-overview>.

Os serviços de redundância de zona replicam suas aplicações e dados em zonas de disponibilidade para proteger contra pontos únicos de falha.

Se uma aplicação precisar de maior disponibilidade e você quiser garantir a disponibilidade mesmo que uma região inteira do Azure esteja inativa, o próximo degrau da escada para a disponibilidade será o recurso Gerenciador de tráfego, que será discutido posteriormente neste capítulo. Vamos entender agora como o Azure assume o balanceamento de carga para VMs.

## Balanceamento de carga

O balanceamento de carga, como o nome sugere, refere-se ao processo de balanceamento de uma carga entre VMs e aplicações. Com uma VM, não há necessidade de balanceador porque a carga inteira está em uma única VM, e não há outra para compartilhá-la. No entanto, com várias VMs contendo a mesma aplicação e serviço, é possível distribuir a carga entre elas por meio do balanceamento de carga. O Azure fornece alguns recursos para viabilizar o balanceamento de carga:

- **Balanceadores de carga:** o balanceador de carga do Azure ajuda a criar soluções com alta disponibilidade. Na pilha do **Protocolo de Controle de Transmissão (TCP)** há um balanceador de carga no nível de transporte da camada 4. Ele é um balanceador de carga da camada 4 que distribui o tráfego de entrada entre instâncias íntegras dos serviços definidos em um conjunto de balanceamento de carga. Os平衡adores de carga de nível 4 funcionam no nível de transporte e têm informações de rede, como endereço IP e porta, para decidir o destino da solicitação de entrada. Os balanceadores de carga são explicados em mais detalhes posteriormente neste capítulo.
- **Gateways de aplicação:** o gateway de aplicação do Azure proporciona alta disponibilidade para suas aplicações. Eles são平衡adores de carga de camada 7 que distribuem o tráfego de entrada entre instâncias íntegras de serviços. Os平衡adores de carga de nível 7 podem funcionar no nível da aplicação e possuem informações sobre ela, como cookies, HTTP, HTTPS e sessões da solicitação de entrada. Os gateways de aplicação são explicados em mais detalhes posteriormente neste capítulo. Eles também são usados ao implantar os serviços do Kubernetes do Azure especificamente para cenários em que o tráfego de ingresso da Internet deve ser roteado para os serviços do Kubernetes no cluster.
- **Azure Front Door:** o Azure Front Door é muito semelhante aos gateways de aplicação. No entanto, ele não funciona no nível da região ou do datacenter. Em vez disso, ele ajuda no roteamento de solicitações entre regiões globalmente. Ele tem o mesmo conjunto de recursos que o oferecido pelos gateways de aplicação, mas no nível global. Ele também fornece um firewall de aplicação Web para a filtragem de solicitações e outra proteção relacionada à segurança. Ele fornece afinidade de sessão, terminação de TLS e roteamento baseado em URL como alguns de seus recursos.
- **O Gerenciador de Tráfego:** ajuda no roteamento de solicitações no nível global em várias regiões com base na integridade e na disponibilidade de pontos de extremidade regionais. Ele permite fazer isso usando entradas de redirecionamento de DNS. Ele também é altamente resiliente e não causa nenhum impacto no serviço durante falhas regionais.

Já que exploramos os métodos e serviços que podem ser usados para alcançar o balanceamento de carga, vamos discutir como tornar as VMs altamente disponíveis.

## Alta disponibilidade de VMs

As VMs fornecem recursos de computação. Elas fornecem poder de processamento e hospedagem para aplicações e serviços. Se uma aplicação for implantada em uma única VM e ela estiver inativa, ela não ficará disponível. Se a aplicação for composta por várias camadas e cada uma delas for implantada em sua própria instância única de uma VM, até mesmo um tempo de inatividade em uma única instância da VM poderá tornar toda a aplicação indisponível. O Azure tenta tornar até mesmo as instâncias de VM única altamente disponíveis 99,9% do tempo, especialmente quando essas VMs de instância única usam armazenamento Premium para seus discos. O Azure fornece um SLA superior para as VMs agrupadas em um conjunto de disponibilidade. Ele fornece um SLA de 99,95% para VMs que fazem parte de um conjunto de disponibilidade com duas ou mais VMs. O SLA será de 99,99% se as VMs forem colocadas em zonas de disponibilidade. Na próxima seção, discutiremos a alta disponibilidade para recursos de computação.

## Alta disponibilidade de computação

As aplicações que exigem alta disponibilidade devem ser implantadas em várias VMs no mesmo conjunto de disponibilidade. Se as aplicações forem compostas por várias camadas, cada camada deverá ter um grupo de VMs em seu conjunto de disponibilidade dedicado. Resumindo, se houver três camadas de uma aplicação, deverá haver três conjuntos de disponibilidade e no mínimo seis VMs (duas em cada conjunto de disponibilidade) para tornar toda a aplicação altamente disponível.

Então, como o Azure fornece um SLA e alta disponibilidade para VMs em um conjunto de disponibilidade com várias VMs em cada conjunto de disponibilidade? Essa é a pergunta que pode vir à sua mente.

Aqui, o uso de conceitos que consideramos antes entra em jogo – ou seja, os domínios de falha e atualização. Quando o Azure vê várias VMs em um conjunto de disponibilidade, ela coloca essas VMs em um FD separado. Em outras palavras, essas VMs são colocadas em racks físicos separados, e não no mesmo rack. Isso garante que pelo menos uma VM continue disponível mesmo que haja uma falha de energia, hardware ou rack. Há dois ou três FDs em um conjunto de disponibilidade e, dependendo do número de VMs em um conjunto de disponibilidade, as VMs são colocadas em FDs separados ou repetidas em um modo de distribuição alternada. Isso garante que a alta disponibilidade não seja afetada pela falha do rack.

O Azure também coloca essas VMs em um domínio de atualização separado. Em outras palavras, o Azure marca essas VMs internamente de modo que elas sejam corrigidas e atualizadas uma após a outra e que qualquer reinicialização em um domínio de atualização não afete a disponibilidade da aplicação. Isso garante que a alta disponibilidade não seja afetada pela manutenção da VM e do host. É importante observar que o Azure não é responsável pela manutenção do nível do sistema operacional e da aplicação.

Com a colocação de VMs em domínios de falha e atualização separados, o Azure garante que nem todas elas fiquem inativas ao mesmo tempo e permaneçam ativas e disponíveis para atender às solicitações, mesmo que estejam em manutenção ou enfrentando desafios de inatividade física.

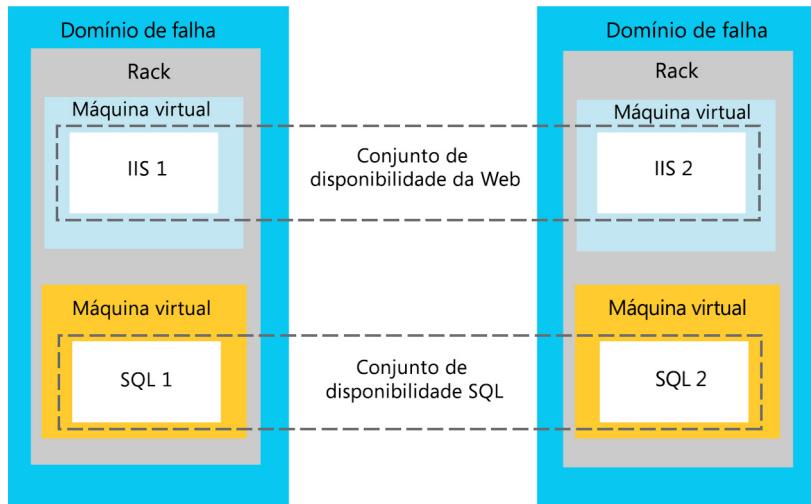


Figura 2.2: distribuição de VMs entre domínios de falha e atualização

A Figura 2.2 mostra quatro VMs (duas têm **Serviços de Informações da Internet (IIS)** e as outras duas têm o SQL Server instalado). As VMs IIS e SQL fazem parte de conjuntos de disponibilidade. As VMs IIS e SQL estão em FDs separados e em racks diferentes no datacenter. Elas também estão em domínios de atualização separados.

A Figura 2.3 mostra a relação entre os domínios de falha e atualização:

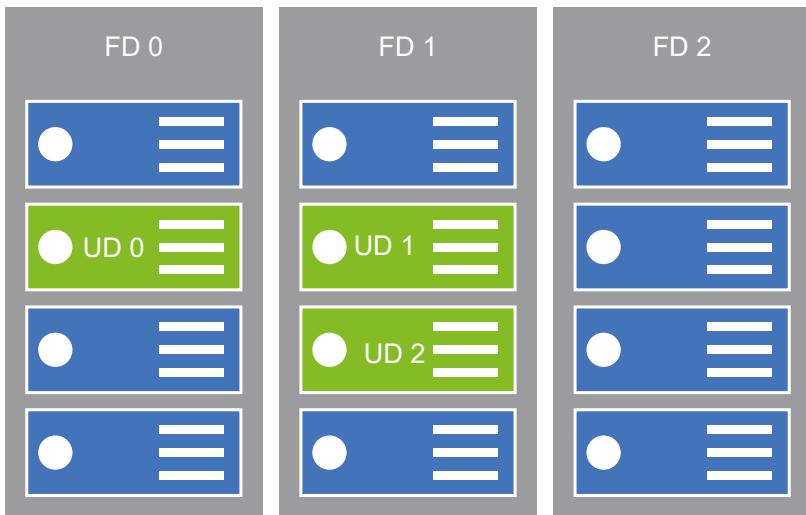


Figura 2.3: layout dos domínios de atualização e FDs em um conjunto de disponibilidade

Até agora, discutimos a obtenção de alta disponibilidade para recursos de computação. Na próxima seção, você verá como a alta disponibilidade pode ser implementada para PaaS.

## Plataformas de alta disponibilidade

O Azure forneceu muitos novos recursos para garantir a alta disponibilidade para PaaS. Alguns deles são listados aqui:

- Contêineres em serviços de aplicativo
- Grupos de instâncias de contêiner do Azure
- Serviço Azure Kubernetes
- Outros orquestradores de contêiner, como DC/OS e Swarm

Outra plataforma importante que traz alta disponibilidade é o **Service Fabric**. Tanto o Service Fabric quanto os orquestradores de contêiner que incluem o Kubernetes garantem que o número desejado de instâncias de aplicações esteja sempre funcionando em um ambiente. Portanto, mesmo que uma das instâncias fique inativa no ambiente, o orquestrador saberá disso por meio de monitoramento ativo e irá gerar uma nova instância em outro nó, mantendo assim o número ideal e desejado de instâncias.

Ele faz isso sem qualquer interferência manual ou automatizada do administrador.

Enquanto o Service Fabric permite que qualquer tipo de aplicação se torne altamente disponível, os orquestradores, como o Kubernetes, DC/OS e Swarm, são específicos dos contêineres. Além disso, é importante entender que essas plataformas fornecem recursos que ajudam a lançar atualizações, em vez de uma grande atualização de banco de dados que pode afetar a disponibilidade da aplicação.

Quando estávamos discutindo a alta disponibilidade para VMs, vimos rapidamente o balanceamento de carga. Vamos dar uma olhada mais de perto para entender melhor como ele funciona no Azure.

## Balanceadores de carga no Azure

O Azure fornece dois recursos que têm a funcionalidade de um balanceador de carga. Ele fornece um balanceador de carga de nível 4 que funciona na camada de transporte dentro da pilha TCP OSI e outro de nível 7 (gateway de aplicação) que funciona nos níveis da aplicação e da sessão.

Embora os gateways de aplicação e os balanceadores de carga forneçam os recursos básicos para balancear a carga, eles têm finalidades diferentes. Há diversos casos de uso em que faz mais sentido implantar um gateway de aplicação, em vez de um balanceador de carga.

O gateway de aplicação fornece os seguintes recursos que não estão disponíveis com os balanceadores de carga do Azure:

- **Firewall da Aplicação Web:** é um firewall adicional, além do firewall do sistema operacional, e permite inspecionar as mensagens recebidas. Isso ajuda a identificar e evitar ataques comuns na Web, como injeção de SQL, ataques de script entre sites e sequestros de sessão.
- **Afinidade de sessão baseada em cookies:** os平衡adores de carga distribuem o tráfego de entrada para instâncias de serviço que estão íntegras e relativamente livres. Uma solicitação pode ser atendida por qualquer instância de serviço. No entanto, há aplicações que precisam de recursos avançados em que todas as solicitações subsequentes à primeira devem ser processadas pela mesma instância de serviço. Isso é conhecido como afinidade de sessão baseada em cookies. O gateway de aplicação fornece afinidade de sessão baseada em cookies para manter uma sessão de usuário na mesma instância de serviço usando cookies.
- **Descarga SSL (Secure Sockets Layer):** a criptografia e a descriptografia de dados de solicitação e resposta são executadas por SSL e geralmente são caras. O ideal é que os servidores Web gastem recursos no processamento e atendimento das solicitações, e não na criptografia e descriptografia do tráfego. A descarga SSL ajuda a transferir esse processo de criptografia do servidor Web para o balanceador de carga, fornecendo assim mais recursos aos servidores Web que atendem aos usuários. A solicitação do usuário é criptografada, mas é descriptografada no gateway de aplicação, e não no servidor Web. A solicitação do gateway de aplicação para o servidor Web não é criptografada.
- **SSL de ponta a ponta:** embora a descarga SSL seja um bom recurso para determinadas aplicações, há certas aplicações seguras de missão crítica que precisam de criptografia e descriptografia SSL completas, mesmo que o tráfego passe por平衡adores de carga. O gateway de aplicação também pode ser configurado para uma criptografia SSL de ponta a ponta.
- **Roteamento de conteúdo baseado em URL:** os gateways de aplicação também são úteis para redirecionar o tráfego para servidores diferentes, com base no conteúdo do URL das solicitações de entrada. Isso ajuda na hospedagem de vários serviços junto com outras aplicações.

## Balanceadores de carga do Azure

Um balanceador de carga do Azure distribui o tráfego de entrada com base nas informações de nível de transporte disponíveis para isso. Ele depende dos seguintes recursos:

- Um endereço IP de origem
- Um endereço IP de destino
- Um número da porta de origem
- Um número da porta de destino
- Um tipo de protocolo – TCP ou HTTP

Um balanceador de carga do Azure pode ser um balanceador de carga privado ou público. Um balanceador de carga privado pode ser usado para distribuir o tráfego dentro da rede interna. Como isso é interno, não haverá IPs públicos atribuídos, e eles não podem ser acessados pela Internet. Um balanceador de carga público tem um IP público externo anexado a ele e pode ser acessado por meio da Internet. Na Figura 2.4, você pode ver como os平衡adores de carga internos (privados) e públicos são incorporados a uma única solução para lidar com o tráfego interno e externo, respectivamente:

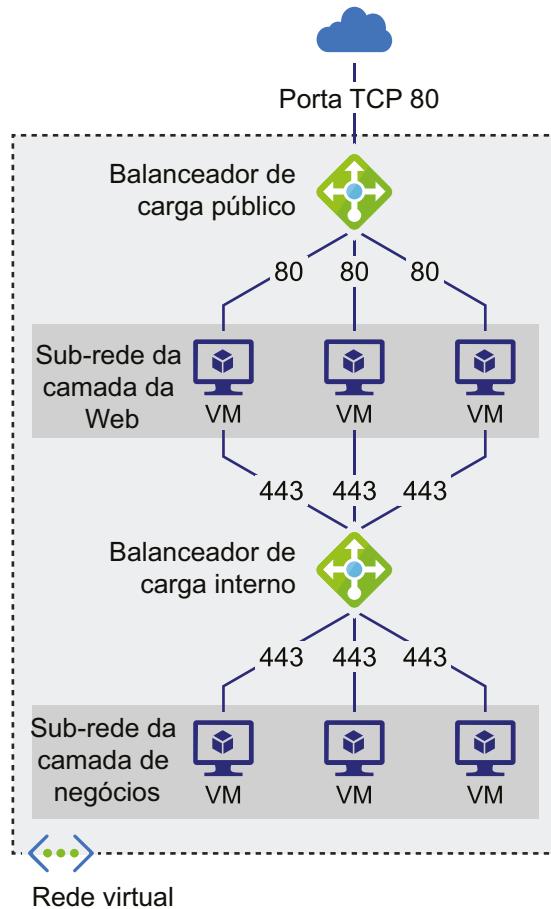


Figura 2.4: distribuição de tráfego usando平衡adores de carga do Azure

Na Figura 2.4, você pode ver que os usuários externos estão acessando as VMs por meio do balanceador de carga público e, em seguida, o tráfego da VM é distribuído em outro conjunto de VMs usando um balanceador de carga interno.

Fizemos uma comparação de como os平衡adores de carga do Azure diferem dos gateways de aplicação. Na próxima seção, discutiremos os gateways de aplicação em mais detalhes.

## O Gateway de Aplicação do Azure

Um balanceador de carga do Azure nos ajuda a habilitar soluções no nível da infraestrutura. No entanto, às vezes usar um balanceador de carga requer recursos e serviços avançados. Esses serviços avançados incluem terminação SSL, sessões fixas e segurança avançada, entre outros. Um Gateway de Aplicação do Azure fornece esses recursos adicionais. O Gateway de Aplicação do Azure é um balanceador de carga de nível 7 que funciona com a aplicação e a carga de sessão em uma pilha TCP OSI.

Os gateways de aplicação têm mais informações em comparação com os平衡adores de carga do Azure para tomar decisões sobre o roteamento de solicitações e o balanceamento de carga entre servidores. Os gateways de aplicação são gerenciados pelo Azure e altamente disponíveis.

Um gateway de aplicação situa-se entre os usuários e as VMs, como mostrado na Figura 2.5:

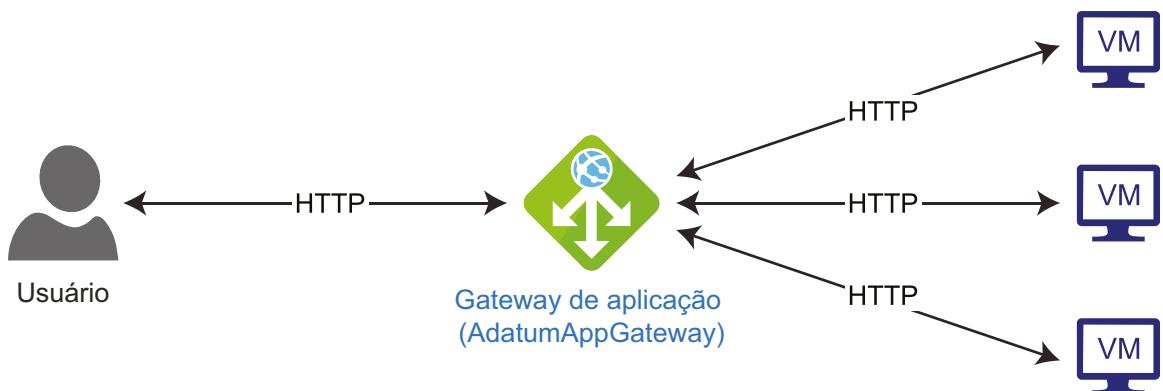


Figura 2.5: um gateway de aplicação do Azure

Os gateways de aplicação são um serviço gerenciado. Eles usam o **Application Request Routing (ARR)** para rotear solicitações para diferentes serviços e pontos de extremidade. A criação de um gateway de aplicação requer um endereço IP privado ou público. Então, o gateway de aplicação roteia o tráfego HTTP/HTTPS para pontos de extremidade configurados.

Um gateway de aplicação é semelhante a um balanceador de carga do Azure de uma perspectiva de configuração, com componentes e recursos adicionais. O gateway de aplicação pode ser configurado com um endereço IP de front-end, certificado, configuração de porta, pool de back-end, afinidade de sessão e informações de protocolo.

Outro serviço que discutimos em relação à alta disponibilidade para VMs foi o Gerenciador de Tráfego do Azure. Vamos tentar entender mais sobre esse serviço na próxima seção.

## Gerenciador de Tráfego do Azure

Depois de entender bem os平衡adores de carga do Azure e os gateways de aplicação, é hora de entrar nos detalhes do Gerenciador de Tráfego. Os balanceadores de carga e os gateways de aplicação do Azure são recursos muito necessários para a alta disponibilidade em um datacenter e uma região. No entanto, para obter a alta disponibilidade entre regiões e datacenters, há a necessidade de outro recurso e o Gerenciador de Tráfego nos ajuda nisso.

O Gerenciador de Tráfego nos ajuda a criar soluções altamente disponíveis que abrangem várias localidades geográficas, regiões e datacenters. O Gerenciador de Tráfego não é semelhante aos balanceadores de carga. Ele usa o **Serviço de Nomes de Domínio (DNS)** para redirecionar solicitações a um ponto de extremidade apropriado, determinado pela integridade e configuração. O Gerenciador de Tráfego não é um proxy nem um gateway e não vê o tráfego que passa entre o cliente e o serviço. Ele simplesmente redireciona as solicitações com base nos pontos de extremidade mais adequados.

O Gerenciador de Tráfego do Azure ajuda a controlar o tráfego distribuído pelos pontos de extremidade da aplicação. Um ponto de extremidade pode ser denominado como qualquer serviço voltada para a Internet hospedado dentro ou fora do Azure.

Pontos de extremidade são URLs públicos que podem ser acessados pela Internet. As aplicações são provisionadas em várias localidades geográficas e regiões do Azure. As aplicações implantadas em cada região têm um ponto de extremidade exclusivo chamado **DNS CNAME**. Esses pontos de extremidade são mapeados para o ponto de extremidade do Gerenciador de Tráfego. Quando uma instância do Gerenciador de Tráfego é provisionada, ela obtém um ponto de extremidade por padrão com uma extensão de URL **.trafficmanager.net**.

Quando uma solicitação chega ao URL do Gerenciador de Tráfego, ele encontra o ponto de extremidade mais apropriado na lista e redireciona a solicitação para ele. Resumindo, o Gerenciador de Tráfego do Azure age como um DNS global para identificar a região que atenderá à solicitação.

Mas, como o Gerenciador de Tráfego sabe quais pontos de extremidade usar e para qual deles as solicitações do cliente devem ser redirecionadas? Há dois aspectos que o Gerenciador de Tráfego considera para determinar o ponto de extremidade e a região mais apropriados.

Primeiro, o Gerenciador de Tráfego monitora ativamente a integridade de todos os pontos de extremidade. Ele pode monitorar a integridade de VMs, serviços de nuvem e de aplicativo. Se ele determinar que a integridade de uma aplicação implantada em uma região não é adequada para redirecionar o tráfego, redirecionará as solicitações para um ponto de extremidade íntegro.

Em segundo lugar, o Gerenciador de Tráfego pode ser configurado com informações de roteamento. Há seis métodos de roteamento de tráfego disponíveis no Gerenciador de Tráfego:

- **Prioridade:** deve ser usado quando todo o tráfego deve ir para um ponto de extremidade padrão, com backups disponíveis caso os pontos de extremidade principais não estejam disponíveis.
- **Ponderado:** deve ser usado para distribuir uniformemente o tráfego entre os pontos de extremidade ou de acordo com os pesos definidos.
- **Performance:** deve ser usado para pontos de extremidade em diferentes regiões, e os usuários devem ser redirecionados ao ponto de extremidade mais próximo com base em sua localização. Isso causa um impacto direto na latência da rede.
- **Geográfico:** isso deve ser usado para redirecionar os usuários para um ponto de extremidade (Azure, externo ou aninhado) com base no local geográfico mais próximo. Isso pode ajudar na conformidade relacionada à proteção de dados, localização e coleta de tráfego baseada na região.
- **Sub-rede:** é um novo método de roteamento e ajuda a fornecer aos clientes pontos de extremidade diferentes com base em seus endereços IP. Nesse método, um intervalo de endereços IP é atribuído a cada ponto de extremidade. Esses intervalos de endereços IP são mapeados para o endereço IP do cliente para determinar um ponto de extremidade de retorno apropriado. Com esse método de roteamento, é possível fornecer conteúdo diferente para pessoas diferentes com base no endereço IP de origem delas.
- **Multivalor:** é também um novo método adicionado ao Azure. Nele, vários pontos de extremidade são retornados para o cliente, e qualquer um deles pode ser usado. Isso garante que, se um ponto de extremidade não estiver íntegro, outros poderão ser usados. Isso ajuda a aumentar a disponibilidade geral da solução.

É importante observar que, depois que o Gerenciador de Tráfego determina um ponto de extremidade íntegro válido, os clientes se conectam diretamente à aplicação. Agora vamos entender os recursos do Azure no roteamento de solicitações de usuários globalmente.

Na próxima seção, discutiremos outro serviço, chamado Azure Front Door. Esse serviço é como o Gateway de Aplicação do Azure. No entanto, há uma pequena diferença que torna esse serviço distinto. Vamos avançar e saber mais sobre o Azure Front Door.

## Azure Front Door

O Azure Front Door é a oferta mais recente no Azure que ajuda a encaminhar solicitações para serviços em um nível global, em vez de um nível local ou de datacenter, como no caso do Gateway de Aplicação do Azure e dos平衡adores de carga. O Azure Front Door é como o Gateway de Aplicação, com a diferença no escopo. É um balanceador de carga de camada 7 que ajuda no roteamento de solicitações para o ponto de extremidade de serviço de melhor performance mais próximo implantado em várias regiões. Ele fornece recursos como terminação de TLS, afinidade de sessão, roteamento baseado em URL e hospedagem de vários sites, junto com um firewall de aplicação Web. Ele é semelhante ao Gerenciador de Tráfego, pois, por padrão, é resiliente a falhas de regiões inteiras e fornece recursos de roteamento. Ele também realiza investigações de integridade de ponto de extremidade periodicamente para garantir que as solicitações sejam encaminhadas somente para pontos de extremidade íntegros.

Ele fornece quatro métodos de roteamento:

- **Latência:** as solicitações serão roteadas para pontos de extremidade que terão a menor latência de ponta a ponta.
- **Prioridade:** as solicitações serão roteadas para um ponto de extremidade principal e para um ponto de extremidade secundário em caso de falha do principal.
- **Ponderado:** as solicitações serão roteadas com base nos pesos atribuídos aos pontos de extremidade.
- **Afinidade de sessão:** as solicitações em uma sessão terminam com o mesmo ponto de extremidade para fazer uso dos dados da sessão de solicitações anteriores. A solicitação original pode acabar com qualquer ponto de extremidade disponível.

As implantações que buscam resiliência no nível global devem incluir o Azure Front Door em sua arquitetura, junto com gateways de aplicação e平衡adores de carga. Na próxima seção, você verá algumas das considerações de arquitetura que você deve levar em conta ao criar soluções altamente disponíveis.

## Considerações sobre arquitetura para alta disponibilidade

O Azure fornece alta disponibilidade por diversos meios e em vários níveis. A alta disponibilidade pode estar no nível do datacenter, da região ou até mesmo de todo o Azure. Nesta seção, explicaremos algumas das arquiteturas de alta disponibilidade.

Alta disponibilidade em regiões do Azure

A arquitetura mostrada na Figura 2.6 mostra uma implantação de alta disponibilidade em uma única região do Azure. A alta disponibilidade é desenvolvida no nível de recurso individual. Nessa arquitetura, há várias VMs em cada camada conectada por meio de um gateway de aplicação ou balanceador de carga e elas fazem parte de um conjunto de disponibilidade. Cada nível é associado a um conjunto de disponibilidade. Essas VMs são colocadas em domínios de falha e atualização separados. Enquanto os servidores Web são conectados a gateways de aplicação, as camadas restantes, como aplicação e banco de dados, têm平衡adores de carga internos:

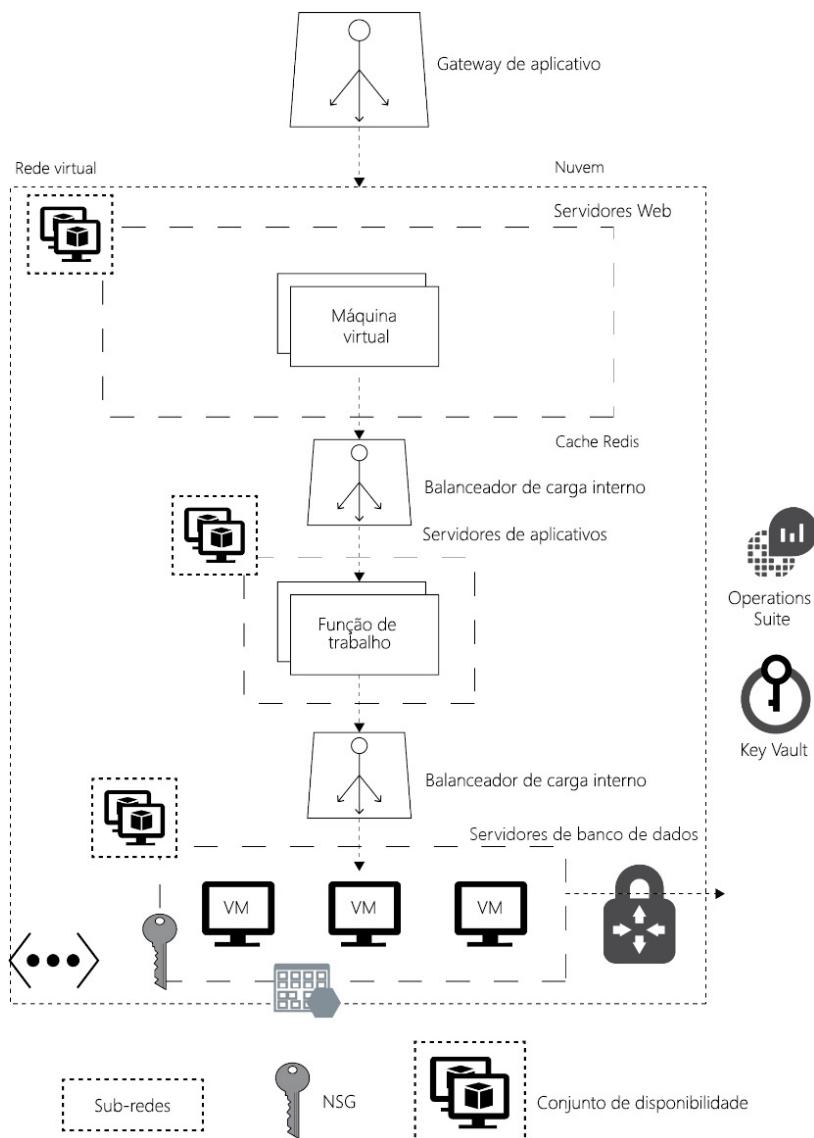


Figura 2.6: criando alta disponibilidade em uma região

Agora que você já sabe como criar soluções altamente disponíveis na mesma região, vamos discutir como uma arquitetura semelhante, mas espalhada por regiões do Azure, pode ser criada.

## Alta disponibilidade entre regiões do Azure

Essa arquitetura mostra implantações semelhantes em duas regiões diferentes do Azure. Conforme mostrado na Figura 2.7, ambas as regiões têm os mesmos recursos implantados. A alta disponibilidade é desenvolvida no nível de recurso individual nessas regiões. Há várias VMs em cada camada, conectadas por meio de平衡adores de carga, e elas fazem parte de um conjunto de disponibilidade. Essas VMs são colocadas em domínios de falha e atualização separados. Enquanto os servidores Web são conectados a平衡adores de carga externos, as camadas restantes, como aplicação e banco de dados, têm平衡adores de carga internos. Lembre-se de que os平衡adores de carga de aplicação podem ser usados para camadas de servidores Web e de aplicação, em vez de平衡adores de carga do Azure, se houver necessidade de serviços avançados, como afinidade de sessão, terminação SSL, segurança avançada usando um **firewall do aplicação Web (WAF)** e roteamento baseado em caminho. Os bancos de dados em ambas as regiões são conectados entre si usando emparelhamento de redes virtuais e gateways. Isso é útil na configuração de envio de log, do SQL Server AlwaysOn e de outras técnicas de sincronização de dados.

Os pontos de extremidade dos平衡adores de carga de ambas as regiões são usados para configurar pontos de extremidade do Gerenciador de Tráfego; o tráfego é roteado com base no método de balanceamento de carga prioritário. O Gerenciador de Tráfego ajuda a rotear todas as solicitações para a região do leste dos EUA e, após o failover, para a Europa Ocidental em caso de indisponibilidade da primeira região:

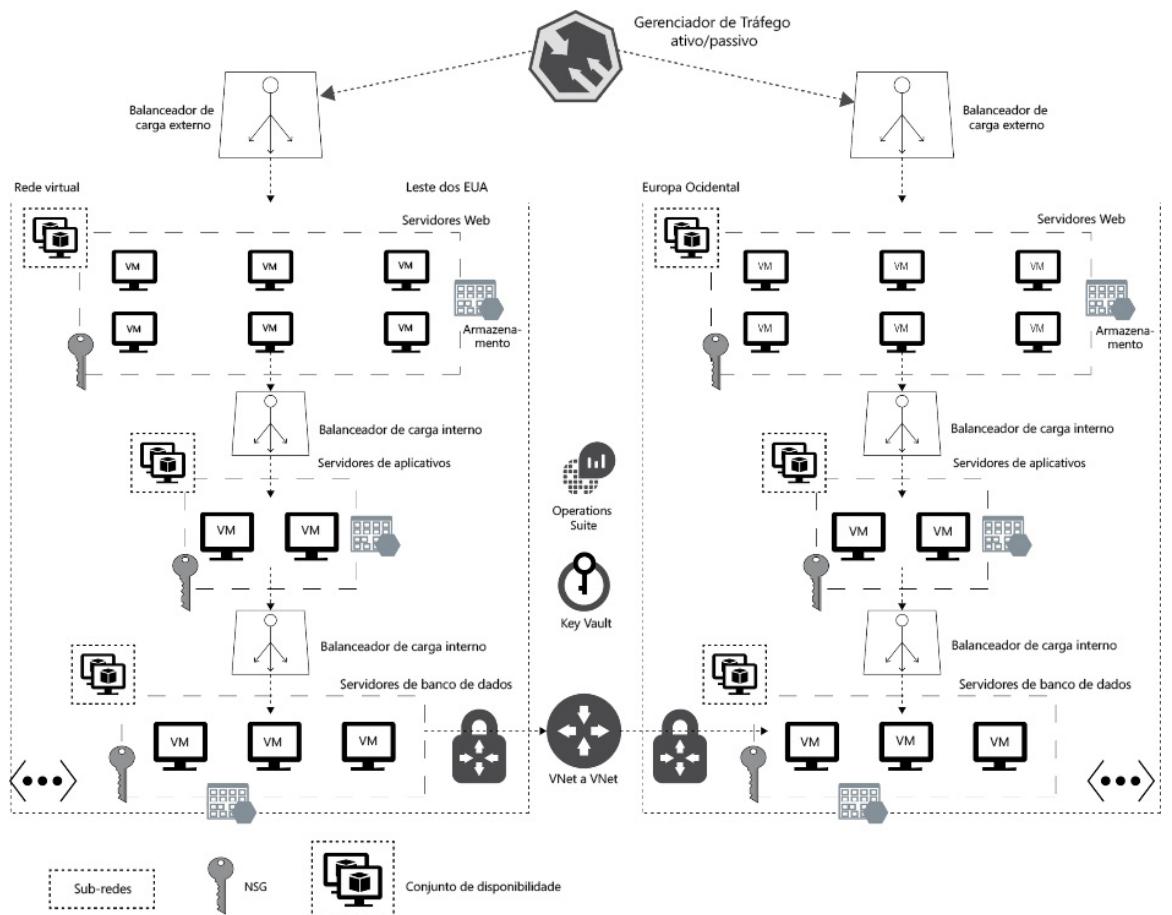


Figura 2.7: criando alta disponibilidade em regiões do Azure

Na próxima seção, exploraremos a escalabilidade, que é outra vantagem da nuvem.

## Escalabilidade

A execução de aplicações e sistemas que estão disponíveis para os usuários para consumo é importante para arquitetos de qualquer aplicação crítica para os negócios. No entanto, há outro recurso de aplicação igualmente importante que é uma das principais prioridades para os arquitetos: a escalabilidade da aplicação.

Imagine uma situação em que uma aplicação é implantada e obtém excelente performance e disponibilidade com poucos usuários, mas tanto a disponibilidade quanto a performance diminuem à medida que o número de usuários começa a aumentar. Há momentos em que uma aplicação sob carga normal tem boa performance, mas essa performance diminui com o aumento do número de usuários. Isso pode acontecer quando há um aumento repentino no número de usuários e o ambiente não foi criado para um número tão grande de usuários.

Para acomodar esses picos no número de usuários, você pode provisionar o hardware e a largura de banda para gerenciar esses picos. O desafio com isso é que a capacidade adicional não é usada na maior parte do ano e, portanto, não fornece retorno sobre o investimento. Ela é provisionada para uso apenas durante a temporada de férias ou vendas. Espero que até o momento você esteja se familiarizando com os problemas que os arquitetos estão tentando resolver. Todos esses problemas estão relacionados ao dimensionamento da capacidade e à escalabilidade de uma aplicação. O foco deste capítulo é entender a escalabilidade como uma preocupação de arquitetura e conferir os serviços fornecidos pelo Azure para implementar a escalabilidade.

Planejamento e dimensionamento de capacidade são algumas das principais prioridades dos arquitetos e seus serviços e aplicações. Os arquitetos devem encontrar um equilíbrio entre a compra e o provisionamento de recursos a mais ou a menos. Se tiver menos recursos, você não conseguirá atender a todos os usuários, e eles recorrerão aos concorrentes. Por outro lado, ter mais recursos pode prejudicar seu orçamento e retorno sobre o investimento, pois a maioria dos recursos permanece não utilizada na maior parte do tempo. Além disso, o problema aumenta com os níveis variados de demanda em diferentes momentos. É quase impossível prever o número de usuários de uma aplicação em um dia, muito menos um ano. No entanto, é possível encontrar um número aproximado usando informações anteriores e monitoramento contínuo.

A escalabilidade refere-se à capacidade de gerenciar um número crescente de usuários e de fornecer a eles o mesmo nível de performance quando há menos usuários fazendo uso de recursos de implantação de aplicações, processos e tecnologia. A escalabilidade pode significar o atendimento de mais solicitações sem reduzir a performance ou a capacidade de lidar com um trabalho maior e mais demorado sem qualquer perda de performance nos dois casos.

Os exercícios de planejamento e dimensionamento de capacidade devem ser realizados pelos arquitetos no início de um projeto e durante a fase de planejamento para oferecer escalabilidade para aplicações.

Algumas aplicações têm padrões de demanda estáveis, embora seja difícil prever outros. Requisitos de escalabilidade são conhecidos para aplicações de demanda estável, embora o discernimento deles possa ser um processo mais envolvido para aplicações de demanda variável. O dimensionamento automático, um conceito que veremos na próxima seção, deve ser usado para as aplicações cujas demandas não podem ser previstas.

As pessoas frequentemente tendem a confundir escalabilidade com performance. Na próxima seção, você verá uma comparação rápida desses dois termos.

## **Escalabilidade versus performance**

É muito fácil ficar confuso entre as preocupações de arquitetura de escalabilidade e performance, pois a escalabilidade tem a ver com garantir que, independentemente do número de usuários que consomem a aplicação, todos recebam o mesmo nível predeterminado de performance.

A performance está relacionada à garantia de que uma aplicação atenda a taxas de transferência e tempos de resposta predefinidos. A escalabilidade refere-se a ter provisões para mais recursos quando necessário com a finalidade de acomodar mais usuários sem sacrificar a performance.

É melhor entender isso usando uma analogia: a velocidade de um trem está diretamente relacionada à performance de uma rede ferroviária. No entanto, fazer com que mais trens funcionem em paralelo a velocidades iguais ou maiores representa a escalabilidade da rede ferroviária.

Agora que você sabe a diferença entre escalabilidade e performance, vamos discutir como o Azure fornece escalabilidade.

## Escalabilidade do Azure

Nesta seção, veremos as funcionalidades e os recursos fornecidos pelo Azure para tornar as aplicações altamente disponíveis. Antes de entrarmos em detalhes de arquitetura e configuração, é importante compreender os conceitos de alta disponibilidade do Azure. Em outras palavras, dimensionamento.

O dimensionamento refere-se ao aumento ou à diminuição da quantidade de recursos usados para atender a solicitações de usuários. A escala pode ser manual ou automática. A escala manual exige que um administrador inicie manualmente o processo de escalonamento, enquanto a escala automática refere-se a um aumento ou diminuição de forma automática de recursos com base nos eventos disponíveis no ambiente e no ecossistema, como disponibilidade da memória e da CPU. Os recursos podem ter expansão ou redução vertical ou horizontal, o que será explicado a seguir nesta seção.

Além do lançamento de atualizações, as construções fundamentais proporcionadas pelo Azure para obter alta disponibilidade são as seguintes:

- Expansão e redução verticais
- Expansão e redução horizontais
- Dimensionamento automático

### Expansão vertical

A escala vertical de uma VM ou de um serviço envolve a adição de mais recursos para servidores existentes, como CPU, memória e discos. O objetivo é aumentar a capacidade do hardware e dos recursos físicos existentes.

### Redução vertical

A redução vertical de uma VM ou de um serviço envolve a remoção de recursos existentes de servidores existentes, como CPU, memória e discos. O objetivo é reduzir a capacidade do hardware e dos recursos físicos e virtuais existentes.

## Expansão horizontal

A expansão envolve a adição de mais hardware, como servidores e capacidade adicionais. Isso normalmente envolve adicionar novos servidores, atribuir endereços IP a eles, implantar aplicações neles e torná-los parte dos平衡adores de carga existentes, de modo que o tráfego possa ser encaminhado para eles. A escala horizontal também pode ser manual ou automática. No entanto, para melhores resultados, a automação deve ser usada:



Figura 2.8: expansão horizontal

## Redução horizontal

A redução horizontal refere-se ao processo de remoção do hardware existente em termos de capacidade e servidores existentes. Isso normalmente envolve remover servidores existentes, desalocar endereços IP e removê-los da configuração existente de平衡adores de carga de modo que o tráfego não possa ser roteado para eles. Assim como a escala horizontal, a redução horizontal pode ser automática ou manual.

## Dimensionamento automático

A escala automática refere-se ao processo de expandir/reduzir vertical ou horizontalmente de forma dinâmica com base na demanda da aplicação e ocorre por meio da automação. O dimensionamento automático é útil porque garante que a implantação sempre consista em um número ideal de instâncias de servidor. A escala automática ajuda a criar aplicações tolerantes a falhas. Isso não só ajuda na escalabilidade, mas também torna as aplicações altamente disponíveis. Por fim, ela proporciona o melhor gerenciamento de custos. O dimensionamento automático possibilita a configuração ideal para instâncias de servidor com base na demanda. Ele ajuda a não provisionar demais os servidores para que eles acabem subutilizados e remove servidores que não são mais necessários após a escala horizontal.

Até agora, discutimos a escalabilidade no Azure. O Azure oferece opções de escalabilidade para a maioria dos seus serviços. Vamos explorar a escalabilidade para PaaS no Azure na próxima seção.

## Escalabilidade da PaaS

O Azure fornece serviços de aplicativo para hospedar aplicações gerenciadas. O serviço de aplicativo é uma oferta de PaaS do Azure. Ele fornece serviços para a Web e plataformas móveis. Por trás das plataformas Web e móvel está uma infraestrutura gerenciada pelo Azure em nome de seus usuários. Os usuários não veem nem gerenciam nenhuma infraestrutura. No entanto, eles podem estender a plataforma e implantar suas aplicações nela. Com isso, os arquitetos e desenvolvedores podem se concentrar em seus problemas de negócios em vez de se preocupar com a plataforma básica e o provisionamento, a configuração e a solução de problemas da infraestrutura. Os desenvolvedores têm a flexibilidade de escolher qualquer linguagem, sistema operacional e estrutura para desenvolver suas aplicações. O serviço de aplicativo fornece vários planos e, com base nos planos escolhidos, vários graus de escalabilidade são disponibilizados. O serviço de aplicativo fornece os cinco planos a seguir:

- **Gratuito:** usa uma infraestrutura compartilhada. Isso significa que várias aplicações serão implantadas na mesma infraestrutura do mesmo ou de vários locatários. Ele fornece 1 GB de armazenamento gratuito. No entanto, não há capacidade de escala nesse plano.
- **Compartilhado:** também usa uma infraestrutura compartilhada e fornece 1 GB de armazenamento gratuito. Além disso, domínios personalizados também são fornecidos como um recurso extra. No entanto, não há capacidade de escala nesse plano.
- **Básico:** tem três **unidades de manutenção de estoque (SKUs)**: B1, B2 e B3. Elas têm um número maior de unidades de recursos disponíveis em termos de CPU e memória. Resumindo, essas unidades fornecem uma configuração melhor das VMs com apoio para esses serviços. Além disso, fornece armazenamento, domínios personalizados e suporte para SSL. O plano básico oferece recursos básicos para escala manual. Não há dimensionamento automático disponível nesse plano. No máximo três instâncias podem ser usadas para a escala horizontal da aplicação.
- **Padrão:** também tem três SKUs diferentes: S1, S2 e S3. Elas têm um número maior de unidades de recursos disponíveis em termos de CPU e memória. Resumindo, essas unidades fornecem uma configuração melhor das VMs com apoio para esses serviços. Além disso, elas fornecem armazenamento, domínios personalizados e suporte para SSL semelhantes ao do plano básico. Esse plano também fornece uma instância do Gerenciador de Tráfego, slots de preparo e um backup diário como recurso adicional acima do plano básico. O plano padrão oferece recursos para dimensionamento automático. No máximo 10 instâncias podem ser usadas para a escala horizontal da aplicação.

- Premium:** também tem três SKUs diferentes: P1, P2 e P3. Elas têm um número maior de unidades de recursos disponíveis em termos de CPU e memória. Resumindo, essas unidades fornecem uma configuração melhor das VMs com apoio para esses serviços. Além disso, elas fornecem armazenamento, domínios personalizados e suporte para SSL semelhantes ao plano básico. Esse plano também fornece uma instância do Gerenciador de Tráfego, slots de preparo e 50 backups diários como recurso adicional acima do plano básico. O plano padrão oferece recursos para dimensionamento automático. No máximo 20 instâncias podem ser usadas para a escala horizontal da aplicação.

Exploramos os níveis de escalabilidade disponíveis para serviços PaaS. Agora, vamos ver como o dimensionamento pode ser feito no caso de um plano de serviço de aplicativo.

### PaaS – Expansão e redução verticais

A expansão e a redução verticais de serviços hospedados pelo serviço de aplicativo são muito simples. O menu dos serviços de aplicativo do Azure abre um novo painel com todos os planos e suas SKUs listadas. A escolha de um plano e um SKU expandirá ou reduzirá o serviço, conforme mostrado na Figura 2.9:

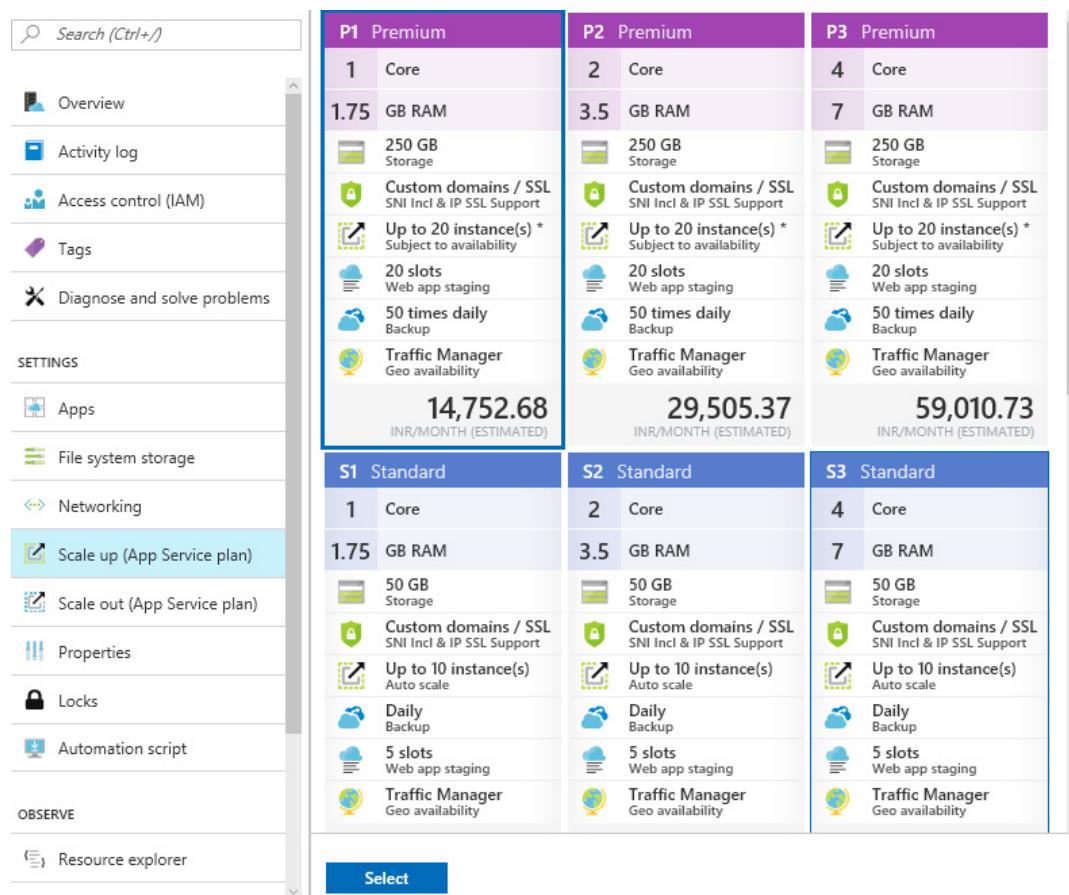


Figura 2.9: planos diferentes com suas SKUs

## PaaS – Expansão e redução horizontais

A expansão e a redução horizontais de serviços hospedados no serviço de aplicativo também são muito simples. O item de menu de escala horizontal dos serviços de aplicativo do Azure abre um novo painel com opções de configuração de escala.

Por padrão, a escala automática é desativada para os planos Premium e padrão. Ela pode ser ativada usando o item de menu **Scale Out** e clicando no botão **Enable autoscale**, conforme mostrado na Figura 2.10:

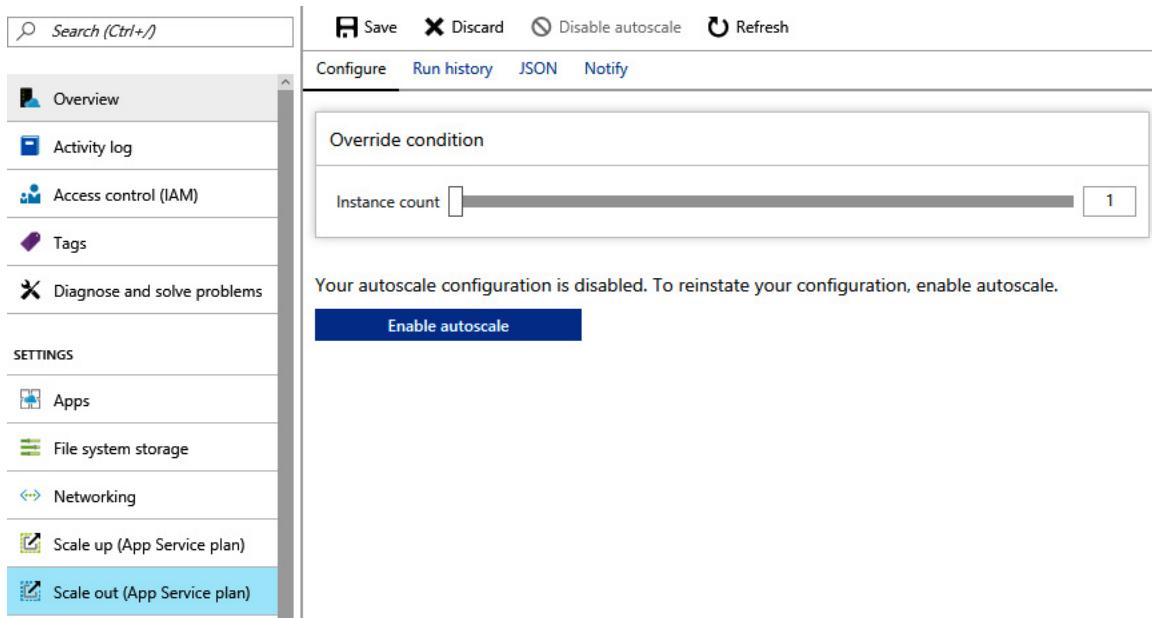


Figura 2.10: habilitando a opção dimensionamento

O dimensionamento manual não precisa de configuração, mas o dimensionamento automático ajuda na configuração por meio das seguintes propriedades de escala:

- **Modo de dimensionamento:** baseia-se em uma métrica de performance, como CPU ou uso de memória. Os usuários também podem especificar o número de instâncias para dimensionamento.
- **Quando escalar:** várias regras podem ser adicionadas que determinam quando expandir ou reduzir horizontalmente. Cada regra pode determinar critérios como consumo de CPU ou memória, se é preciso aumentar ou diminuir o número de instâncias, quantas instâncias aumentar ou diminuir por vez. É preciso configurar pelo menos uma regra para a escala horizontal e uma regra para a redução horizontal. As definições de limites ajudam a definir os limites superior e inferior que devem acionar o dimensionamento automático – aumentando ou diminuindo o número de instâncias.
- **Como escalar:** especifica quantas instâncias criar ou remover em cada expansão ou redução horizontal:

The screenshot shows the Azure portal interface for managing an App Service plan. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main area is titled 'Configure' and shows the 'Scale out (App Service plan)' settings. It includes fields for 'Autoscale setting name' (set to 'destRG'), 'Resource group', and 'Default' scale condition. The 'Scale mode' is set to 'Scale based on a metric'. A detailed rule is defined: 'Scale out and scale in your instances based on metric. For example, add a rule that increases instance count is above 70%'. Below this, there are sections for 'Instance limits' (with Minimum, Maximum, and Default all set to 1) and 'Schedule' (with a note: 'This scale condition is executed when none of the other scale condition(s) match'). There are also buttons for '+ Add a rule' and '+ Add a scale condition'. On the right side, there's a 'Criteria' section with various configuration options: Time aggregation (Average), Metric name (CPU Percentage), Time grain statistic (1 minute time grain), Operator (Greater than), Threshold (70), Duration (in minutes) (10), and an 'Action' section with Operation (Increase count by 1), Instance count (1), and Cool down (minutes) (5). A large blue 'Add' button is located at the bottom right.

Figura 2.11: definindo os limites da instância

É um recurso muito bom para ser habilitado em qualquer implantação. No entanto, você deve habilitar as escalas vertical e horizontal em conjunto para garantir que seu ambiente volte à capacidade normal após a expansão.

Como já vimos a escalabilidade em PaaS, vamos discutir a escalabilidade na IaaS a seguir.

## Escalabilidade da IaaS

Há usuários que querem ter controle total sobre sua infraestrutura de base, a plataforma e a aplicação. Eles preferem consumir soluções de IaaS, em vez de soluções de PaaS. Quando esses clientes criam VMs, eles também são responsáveis por dimensionar e escalar a capacidade. Não há nenhuma configuração pronta para o uso para dimensionamento manual ou automático de VMs. Esses clientes precisarão escrever os próprios scripts de automação, gatilhos e regras para obter o dimensionamento automático. Com as VMs vem a responsabilidade de mantê-las. A aplicação de patches, atualização e upgrade de VMs é da responsabilidade dos proprietários. Os arquitetos devem pensar tanto na manutenção planejada quanto na não planejada. Como essas VMs devem receber patches, a ordem, o agrupamento e outros fatores devem ser considerados para garantir que nem a escalabilidade nem a disponibilidade de uma aplicação sejam comprometidas. Para ajudar a aliviar esses problemas, o Azure fornece **conjuntos de escalas de VM (VMSS)** como uma solução, o que discutiremos a seguir.

## Conjuntos de escalas de VM

Os VMSSs são recursos de computação do Azure que você pode usar para implantar e gerenciar um conjunto de VMs idênticas. Com todas as VMs configuradas da mesma maneira, os conjuntos de escalas são projetados para oferecer suporte à escala automática real, e nenhum pré-provisionamento de VMs é necessário. Isso ajuda no provisionamento de várias VMs idênticas conectadas umas às outras por meio de uma rede virtual e sub-rede.

Um VMSS consiste em várias VMs, mas é gerenciado no nível dos VMSS. Todas as VMs fazem parte dessa unidade e todas as alterações feitas são aplicadas à unidade que, por sua vez, as aplica a VMs que usam um algoritmo predeterminado:

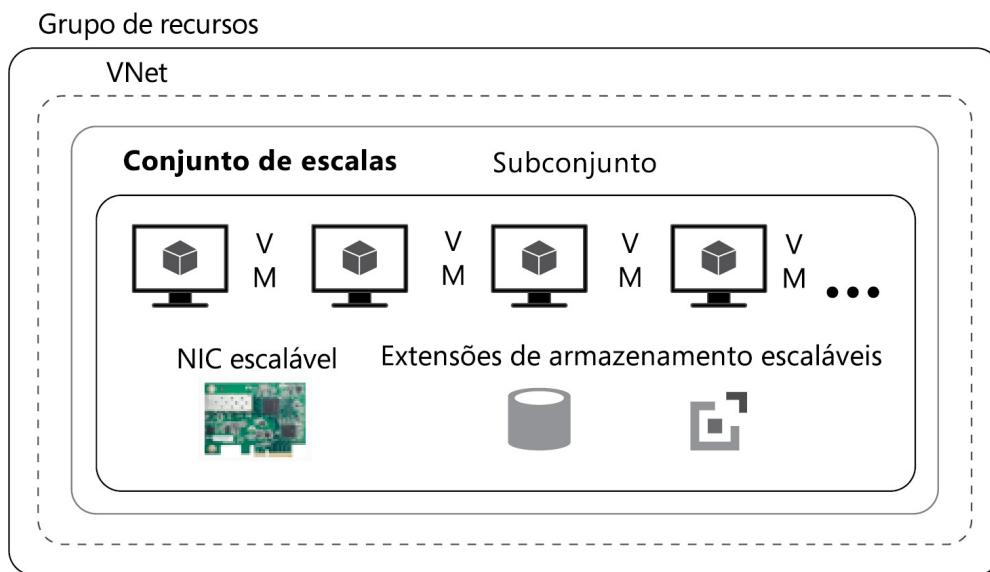


Figura 2.12: conjunto de escala de VM

Isso permite que essas VMs tenham a carga balanceada usando um平衡ador de carga do Azure ou um gateway de aplicação. As VMs podem ser Windows ou Linux. Elas podem executar scripts automatizados usando uma extensão do PowerShell e podem ser gerenciadas centralmente usando uma configuração de estado. Elas podem ser monitoradas como uma unidade ou individualmente usando a análise de log.

Os VMSSs podem ser provisionados do portal do Azure, da CLI do Azure, dos modelos do Azure Resource Manager, das APIs REST e de cmdlets do PowerShell. É possível invocar as APIs REST e a CLI do Azure de qualquer plataforma, ambiente e sistema operacional ou em qualquer linguagem.

Muitos serviços do Azure já usam VMSSs como arquitetura subjacente. Entre eles estão o Lote do Azure, Azure Service Fabric e Serviço de Contêiner do Azure. O Serviço de Contêiner do Azure, por sua vez, provisionam o Kubernetes e o DC/OS nesses VMSSs.

## Arquitetura de VMSS

Os VMSSs permitem a criação de até 1.000 VMs em um conjunto de dimensionamento ao usar uma imagem de plataforma e 100 VMs, se estiver usando uma imagem personalizada. Se o número de VMs for inferior a 100 em um conjunto de dimensionamento, elas serão colocadas em um único conjunto de disponibilidade. No entanto, se o número for superior a 100, vários conjuntos de disponibilidade serão criados (conhecidos como grupos de posicionamento), e as VMs serão distribuídas entre esses conjuntos de disponibilidade. Sabemos desde o Capítulo 1, *Introdução ao Azure*, que as VMs em um conjunto de disponibilidade são colocadas em domínios de falha e atualização separados. Os conjuntos de disponibilidade relacionados aos VMSSs têm cinco domínios de falha e atualização por padrão. Os VMSSs fornecem um modelo que contém informações de metadados para todo o conjunto. Alterar esse modelo e aplicar alterações afeta todas as instâncias de VM. Essas informações incluem os números de instâncias de VM máximas e mínimas, a SKU e a versão do sistema operacional, o número atual de VMs, domínios de falha e atualização e muito mais. Isso é demonstrado na Figura 2.13:

Conjunto de disponibilidade

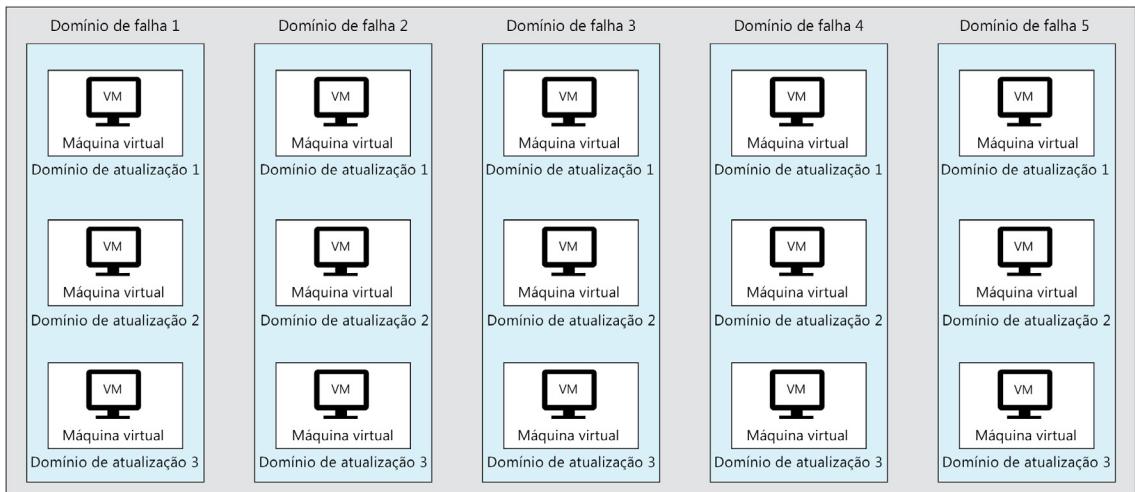


Figura 2.13: VMs em um conjunto de disponibilidade

## Escala VMSS

A escala refere-se a um aumento ou diminuição de recursos de computação e armazenamento. Um VMSS é um recurso avançado que torna a escala fácil e eficiente. Ele oferece a escala automática, que ajuda a expandir ou reduzir verticalmente com base em eventos e dados externos, como uso de CPU e memória. Alguns dos recursos de escala do VMSS são indicados aqui.

### Escala horizontal x vertical

A escala pode ser horizontal, vertical ou uma combinação de ambas. Escala horizontal é outro nome para a expansão e redução horizontais, enquanto a escala vertical refere-se à expansão e redução verticais.

## Capacidade

Os VMSSs têm uma propriedade de **capacidade** que determina o número de VMs em um conjunto de escalas. Um VMSS pode ser implantado com o zero como o valor dessa propriedade. Isso não criará uma única VM. No entanto, se você provisionar um VMSS fornecendo um número para a propriedade de **capacidade**, esse número de VMs será criado.

## Dimensionamento automático

O dimensionamento automático de VMs no VMSS refere-se à adição ou remoção de instâncias de VM com base no ambiente configurado para atender às demandas de performance e escalabilidade de uma aplicação. Geralmente, na ausência de um VMSS, isso é conseguido por meio de scripts de automação e runbooks.

Os VMSSs ajudam nesse processo de automação, com o suporte da configuração. Em vez de escrever scripts, um VMSS pode ser configurado para expansão e redução verticais automatizadas.

O dimensionamento automático usa vários componentes integrados para atingir seu objetivo final. O dimensionamento automático envolve o monitoramento contínuo de VMs e a coleta de dados de telemetria sobre elas. Esses dados são armazenados, combinados e, então, avaliados com base em um conjunto de regras para determinar se o dimensionamento automático deve ser acionado. O gatilho poderia ser para uma expansão/redução horizontal ou vertical.

O mecanismo de dimensionamento automático usa logs de diagnóstico para coletar dados de telemetria de VMs. Esses logs são armazenados em contas de armazenamento como métricas de diagnóstico. O mecanismo de dimensionamento automático também usa o serviço de monitoramento de insights de aplicação, que lê essas métricas e as combina e armazena em uma conta de armazenamento.

Os trabalhos em segundo plano com dimensionamento automático são executados continuamente para ler os dados de armazenamento de insights de aplicação, avaliá-los com base em todas as regras configuradas para dimensionamento automático e executar o processo de dimensionamento automático, se alguma das regras ou uma combinação de regras for atendida. As regras podem levar em consideração as métricas das VMs convidadas e do servidor host.

As regras definidas usando as descrições de propriedade estão disponíveis em <https://docs.microsoft.com/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview>.

A arquitetura de dimensionamento automático do VMSS é mostrada na Figura 2.14:

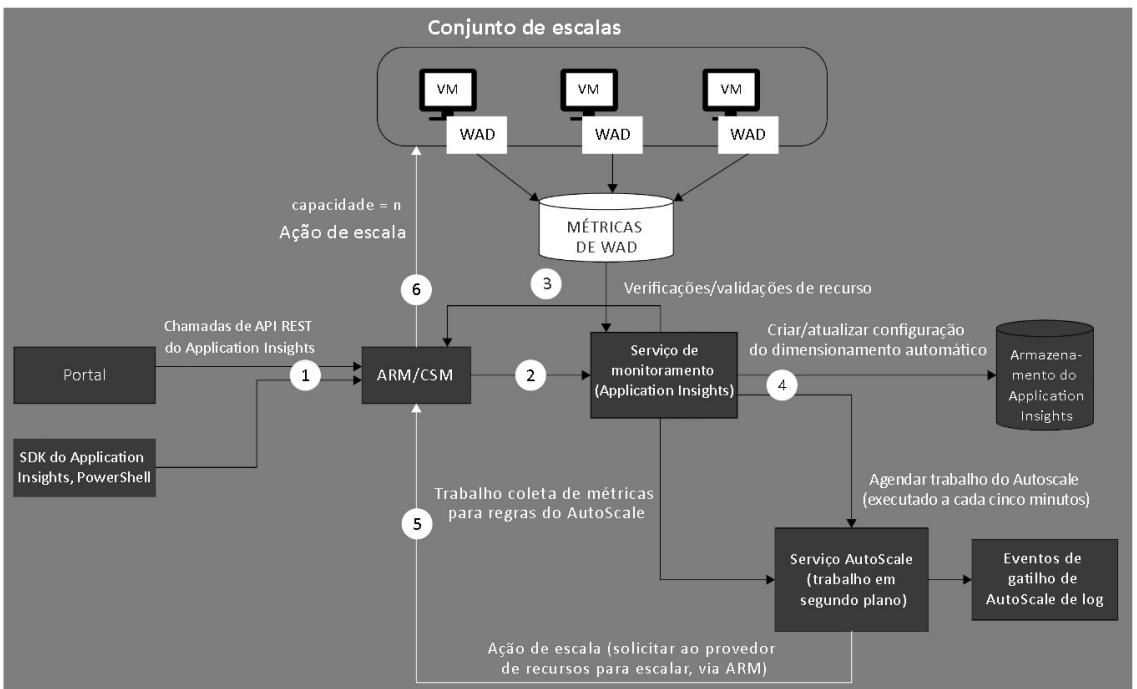


Figura 2.14: arquitetura de dimensionamento automático do VMSS

O dimensionamento automático pode ser configurado para cenários mais complexos do que as métricas gerais disponíveis nos ambientes. Por exemplo, o dimensionamento pode ser baseado em qualquer um dos eventos a seguir:

- Um dia específico
- Uma agenda recorrente, como fins de semana
- Dias úteis versus fins de semana
- Feriados e eventos pontuais
- Várias métricas de recursos

Elas podem ser configuradas usando a propriedade de **agendamento** de recursos de insights de aplicação, o que ajuda a registrar regras.

Os arquitetos devem assegurar que pelo menos duas ações, expansão e redução horizontais, sejam configuradas em conjunto. A expansão e redução horizontais de uma configuração não ajudará a obter os benefícios de escala fornecidos pelos VMSSs.

Para resumir, cobrimos as opções de escalabilidade no Azure e os recursos detalhados de escala no caso de IaaS e PaaS para atender aos seus requisitos de negócios. Se você se lembrar do modelo de responsabilidade compartilhada, lembrará que as atualizações e a manutenção da plataforma devem ser feitas pelo provedor de nuvem. Nesse caso, a Microsoft cuida de atualizações e manutenção relacionadas à plataforma. Vamos ver como isso é feito na próxima seção.

## Atualizações e manutenção

Depois que um VMSS e as aplicações são implantados, eles precisam ser mantidosativamente. A manutenção planejada deve ser realizada periodicamente para garantir que tanto o ambiente quanto a aplicação sejam atualizados com os recursos mais recentes do ponto de vista de segurança e resiliência.

Atualizações podem ser associadas a aplicações, à instância de VM convidada ou à própria imagem. As atualizações podem ser bastante complexas, pois devem acontecer sem afetar a disponibilidade, escalabilidade e performance de ambientes e aplicações. Para garantir que as atualizações possam ocorrer em uma instância de cada vez usando métodos de atualização sem interrupção, é importante que o VMSS ofereça suporte e recursos para esses cenários avançados.

A equipe do Azure fornece um utilitário para gerenciar atualizações dos VMSSs. É um utilitário baseado em Python cujo download pode ser feito em <https://github.com/gbownerman/vmssdashboard>. Ele faz chamadas da API REST para o Azure para gerenciar conjuntos de escalas. Esse utilitário pode ser usado para iniciar, parar, atualizar e recriar VMs em um FD ou grupo de VMs, conforme mostrado na *Figura 2.15*:

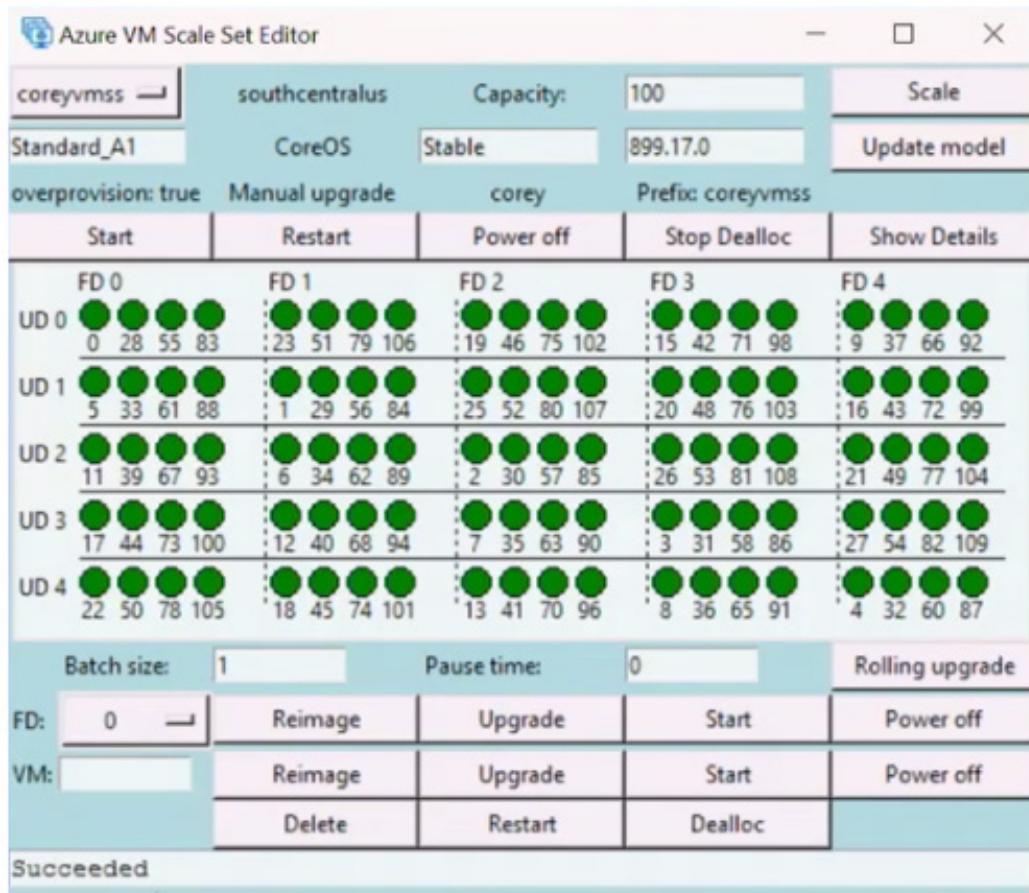


Figura 2.15: utilitário para o gerenciamento de atualizações do VMSS

Já que você tem uma compreensão básica da atualização e da manutenção, vamos ver como as atualizações de aplicações são feitas nos VMSSs.

## Atualizações de aplicações

As atualizações de aplicações nos VMSSs não devem ser executadas manualmente. Elas devem ser executadas como parte do gerenciamento e pipelines de versões que usam a automação. Além disso, a atualização deve ocorrer em uma instância de aplicação por vez e não deve afetar a disponibilidade geral e a escalabilidade de uma aplicação. Ferramentas de gerenciamento de configuração, como a **configuração de estado desejada (DSC)**, devem ser implantadas para gerenciar as atualizações de aplicações. O servidor de pull do DSC pode ser configurado com a versão mais recente da configuração de aplicações, e ele deve ser aplicado sem interrupção a cada instância.

Na próxima seção, vamos nos concentrar em como as atualizações são feitas no sistema operacional convidado.

## Atualizações de convidados

As atualizações de VMs são de responsabilidade do administrador. O Azure não é responsável por corrigir VMs convidadas. As atualizações de convidados estão no modo de visualização, e os usuários devem controlar a aplicação de patches manualmente ou usar métodos de automação personalizada, como runbooks e scripts. No entanto, as atualizações de patch sem interrupção estão no modo de visualização e podem ser configuradas no modelo do Azure Resource Manager usando uma política de atualização, conforme mostrado aqui:

```
"upgradePolicy": {  
    "mode": "Rolling",  
    "automaticOSUpgrade": "true" or "false",  
    "rollingUpgradePolicy": {  
        "batchInstancePercent": 20,  
        "maxUnhealthyUpgradedInstanceCount": 0,  
        "pauseTimeBetweenBatches": "PT0S"  
    }  
}
```

Agora que sabemos como as atualizações de convidados são gerenciadas no Azure, vamos ver como as atualizações de imagem são realizadas.

## Atualizações de imagens

Um VMSS pode atualizar a versão do sistema operacional sem qualquer tempo de inatividade. As atualizações do sistema operacional envolvem a alteração da versão ou da SKU do sistema operacional ou a alteração do URI de uma imagem personalizada. Atualizar sem tempo de inatividade significa atualizar as VMs uma de cada vez ou em grupos (como um FD por vez) em vez de todas de uma vez. Ao fazer isso, as VMs que não estiverem sendo atualizadas poderão continuar funcionando.

Até agora, discutimos atualizações e manutenção. Agora vamos ver quais são as práticas recomendadas de dimensionamento para VMSSs.

## Práticas recomendadas de dimensionamento para VMSSs

Nesta seção, explicaremos algumas das práticas recomendadas que as aplicações devem implementar para aproveitar as vantagens do recurso de dimensionamento fornecido pelos VMSSs.

### A preferência pela expansão horizontal

Expandir horizontalmente é uma solução melhor do que a expansão vertical. Expandir ou reduzir verticalmente significa redimensionar instâncias de VM. Quando uma VM é redimensionada, ela geralmente precisa ser reiniciada, o que tem suas próprias desvantagens. Primeiro, há um tempo de inatividade para a máquina. Segundo, se houver usuários ativos conectados à aplicação nessa instância, poderá haver falta de disponibilidade da aplicação ou até mesmo perda de transações. Expandir horizontalmente não afeta as VMs existentes, mas provisiona máquinas mais recentes e as adiciona ao grupo.

### Instâncias novas versus inativas

O dimensionamento de novas instâncias pode ter duas abordagens amplas: criar a nova instância do zero, o que requer a instalação de aplicações, configurá-las e testá-las ou iniciar as instâncias inativas quando elas forem necessárias devido à pressão de escalabilidade em outros servidores.

### Configuração apropriada do número máximo e mínimo de instâncias

Definir dois como o valor mínimo e máximo de contagem de instâncias com a atual contagem de instâncias sendo dois, nenhuma ação de escala pode ocorrer. Deve haver uma diferença adequada entre as contagens máxima e mínima de instâncias, que são inclusivas. O dimensionamento automático sempre escala entre esses limites.

### Simultaneidade

As aplicações são criadas para escalabilidade com foco na simultaneidade. As aplicações devem usar padrões assíncronos para garantir que as solicitações do cliente não esperem indefinidamente pela aquisição de recursos, caso os recursos estejam ocupados atendendo a outras solicitações. A implementação de padrões assíncronos no código garante que os threads não aguardem pelos recursos e que os sistemas esgotem todos os threads disponíveis. As aplicações devem implementar o conceito de tempo limite se houver falhas intermitentes esperadas.

### Criação de aplicações sem estado

Aplicações e serviços devem ser criados para não terem estado. A escalabilidade pode se tornar um desafio de resolver com serviços com estado e é muito fácil escalar serviços sem estado. Com o estado vem o requisito de componentes adicionais e implementações, como replicação, repositório centralizado ou descentralizado, manutenção e sessões fixas. Todos eles são obstáculos no caminho para a escalabilidade. Imagine um serviço de manutenção de estado ativo em um servidor local. Não importa o número de solicitações na aplicação geral ou no servidor individual, as solicitações subsequentes devem ser atendidas pelo mesmo servidor. As solicitações subsequentes não podem ser processadas por outros servidores. Isso torna a implementação de escalabilidade um desafio.

## Armazenamento em cache e a Rede de Distribuição de Conteúdo (CDN)

Aplicações e serviços devem tirar proveito do armazenamento em cache. O armazenamento em cache ajuda a eliminar várias chamadas subsequentes para qualquer banco de dados ou sistema de arquivos. Isso ajuda a tornar os recursos disponíveis e gratuitos para mais solicitações. A CDN é outro mecanismo usado para armazenar em cache arquivos estáticos, como imagens e bibliotecas de JavaScript. Eles estão disponíveis em servidores no mundo todo. Eles também disponibilizam recursos gratuitamente para solicitações de clientes adicionais – isso torna as aplicações altamente escaláveis.

## Design N+1

O **design N + 1** refere-se à criação de redundância na implantação geral para cada componente. Significa planejar alguma redundância mesmo quando ela não é necessária. Podem ser VMs adicionais, armazenamento e interfaces de rede.

Considerar as práticas recomendadas anteriores enquanto cria workloads usando VMSSs melhorará a escalabilidade de suas aplicações. Na próxima seção, vamos explorar o monitoramento.

## Monitoramento

O monitoramento é uma importante questão de arquitetura que deve ser parte de qualquer solução, seja ela grande ou pequena, de missão crítica ou não, baseada na nuvem ou não. Ele não deve ser negligenciado.

O monitoramento refere-se ao ato de manter o controle de soluções e capturar várias informações de telemetria, processá-las, identificar as informações que se qualificam para alertas com base em regras e criá-los. Geralmente, um agente é implantado dentro do ambiente e o monitora, enviando as informações de telemetria para um servidor centralizado, onde ocorrerá o restante do processamento de geração de alertas e notificação de stakeholders.

O monitoramento realiza ações proativas e reativas e avalia uma solução. Ele também é o primeiro passo da auditoria de uma solução. Sem a capacidade de monitorar registros de log, é difícil auditar um sistema em relação a vários aspectos, como segurança, performance e disponibilidade.

O monitoramento ajuda a identificar problemas de disponibilidade, performance e escalabilidade antes que eles aconteçam. Falhas de hardware, configuração incorreta de software e desafios de atualização de patches podem ser descobertos bem antes de afetar os usuários por meio do monitoramento, e a degradação de performance pode ser corrigida antes de acontecer.

O monitoramento de logs reativamente indica áreas e locais que estão causando problemas, identifica os problemas e permite reparos mais rápidos e melhores.

As equipes podem identificar padrões de problemas usando o monitoramento de informações de telemetria e os eliminam ao inovar com novas soluções e recursos.

O Azure é um ambiente de nuvem sofisticado que fornece várias funcionalidades e recursos sofisticados de monitoramento para monitorar não apenas a implantação baseada em nuvem, mas também a implantação na infraestrutura local.

## Monitoramento do Azure

A primeira pergunta deve ser respondida é: “O que devemos monitorar?” Essa questão se torna mais importante para soluções que são implantadas na nuvem devido ao seu controle restrito.

Há alguns componentes importantes que devem ser monitorados. Eles incluem:

- Aplicações personalizadas
- Recursos do Azure
- Sistemas operacionais convidados (VMs)
- Sistemas operacionais do host (servidores físicos do Azure)
- Infraestrutura do Azure

Há diferentes serviços de log e monitoramento do Azure para esses componentes, e eles serão discutidos nas seções a seguir.

## Logs de atividades do Azure

Anteriormente conhecidos como logs de auditoria e logs operacionais, eles são eventos do plano de controle na plataforma do Azure. Eles fornecem informações gerais e informações de telemetria no nível da assinatura, em vez do nível do recurso individual. Eles rastreiam informações sobre todas as alterações que ocorrem no nível da assinatura, como criação, exclusão e atualização dos recursos usando o **Azure Resource Manager (ARM)**. Os logs de atividades ajudam a descobrir a identidade (como entidade de serviço, usuários ou grupos) e realizar ações (como gravação ou atualização) nos recursos (por exemplo, armazenamento, máquinas virtuais ou bancos de dados SQL) a qualquer momento. Eles fornecem informações sobre os recursos que são modificados em sua configuração, mas não seu funcionamento e execução. Por exemplo, você pode obter os logs para iniciar, redimensionar ou interromper uma VM.

O próximo tópico que vamos discutir são os logs de diagnóstico.

## Logs de diagnóstico do Azure

As informações originadas do funcionamento interno dos recursos do Azure são capturadas nos **logs de diagnóstico**. Eles fornecem informações de telemetria sobre as operações de recursos que são inerentes aos recursos. Nem todo recurso fornece logs de diagnóstico, e os recursos que fornecem logs em seu próprio conteúdo são completamente diferentes de outros recursos. Os logs de diagnóstico são configurados para cada recurso individualmente. Exemplos de logs de diagnóstico incluem armazenar um arquivo em um contêiner em um Blob em uma conta de armazenamento.

O próximo tipo de log que vamos discutir são os logs de aplicações.

## Logs de aplicações do Azure

Os logs de aplicações podem ser capturados pelos recursos do Application Insights e gerenciados de maneira central. Eles obtêm informações sobre o funcionamento interno de aplicações personalizadas, como as métricas de performance disponibilidade, e os usuários podem obter insights deles para gerenciá-las melhor.

Por fim, temos os logs de sistema operacional convidado e do host. Vamos entender o que eles são.

## Logs de sistema operacional convidado e do host

Os logs de sistema operacional convidado e do host são oferecidos aos usuários que utilizam o recurso Azure Monitor. Eles fornecem informações sobre os status dos sistemas operacionais convidado e do host:

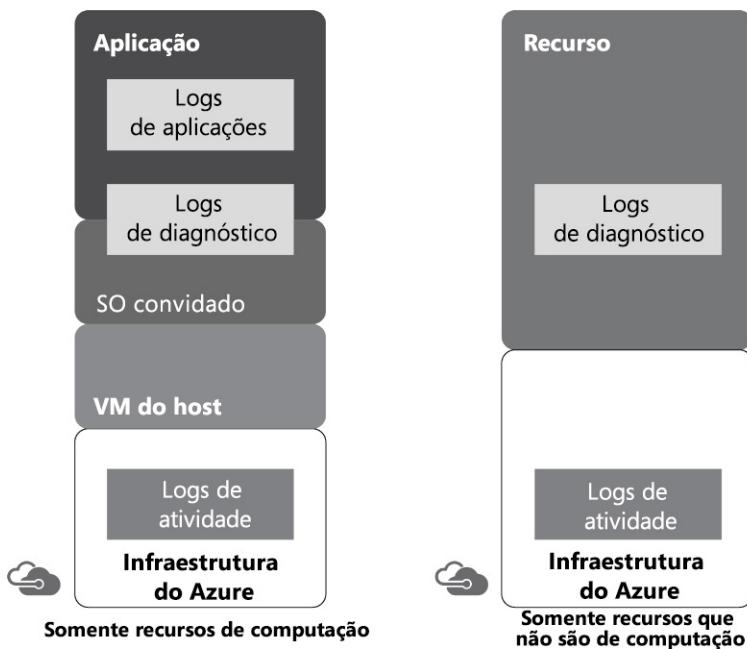


Figura 2.16: fazendo logon no Azure

Os importantes recursos do Azure relacionados ao monitoramento são o Azure Monitor, o Azure Application Insights e o Log Analytics, anteriormente conhecido como **Insights operacionais**.

Há outras ferramentas, como o **System Center Operations Manager (SCOM)**, que não fazem parte do recurso de nuvem, mas que podem ser implantadas em VMs baseadas em IaaS para monitorar qualquer workload no Azure ou em um datacenter na infraestrutura local. Vamos discutir os três recursos de monitoramento na seção a seguir.

## Azure Monitor

O Azure Monitor é um recurso e ferramenta central que fornece funcionalidades completas de gerenciamento que permitem monitorar uma assinatura do Azure. Ele fornece recursos de gerenciamento para logs de atividades, logs de diagnóstico, métricas, Application Insights e Log Analytics. Ele deve ser tratado como um recurso de gerenciamento e painel para todos os outros recursos de monitoramento.

Nosso próximo tópico é o Azure Application Insights.

## Azure Application Insights

O Azure Application Insights fornece recursos centralizados de métricas, logs e monitoramento na escala do Azure para aplicações personalizadas. As aplicações personalizadas podem enviar métricas, logs e outras informações de telemetria para o Azure Application Insights. Ele também fornece recursos de análise, painéis e relatórios para obter insights de dados de entrada e tomar medidas em relação a eles.

Agora que falamos sobre o Application Insights, vamos ver outro serviço semelhante chamado Azure Log Analytics.

## Azure Log Analytics

O Azure Log Analytics permite o processamento centralizado de logs e a geração de insights e alertas deles. Logs de atividades, logs de diagnóstico, logs de aplicações, logs de eventos e até mesmo logs personalizados podem enviar informações para o Log Analytics, que pode fornecer ainda mais recursos sofisticados de relatórios, painéis e análise para obter insights de dados de entrada e tomar medidas sobre eles.

Agora que conhecemos a finalidade do Log Analytics, vamos discutir como os logs são armazenados em um espaço de trabalho do Log Analytics e como eles podem ser consultados.

## Logs

O espaço de trabalho do Log Analytics fornece recursos de pesquisa para buscar entradas de log específicas, exportar todos os dados de telemetria para o Excel e/ou o Power BI e pesquisar uma linguagem de consulta chamada **linguagem de consulta Kusto (KQL)**, que é semelhante ao SQL.

Veja a seguir a tela **Pesquisa de logs**:

The screenshot shows the Azure Log Analytics interface. At the top, there's a navigation bar with 'Logs' selected, followed by 'New Query 1\*', a '+' button, 'Example queries', 'Query explorer', and a gear icon. Below the bar, it says 'Time range : Last 24 hours'. The main area has tabs for 'Tables', 'Queries', and 'Filter', with 'Tables' currently selected. A search bar contains the text 'Heartbeat'. On the left, a sidebar lists 'Favorites' (empty), 'Change Tracking' (ConfigurationChange, ConfigurationData), and 'LogManagement' (Alert, AppCenterError, ComputerGroup, Heartbeat, Operation, ReservedCommonFields, Usage). The main content area shows the results of the search: 'Completed. Showing results from the last 24 hours.' with a timestamp of '00:00:01.099' and '0 records'. A message at the bottom says 'NO RESULTS FOUND (last 24 hours)'.

Figura 2.17: pesquisa de logs em um espaço de trabalho do Log Analytics

Na próxima seção, veremos as soluções do Log Analytics, que são como recursos adicionais em um espaço de trabalho do Log Analytics.

## Soluções

As soluções no Log Analytics são recursos adicionais que podem ser adicionados a um espaço de trabalho com a captura de dados de telemetria adicionais que não são capturados por padrão. Quando essas soluções são adicionadas a um espaço de trabalho, Management Packs adequados são enviados para todos os agentes conectados ao espaço de trabalho para que possam se configurar para capturar dados específicos da solução de VMs e contêineres e, em seguida, enviá-los para o espaço de trabalho do Log Analytics. As soluções de monitoramento da Microsoft e de parceiros estão disponíveis no Azure Marketplace.

O Azure fornece muitas soluções do Log Analytics para controle e monitoramento de diferentes aspectos de ambientes e aplicações. No mínimo, um conjunto de soluções que são genéricas e aplicáveis a praticamente qualquer ambiente deve ser adicionado ao espaço de trabalho:

- Capacidade e performance
- Integridade do agente
- Controle de alterações
- Contêineres
- Segurança e auditoria
- Gerenciamento de atualizações
- Monitoramento de performance de rede

Outro aspecto fundamental do monitoramento são os alertas. Os alertas ajudam a notificar as pessoas certas durante qualquer evento monitorado. Na próxima seção, vamos abordar os alertas.

## Alertas

O Log Analytics permite gerar alertas em relação aos dados ingeridos. Ele faz isso por meio da execução de uma consulta predefinida composta por condições de dados de entrada. Se for encontrado um registro que se encaixe no âmbito dos resultados da consulta, será gerado um alerta. A Log Analytics fornece um ambiente altamente configurável para determinar as condições para a geração de alertas, janelas de tempo em que a consulta deve retornar os registros, janelas de tempo em que a consulta deve ser executada e a ação a ser realizada quando a consulta retorna um alerta:

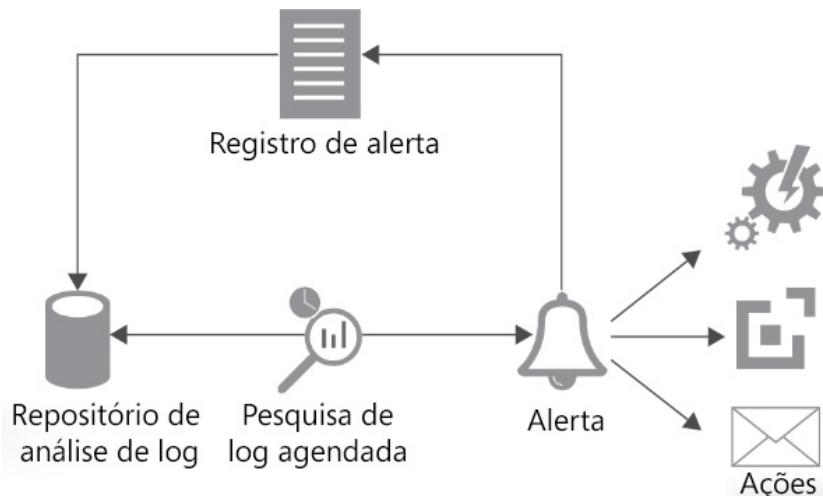


Figura 2.18: configurando alertas por meio do Log Analytics

Vamos percorrer as etapas para configurar alertas por meio do Log Analytics:

1. A primeira etapa na configuração de um alerta é adicionar uma nova regra de alerta do portal do Azure ou da automação do menu de alerta do recurso do Log Analytics.
2. No painel resultante, selecione um escopo para a regra de alerta. O escopo determina qual recurso deve ser monitorado para alertas – pode ser uma instância de recurso, como uma conta de armazenamento do Azure, um tipo de recurso, como uma VM do Azure, um grupo de recursos ou uma assinatura:

**Select a resource** X

Select the resource(s) you want to monitor. Available signal types for your selection will show up on the bottom right.

Filter by subscription \* ⓘ RiteshSubscription ▼

Filter by resource type ⓘ Virtual machines ▼

Filter by location ⓘ All ▼

Search to filter items...

Resource	Resource type	Location
<input type="checkbox"/> RiteshSubscription	Subscription	
<input type="checkbox"/> onlinetuesday	Resource group	West Europe
<input type="checkbox"/> vs2019docker	Virtual machine	West Europe
<input type="checkbox"/> rg-harvestingclouds-infra101	Resource group	East US
<input type="checkbox"/> vmAccounts101	Virtual machine	East US
<input type="checkbox"/> spark	Resource group	West Europe
<input type="checkbox"/> spark	Virtual machine	West Europe

Figura 2.19: selecionando um recurso para o alerta

3. Após a seleção de recursos, as condições devem ser definidas para o alerta. A condição determina a regra que é avaliada em relação aos logs e métricas no recurso selecionado, e um alerta só é gerado depois que a condição torna-se verdadeira. Há muitas métricas e logs disponíveis para gerar condições. No exemplo a seguir, um alerta é criado com um valor limite estático de 80% para **Porcentagem de CPU (Méd.)** e os dados devem ser coletados a cada cinco minutos e avaliados a cada minuto:

The screenshot shows the 'Alert logic' configuration for a metric named 'Percentage CPU (Avg) vs2019docker'. The configuration includes:

- Threshold:** Set to **Static**.
- Operator:** Set to **Greater than**.
- Aggregation type:** Set to **Average**.
- Threshold value:** Set to **80%**.
- Condition preview:** Text stating "Whenever the average percentage cpu is greater than 80 %".
- Evaluated based on:** Set to **5 minutes** for aggregation granularity and **Every 1 Minute** for frequency of evaluation.

Figura 2.20: criando um alerta para Porcentagem de CPU (Méd.)

Os alertas também oferecem suporte a limites dinâmicos, que usam o aprendizado de máquina para aprender o comportamento histórico das métricas e detectar irregularidades que podem indicar problemas de serviço.

4. Por fim, crie um grupo de ação ou reutilize um grupo existente que determine notificações sobre alertas para stakeholders. A seção **Grupos de ação** permite configurar coisas que devem seguir um alerta. Em geral, deve haver uma ação corretiva e/ou de notificação. O Log Analytics permite criar uma nova ação de oito maneiras diferentes. Você pode combiná-las como quiser. Um alerta executará todas ou qualquer uma das seguintes ações configuradas:
  - **Notificação por email/SMS/push/voz:** isso envia uma notificação por email/SMS/push/voz para os destinatários configurados.
  - **Webhooks:** executa um processo externo arbitrário usando um mecanismo de HTTP POST. Por exemplo, uma API REST pode ser executada. Caso contrário, o Service Manager/APIs ServiceNow podem ser invocados para criar um tíquete.
  - **Funções do Azure:** executa uma função do Azure, que passa a carga necessária e executa a lógica que a carga contém.
  - **Aplicativos Lógicos:** executa um fluxo de trabalho de Aplicativos Lógicos.
  - **Enviar email à função do Azure Resource Manager:** envia emails a um titular de uma função do Azure Resource Manager, como um proprietário, colaborador ou leitor.
  - **Webhook seguro:** executa um processo externo arbitrário usando um mecanismo de HTTP POST. Os webhooks são protegidos com um provedor de identidade, como o Azure Active Directory.
  - **Runbooks de automação:** essa ação executa runbooks de Automação do Azure.
  - **ITSM:** as soluções ITSM devem ser provisionadas antes do uso dessa opção. Elas ajudam a conectar e enviar informações para sistemas ITSM.
5. Depois de toda essa configuração, você precisa fornecer os valores **Nome**, **Descrição** e **Gravidade** para a regra de alerta para gerá-lo.

Como mencionado no início desta seção, os alertas desempenham um papel vital no monitoramento que ajuda o pessoal autorizado a tomar as medidas necessárias com base no alerta acionado.

## Resumo

A alta disponibilidade e a escalabilidade são preocupações de arquitetura muito importantes. Quase todos os arquitetos e aplicações tentam implementar a alta disponibilidade. O Azure é uma plataforma madura que comprehende a necessidade dessas preocupações de arquitetura em aplicações e fornece recursos para implementá-las em vários níveis. Essas preocupações de arquitetura não são um conceito tardio e devem fazer parte do desenvolvimento do ciclo de vida da aplicação, começando pela própria fase de planejamento.

O monitoramento é um aspecto de arquitetura importante de qualquer solução. Ele também é o primeiro passo para auditar uma aplicação corretamente. Ele permite que as operações gerenciem uma solução, tanto reativamente quanto proativamente. Ele oferece os registros necessários para solução e correção de problemas que podem surgir em plataformas e aplicações. Há muitos recursos no Azure que são específicos para implementar o monitoramento no Azure, outras nuvens e datacenters na infraestrutura local. O Application Insights e o Log Analytics são dois dos recursos mais importantes para isso. Evidentemente, o monitoramento é uma obrigatoriedade de melhorar suas soluções e produtos ao inovar com base em insights derivados de dados de monitoramento.

Este capítulo tratava apenas da disponibilidade, escalabilidade e monitoramento de soluções; o próximo capítulo é sobre padrões de design relacionados a redes virtuais, contas de armazenamento, regiões, zonas de disponibilidade e conjuntos de disponibilidade. Ao criar soluções na nuvem, esses princípios são muito importantes na criação de soluções econômicas com maior produtividade e disponibilidade.



# 3

## Padrão de design – redes, armazenamento, mensagens e eventos

No capítulo anterior, você teve uma visão geral da nuvem do Azure e conheceu alguns dos conceitos importantes relacionados a ela. Este capítulo apresenta padrões de nuvem do Azure relacionados a redes virtuais, contas de armazenamento, regiões, zonas de disponibilidade e conjuntos de disponibilidade. Esses são componentes importantes que afetam a arquitetura final fornecida aos clientes em termos de custo, eficiência e produtividade geral. O capítulo também discutirá brevemente os padrões de nuvem que nos ajudam a implementar escalabilidade e performance em uma arquitetura.

Neste capítulo, vamos abordar os seguintes tópicos:

- Design de rede virtual do Azure
- Design de armazenamento do Azure
- Zonas de disponibilidade, regiões e conjuntos de disponibilidade do Azure
- Padrões de design do Azure relacionados a mensagens, performance e escalabilidade

## Zonas e regiões de disponibilidade do Azure

O Azure tem o apoio de grandes datacenters interligados em uma rede única e ampla. Os datacenters são agrupados com base em sua proximidade física às regiões do Azure. Por exemplo, alguns datacenters na Europa Ocidental estão disponíveis para usuários do Azure na região da Europa Ocidental. Os usuários não podem escolher seu datacenter preferido. Eles podem selecionar sua região do Azure, e o Azure alocará um datacenter apropriado.

Selecionar uma região apropriada é uma decisão de arquitetura importante, pois afeta o seguinte:

- A disponibilidade dos recursos
- Conformidade de dados e privacidade
- A performance da aplicação
- O custo da execução das aplicações

Vamos discutir cada um desses pontos em detalhes.

### Disponibilidade dos recursos

Nem todos os recursos estão disponíveis em todas as regiões do Azure. Se sua arquitetura da aplicação exigir um recurso que não está disponível em uma região, escolher essa região não ajudará. Em vez disso, uma região deve ser escolhida com base na disponibilidade dos recursos exigidos pela aplicação. Pode ser que o recurso não esteja disponível durante o desenvolvimento da arquitetura da aplicação e pode estar no roteiro do Azure para disponibilizá-lo posteriormente.

Por exemplo, o Log Analytics não está disponível em todas as regiões. Se suas fontes de dados estiverem na Região A e o espaço de trabalho do Log Analytics estiver na região B, você precisará pagar pela largura de banda, que é a taxa de saída de dados da Região A para a B. Da mesma forma, alguns serviços podem funcionar com recursos que estão localizados na mesma região. Por exemplo, se você quiser criptografar os discos de sua máquina virtual que são implantados na Região A, precisa ter o Azure Key Vault implantado na Região A para armazenar as chaves de criptografia. Antes de implantar qualquer serviço, você precisa verificar se seus serviços de dependência estão disponíveis nessa região. Uma boa fonte para verificar a disponibilidade de produtos do Azure entre regiões é esta página do produto: <https://azure.microsoft.com/global-infrastructure/services>.

### Conformidade de dados e privacidade

Cada país tem suas próprias regras para conformidade de dados e privacidade. Alguns países são muito específicos sobre como armazenar os dados de seus cidadãos em seus próprios territórios. Portanto, tais requisitos legais devem ser levados em consideração na arquitetura de todas as aplicações.

## Performance de aplicações

A performance de uma aplicação depende da rota de rede adotada pelas solicitações e respostas para chegar ao seu destino e voltar. O local geograficamente mais perto de você nem sempre é a região com a menor latência. Calculamos a distância em quilômetros ou milhas, mas a latência baseia-se na rota que o pacote leva. Por exemplo, uma aplicação implantada na Europa Ocidental para usuários no sudeste da Ásia não terá uma performance tão boa quanto uma aplicação implantada na região do leste da Ásia para usuários dessa região. Portanto, é muito importante que você arquitete suas soluções na região mais próxima para fornecer a menor latência e, assim, a melhor performance.

## Custo da execução das aplicações

O custo dos serviços do Azure difere de região para região. Deve-se escolher uma região com um custo geral mais baixo. Há um capítulo completo sobre gerenciamento de custos neste livro (*Capítulo 6: Gerenciamento de custos para soluções do Azure*), que deve ser consultado para mais detalhes sobre custos.

Até o momento, discutimos como escolher a região certa para arquitetar nossa solução. Agora que temos uma região adequada em mente para nossa solução, vamos discutir como criar nossas redes virtuais no Azure.

## Redes virtuais

As redes virtuais devem ser consideradas como uma configuração de rede LAN física em sua casa ou escritório. Conceitualmente, elas são iguais, embora a **Rede Virtual do Azure (VNet)** seja implementada como uma rede definida por software com o apoio de uma infraestrutura de rede física gigante.

Uma VNet é necessária para hospedar uma máquina virtual. Ela fornece um mecanismo de comunicação segura entre os recursos do Azure para que eles se conectem entre si. As VNets fornecem endereços IP internos aos recursos, facilitam o acesso e a conectividade a outros recursos (incluindo máquinas virtuais na mesma rede virtual), encaminham solicitações e fornecem conectividade com outras redes.

Uma rede virtual está contida em um grupo de recursos e é hospedada em uma região, por exemplo, Europa Ocidental. Ela não pode abranger várias regiões, mas pode abranger todos os datacenters dentro de uma região. Portanto, podemos abranger redes virtuais em várias zonas de disponibilidade em uma região. Para a conectividade entre regiões, as redes virtuais podem ser conectadas usando a conectividade de VNet a VNet.

As redes virtuais também fornecem conectividade a datacenters na infraestrutura local, habilitando as nuvens híbridas. Há vários tipos de tecnologias VPN que você pode usar para estender seus datacenters na infraestrutura local para a nuvem, como VPN site a site e VPN ponto a site. Há também uma conectividade dedicada entre a VNet do Azure e as redes na infraestrutura local com o uso do ExpressRoute.

As redes virtuais são gratuitas. Cada assinatura pode criar até 50 redes virtuais em todas as regiões. No entanto, é possível entrar em contato com o suporte do Azure para aumentar esse número. Você não será cobrado se os dados não saírem da região de implantação. No momento da gravação, as transferências de dados de entrada e saída em zonas de disponibilidade da mesma região não geram encargos. No entanto, a cobrança começará a partir de 1º de julho de 2020.

As informações sobre limites de rede estão disponíveis na documentação da Microsoft em <https://docs.microsoft.com/azure/azure-resource-manager/management/azure-subscription-service-limits>.

## Considerações sobre a arquitetura de redes virtuais

As redes virtuais, como qualquer outro recurso, podem ser provisionadas usando modelos do ARM, APIs REST, PowerShell e a CLI. É muito importante planejar a topologia de rede o quanto antes para evitar problemas mais tarde no ciclo de vida de desenvolvimento. Isso ocorre porque, depois que uma rede é provisionada e os recursos começam a usá-la, é difícil mudá-la sem haver um tempo de inatividade. Por exemplo, para mover uma máquina virtual de uma rede para outra é necessário desligar a máquina virtual.

Vamos analisar algumas das principais considerações de arquitetura ao criar uma rede virtual.

### Regiões

A VNet é um recurso do Azure e é provisionada em uma região, como Europa Ocidental. As aplicações que abrangem várias regiões precisam de redes virtuais separadas, uma por região, e também precisam ser conectadas usando a conectividade de VNet a VNet. Há um custo associado à conectividade de VNet a VNet para os tráfegos de entrada e de saída. Não há cobranças por dados de entrada (ingresso), mas há encargos associados aos dados de saída.

## DNS dedicado

A VNet, por padrão, usa o DNS do Azure para resolver nomes em uma rede virtual, e isso também permite a resolução de nomes na Internet. Se uma aplicação quiser um serviço de resolução de nome dedicado ou se conectar aos datacenters na infraestrutura local, ela deverá provisionar seu próprio servidor DNS, que deverá ser configurado na rede virtual para a resolução de nomes ser bem-sucedida. Além disso, você pode hospedar seu domínio público no Azure e gerenciar completamente os registros do portal do Azure, sem a necessidade de gerenciar servidores DNS adicionais.

## Número de redes virtuais

O número de redes virtuais é afetado por fatores como número de regiões, uso de largura de banda pelos serviços, conectividade entre regiões e segurança. Ter menos VNets maiores, em vez de várias VNets menores, eliminará a sobrecarga de gerenciamento.

## Número de sub-redes em cada rede virtual

As sub-redes fornecem isolamento dentro de uma rede virtual. Elas também podem fornecer um limite de segurança. **Grupos de segurança de rede (NSGs)** podem ser associados a sub-redes, restringindo ou permitindo o acesso específico a endereços IP e portas. Os componentes da aplicação com requisitos de segurança e acessibilidade separados devem ser colocados em sub-redes separadas.

## Intervalos de IP para redes e sub-redes

Cada sub-rede tem um intervalo de IP. O intervalo de IP não deve ser tão grande de modo que os IPs sejam subutilizados, mas, inversamente, não devem ser tão pequenos de modo que as sub-redes sejam sufocadas devido à falta de endereços IP. Isso deve ser considerado após a compreensão das necessidades futuras de endereços IP da implantação.

O planejamento deve ser feito para intervalos e endereços IP para redes do Azure, sub-redes e datacenters na infraestrutura local. Não deve haver sobreposição para garantir conectividade e acessibilidade sem problemas.

## Monitoramento

O monitoramento é uma faceta importante da arquitetura e deve ser incluído na implantação geral. O Observador de Rede do Azure fornece recursos de registro e diagnóstico com insights sobre a performance e a integridade da rede. Alguns dos recursos do Observador de Rede do Azure são:

- Diagnosticar problemas de filtragem de tráfego de rede de ou para uma máquina virtual
- Entender o próximo salto de rotas definidas pelo usuário
- Visualizar os recursos em uma rede virtual e seus relacionamentos
- Monitoramento de comunicação entre uma máquina virtual e um ponto de extremidade
- Captura de tráfego de uma máquina virtual
- Logs de fluxo de NSG, que registram informações relacionadas ao tráfego que fluem por meio de um NSG. Esses dados ficarão no Armazenamento do Azure para análise posterior

Ele também fornece logs de diagnóstico para todos os recursos de rede em um grupo de recursos.

A performance da rede pode ser monitorada por meio do Log Analytics. A solução de gerenciamento do Monitor de Performance da Rede fornece o recurso de monitoramento de rede. Ele monitora a integridade, disponibilidade e acessibilidade das redes. Esse recurso também é usado para monitorar a conectividade entre a nuvem pública e as sub-redes da infraestrutura local que hospedam várias camadas de uma aplicação multicamadas.

## Considerações sobre segurança

As redes virtuais estão entre os primeiros componentes a serem acessados por qualquer recurso no Azure. A segurança desempenha um papel importante na permissão ou negação de acesso a um recurso. Os NSGs são os principais meios de segurança das redes virtuais. Eles podem ser conectados a sub-redes de rede virtual, e cada fluxo de entrada e saída é restrinido, filtrado e permitido por eles.

O **roteamento definido pelo usuário (UDR)** e o encaminhamento de IP também ajudam na filtragem e no roteamento de solicitações para recursos no Azure. Você pode ler mais sobre UDR e túnel forçado em <https://docs.microsoft.com/azure/virtual-network/virtual-networks-udr-overview>.

O Firewall do Azure é um firewall totalmente gerenciado como uma oferta de serviço do Azure. Ele pode ajudá-lo a proteger os recursos em sua rede virtual. O Firewall do Azure pode ser usado para filtragem de pacotes no tráfego de entrada e de saída, entre outras coisas. Além disso, o recurso de inteligência contra ameaças do Firewall do Azure pode ser usado para alertar e negar o tráfego de ou para endereços IP e domínios maliciosos. A fonte de dados para endereços IP e domínios é o feed de inteligência contra ameaças da Microsoft.

Os recursos também podem ser protegidos por meio da implantação de dispositivos de rede (<https://azure.microsoft.com/solutions/network-appliances>) como Barracuda, F5 e outros componentes de terceiros.

## Implantação

As redes virtuais devem ser implantadas em seus próprios grupos de recursos dedicados. Os administradores de rede devem ter a permissão do proprietário para usar esse grupo de recursos, enquanto os desenvolvedores ou membros da equipe devem ter permissões de colaborador para permitir que eles criem outros recursos do Azure em outros grupos de recursos que consomem serviços da rede virtual.

Também é recomendável implantar recursos com endereços IP estáticos em uma sub-rede dedicada, enquanto os recursos relacionados ao endereço IP dinâmico podem ficar em outra sub-rede.

Políticas devem ser criadas para que somente os administradores de rede possam excluir a rede virtual, mas também deve ser marcada para fins de cobrança.

## Conectividade

Os recursos em uma região de uma rede virtual podem interagir perfeitamente. Até mesmo os recursos de outras sub-redes em uma rede virtual podem interagir sem nenhuma configuração explícita. Os recursos de várias regiões não podem usar a mesma rede virtual. O limite de uma rede virtual é dentro de uma região. Para fazer um recurso se comunicar entre regiões, precisamos de gateways dedicados em ambas as extremidades para facilitar a conversa.

Dito isso, se você quiser iniciar uma conexão privada entre duas redes em diferentes regiões, poderá usar o emparelhamento VNet global. Com o emparelhamento VNet global, a comunicação é feita por meio da rede de backbone da Microsoft, o que significa que nenhum gateway, criptografia ou Internet pública é necessária durante a comunicação. Se suas redes virtuais estiverem na mesma região com diferentes espaços de endereço, os recursos em uma rede não poderão se comunicar com a outra. Como eles estão na mesma região, podemos usar o emparelhamento de rede virtual, que é semelhante ao emparelhamento VNet global; a única diferença é que as redes virtuais de origem e destino são implantadas na mesma região.

Como muitas organizações têm uma nuvem híbrida, os recursos do Azure às vezes precisam se comunicar ou se conectar com datacenters na infraestrutura local ou vice-versa. As redes virtuais do Azure podem se conectar aos datacenters na infraestrutura local usando a tecnologia VPN e o ExpressRoute. Na verdade, uma rede virtual é capaz de se conectar a vários datacenters na infraestrutura local e a outras regiões do Azure em paralelo. Como prática recomendada, cada uma dessas conexões deve estar em suas sub-redes dedicadas dentro de uma rede virtual.

Agora que exploramos vários aspectos das redes virtuais, vamos discutir os benefícios delas.

## Benefícios das redes virtuais

As redes virtuais são essenciais para a implantação de qualquer solução IaaS significativa. As máquinas virtuais não podem ser provisionadas sem redes virtuais. Além de serem quase um componente obrigatório nas soluções de IaaS, elas fornecem ótimos benefícios arquitetônicos. Alguns deles são descritos aqui:

- **Isolamento:** a maioria dos componentes da aplicação tem requisitos de segurança e largura de banda separados e tem um gerenciamento de ciclo de vida diferente. As redes virtuais ajudam a criar bolsos isolados para esses componentes que podem ser gerenciados independentemente de outros componentes com a ajuda de redes virtuais e sub-redes.
- **Segurança:** filtrar e rastrear os usuários que estão acessando os recursos é uma funcionalidade importante fornecida pelas redes virtuais. Isso pode impedir o acesso a portas e endereços IP mal-intencionados.
- **Extensibilidade:** as redes virtuais atuam como uma LAN privada na nuvem. Elas também podem ser estendidas para uma **Rede de longa distância (WAN)** conectando outras redes virtuais do mundo todo e podem ser extensões dos datacenters na infraestrutura local.

Exploramos os benefícios das redes virtuais. Agora, a questão é como podemos aproveitar esses benefícios e criar uma rede virtual para hospedar nossa solução. Na próxima seção, analisaremos o design das redes virtuais.

## Design da rede virtual

Nesta seção, vamos considerar alguns dos cenários de design e uso conhecidos de redes virtuais.

Pode haver vários usos de redes virtuais. Um gateway pode ser implantado em cada ponto de extremidade de rede virtual para viabilizar a segurança e transmitir pacotes com integridade e confidencialidade. Um gateway é imprescindível para a conexão com redes na infraestrutura local. No entanto, ele é opcional quando se usa o emparelhamento de VNet do Azure. Além disso, você pode usar o recurso Gateway Transit para simplificar o processo de estender seu datacenter na infraestrutura local sem implantar vários gateways. O Gateway Transit permite compartilhar um gateway do ExpressRoute ou VPN com todas as redes virtuais emparelhadas. Isso facilitará o gerenciamento e a redução do custo de implantação de vários gateways.

Na seção anterior, falamos sobre emparelhamento e mencionamos que não usamos gateways ou a Internet pública para estabelecer a comunicação entre redes emparelhadas. Vamos seguir em frente e explorar alguns dos aspectos de design do emparelhamento, e qual emparelhamento precisa ser usado em cenários específicos.

## Coneção com recursos dentro da mesma região e assinatura

Várias redes virtuais dentro da mesma região e assinatura podem ser conectadasumas às outras. Com a ajuda do emparelhamento de VNet, ambas as redes podem ser conectadas e usar o backbone de rede privada do Azure para transmitir pacotes entre si. As máquinas virtuais e os serviços nessas redes podem interagir entre si, sujeitos às restrições de tráfego da rede. No diagrama a seguir, VNet1 e VNet2 são implantadas na região Oeste dos EUA. No entanto, o espaço de endereço para a VNet1 é 172.16.0.0/16 e, para a VNet2, é 10.0.0.0/16. Por padrão, os recursos na VNet1 não poderão se comunicar com recursos na VNet2. Como estabelecemos o emparelhamento VNet entre os dois, os recursos poderão se comunicar entre si por meio da rede de backbone da Microsoft:

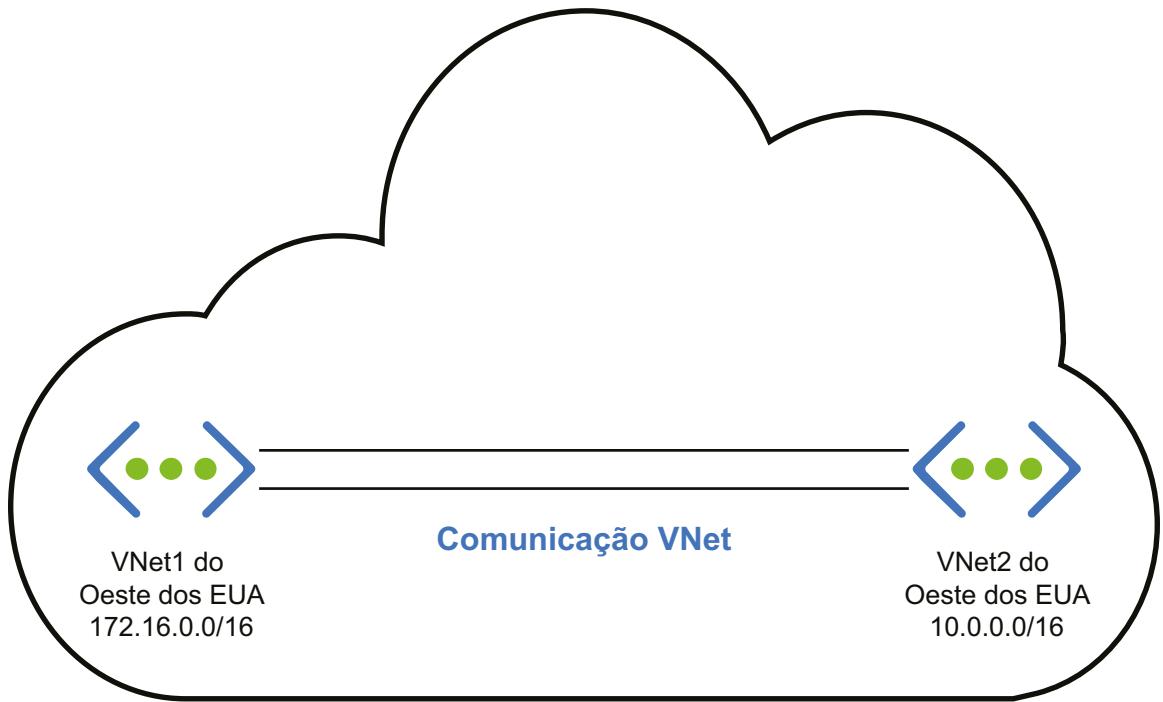


Figura 3.1: o emparelhamento VNet de recursos com a mesma assinatura

## Coneção com recursos dentro da mesma região em outra assinatura

Esse cenário é muito semelhante ao anterior, exceto que as redes virtuais são hospedadas em duas assinaturas diferentes. As assinaturas podem fazer parte do mesmo locatário ou de vários locatários. Se ambos os recursos fizerem parte da mesma assinatura e da mesma região, o cenário anterior se aplicará. Esse cenário pode ser implementado de duas maneiras: usando gateways ou o emparelhamento de rede virtual.

Se estamos usando gateways neste cenário, precisamos implantar um gateway em ambas as extremidades para facilitar a comunicação. Veja a seguir a representação arquitetônica do uso de gateways para conectar dois recursos com assinaturas diferentes:

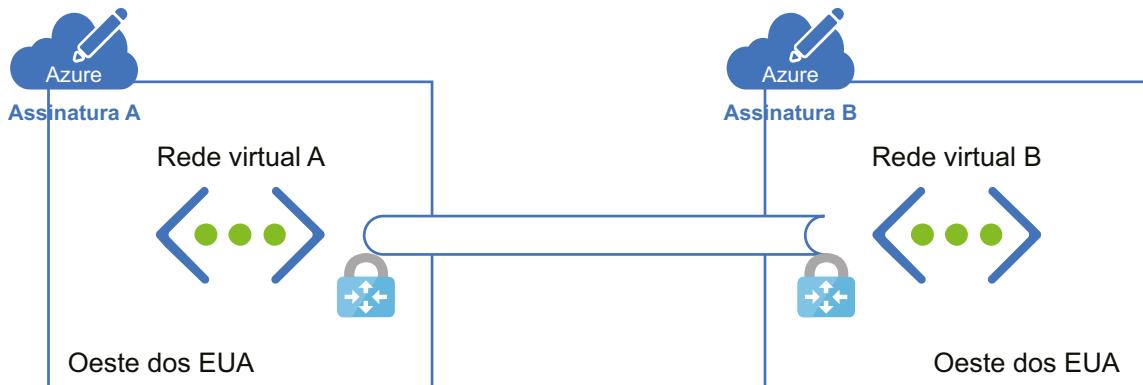


Figura 3.2: emparelhamento VNet de recursos com diferentes assinaturas usando gateways

No entanto, a implantação de gateways acarreta alguns encargos. Discutiremos o emparelhamento VNet e, depois, compararemos essas duas implementações para ver o que é melhor para a nossa solução.

Ao usar o emparelhamento, não implantamos gateways. A Figura 3.3 representa como o emparelhamento é feito:

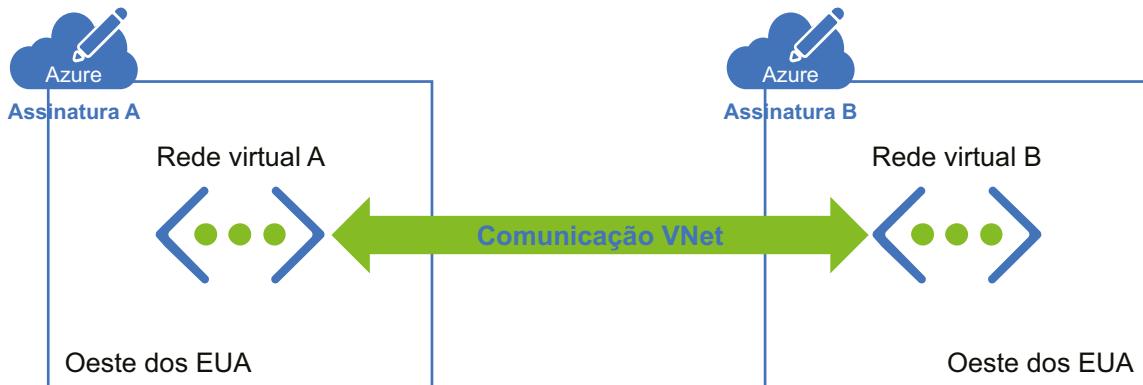


Figura 3.3: emparelhamento VNet entre assinaturas

O emparelhamento VNet fornece uma conexão de baixa latência com alta largura de banda e, como mostrado no diagrama, não estamos implantando gateways para fazer a comunicação acontecer. Isso é útil para cenários como replicação de dados ou failover. Como mencionado anteriormente, o emparelhamento usa a rede de backbone da Microsoft, que elimina a necessidade da Internet pública.

Os gateways são usados em cenários onde a criptografia é necessária e a largura de banda não é uma preocupação, pois essa será uma conexão de largura de banda limitada. No entanto, isso não significa que haja uma restrição na largura de banda. Além disso, essa abordagem é usada onde os clientes não são tão sensíveis à latência.

Até agora, analisamos recursos na mesma região em assinaturas. Na próxima seção, exploraremos como estabelecer uma conexão entre redes virtuais em duas regiões diferentes.

### Coneção com recursos em diferentes regiões em outra assinatura

Neste cenário, temos duas implementações novamente. Um usa um gateway, e o outro usa o emparelhamento VNet global.

O tráfego passará pela rede pública, e teremos gateways implantados em ambas as extremidades para facilitar uma conexão criptografada. A Figura 3.4 explica como isso é feito:

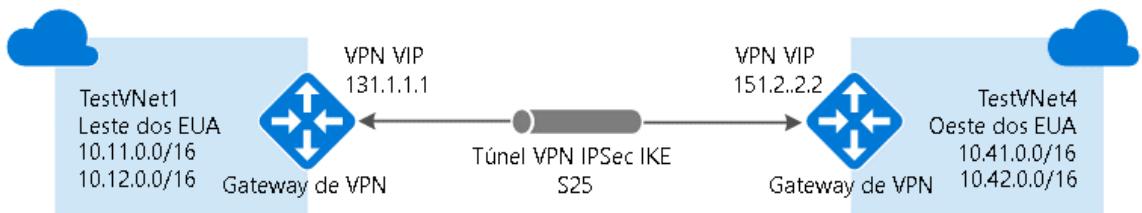


Figura 3.4: conectando recursos em diferentes regiões com assinaturas diferentes

Vamos adotar uma abordagem semelhante usando o emparelhamento VNet global. A Figura 3.5 mostra como o emparelhamento VNet global é feito:



Figura 3.5: conectando recursos em diferentes regiões usando o emparelhamento VNet global

As considerações sobre a escolha de gateways ou emparelhamento já foram discutidas. Essas considerações também são aplicáveis nesse cenário. Até agora, conectamos redes virtuais entre regiões e assinaturas. Entretanto, ainda não falamos sobre como conectar um datacenter na infraestrutura local à nuvem. Na próxima seção, falaremos sobre como fazer isso.

## Conectando-se a datacenters na infraestrutura local

As redes virtuais podem ser conectadas a datacenters na infraestrutura local de forma que o Azure e datacenters na infraestrutura local tornem-se uma única WAN. Uma rede na infraestrutura local precisa ser implantada em gateways e VPNs em ambos os lados da rede. Existem três tecnologias diferentes disponíveis para essa finalidade.

### VPN site a site

Deve ser usada quando a rede do Azure e o datacenter na infraestrutura local estão conectados para formar uma WAN, onde qualquer recurso nas duas redes pode acessar qualquer outro recurso nas redes, independentemente de serem implantados no Azure ou em um datacenter na infraestrutura local. Os gateways de VPN precisam estar disponíveis nos dois lados das redes por motivos de segurança. Além disso, os gateways do Azure devem ser implantados em suas próprias sub-redes em redes virtuais conectadas a datacenters na infraestrutura local. Endereços IP públicos devem ser atribuídos aos gateways na infraestrutura local para o Azure conectar-se a eles pela rede pública:

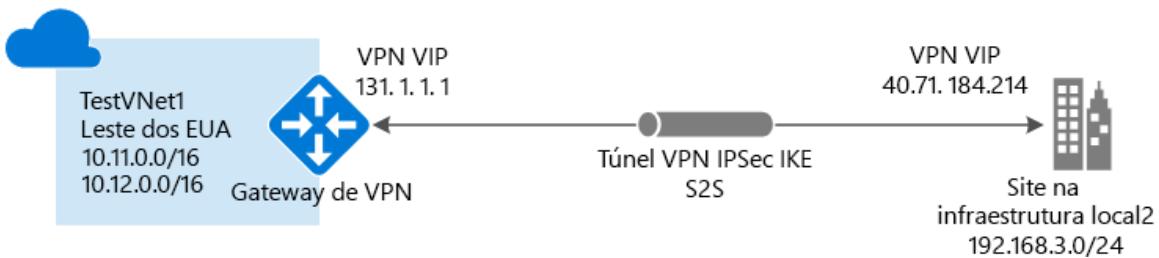


Figura 3.6: arquitetura de VPN site a site

## VPN ponto a site

Similar à conectividade VPN site a site. No entanto, há um único servidor ou computador conectado ao datacenter na infraestrutura local. Ela deve ser usada quando há pouquíssimos usuários ou clientes que se conectariam ao Azure com segurança a partir de locais remotos. Além disso, não há necessidade de gateways e IPs públicos na infraestrutura local neste caso:

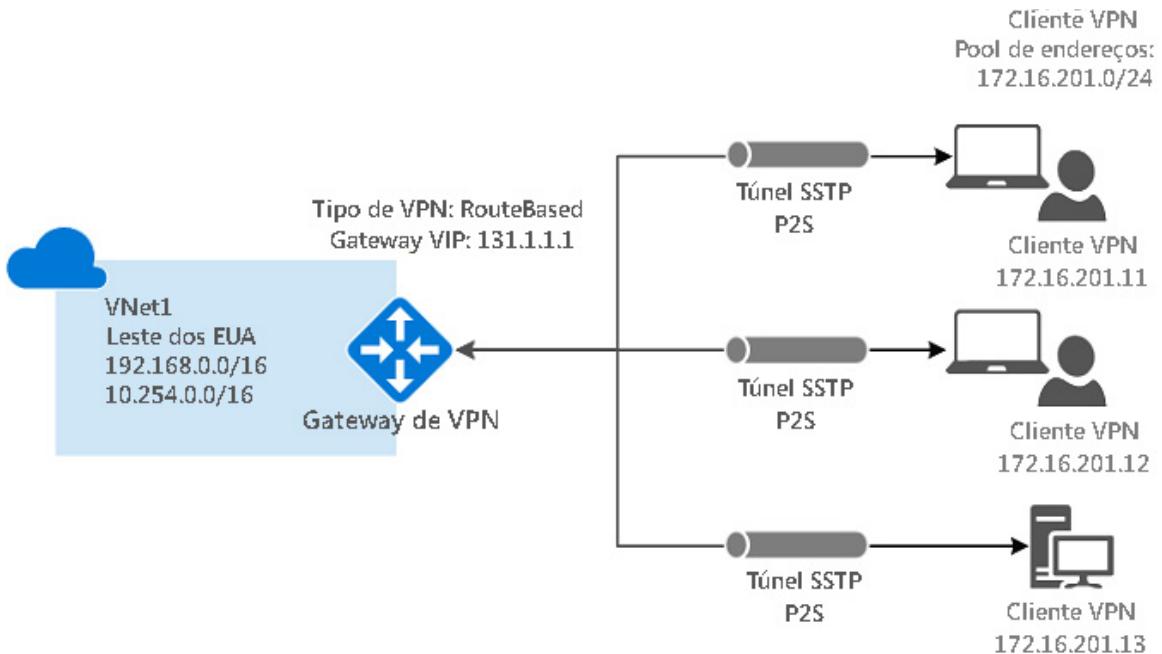


Figura 3.7: arquitetura de VPN ponto a site

## ExpressRoute

As conectividades VPNs site a site e ponto a site funcionam usando a Internet pública. Elas criptografam o tráfego nas redes usando a tecnologia de VPN e certificados. No entanto, há aplicações que devem ser implantadas com tecnologias híbridas – alguns componentes no Azure, com outros em um datacenter na infraestrutura local – e, ao mesmo tempo, não devem usar a Internet pública para se conectar a datacenters do Azure e na infraestrutura local. O Azure ExpressRoute é a melhor solução para eles, embora seja uma opção cara em comparação com os outros dois tipos de conexão. Ele também é o provedor mais seguro e confiável, com maior velocidade e latência reduzida porque o tráfego nunca atinge a Internet pública. O Azure ExpressRoute pode ajudar a estender as redes na infraestrutura local para o Azure por meio de uma conexão privada dedicada facilitada por um provedor de conectividade: Se sua solução for de rede intensiva, por exemplo, uma aplicação corporativa transacional como o SAP, o uso do ExpressRoute é altamente recomendável.

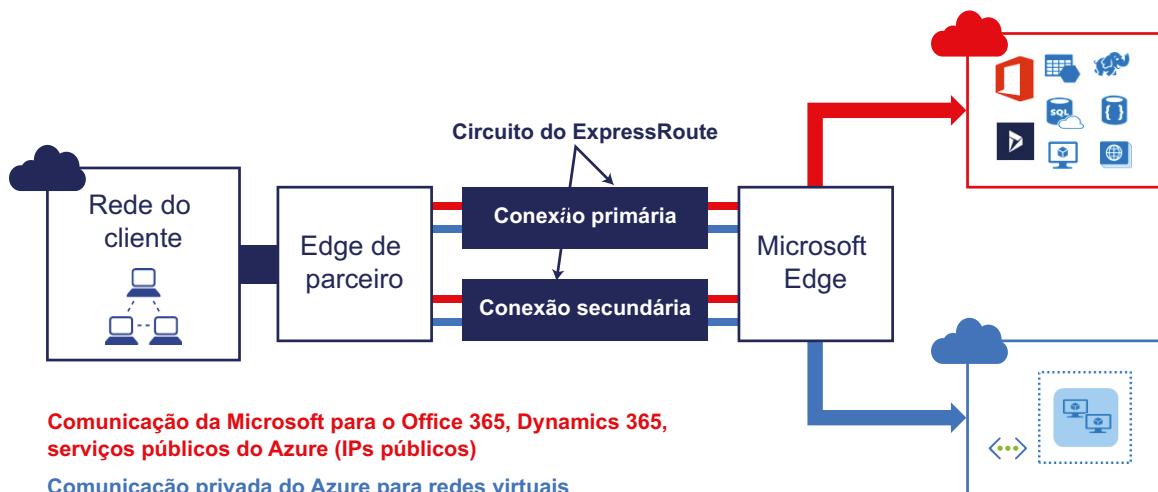


Figura 3.8: arquitetura de rede ExpressRoute

A Figura 3.9 mostra os três tipos de redes híbridas:

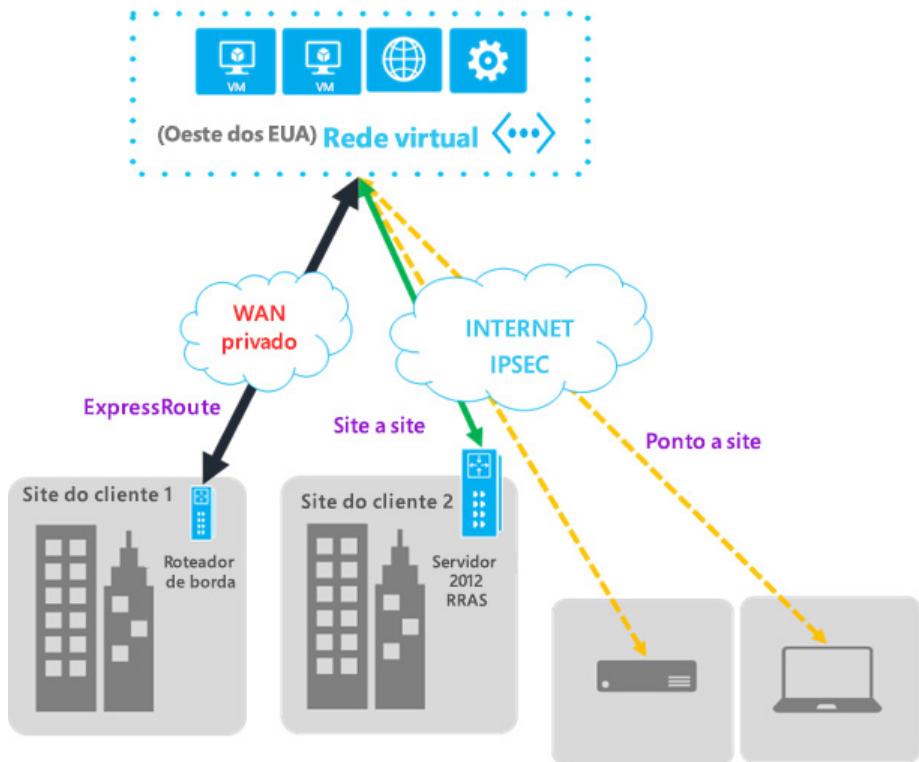


Figura 3.9: diferentes tipos de redes híbridas

É uma boa prática para as redes virtuais terem sub-redes separadas para cada componente lógico com implantações separadas da perspectiva de segurança e isolamento.

Todos os recursos que implantamos no Azure exigem redes de uma forma ou de outra, portanto, uma compreensão profunda da rede é necessária ao arquitetar soluções no Azure. Outro elemento-chave é o armazenamento. Na próxima seção, você saberá mais sobre o armazenamento.

## Armazenamento

O Azure fornece uma solução de armazenamento durável, altamente disponível e escalável por meio de serviços de armazenamento.

O armazenamento é usado para persistência de dados para as necessidades de longo prazo. O armazenamento do Azure está disponível na Internet para quase todas as linguagens de programação.

### Categorias de armazenamento

O armazenamento tem duas categorias de contas de armazenamento:

- Uma camada de performance de armazenamento padrão que permite armazenar tabelas, filas, arquivos, blobs e discos de máquina virtual do Azure.
- Uma camada de performance de armazenamento Premium que dá suporte aos discos de máquina virtual do Azure no momento da gravação. O armazenamento Premium fornece maior performance e IOPS do que o armazenamento padrão geral. O armazenamento Premium está atualmente disponível como discos de dados para máquinas virtuais com backup de SSDs.

Dependendo do tipo de dados que estão sendo armazenados, o armazenamento é classificado em diferentes tipos. Vamos analisar os tipos de armazenamento e saber mais sobre eles.

### Tipos de armazenamento

O Azure fornece quatro tipos de serviços de armazenamento geral:

- **Armazenamento de blobs do Azure:** esse tipo de armazenamento é mais adequado para dados não estruturados, como documentos, imagens e outros tipos de arquivos. O armazenamento de blobs pode estar na camada de acesso frequente, de acesso esporádico ou de arquivamento. A camada de acesso frequente é destinada ao armazenamento de dados que precisam ser acessados com muita frequência. A camada de acesso esporádico é para dados acessados com menos frequência do que os da camada de acesso frequente. Eles são armazenados por 30 dias. Por fim, a camada de arquivamento é para fins de arquivamento, em que a frequência de acesso é muito baixa.
- **Armazenamento de tabelas do Azure:** é um repositório de dados de atributo-chave NoSQL. Ele deve ser usado para dados estruturados. Os dados são armazenados como entidades.
- **Armazenamento de filas do Azure:** fornece armazenamento confiável de mensagens para armazenar grandes quantidades delas. Essas mensagens podem ser acessadas de qualquer lugar por meio de chamadas HTTP ou HTTPS. Uma mensagem de fila pode ter até 64 KB de tamanho.

- **Arquivos do Azure:** armazenamento compartilhado baseado no protocolo SMB. Normalmente é usado para armazenar e compartilhar arquivos. Também armazena dados não estruturados, mas a principal diferença é que ele é compartilhável pelo protocolo SMB.
- **Discos do Azure:** é o armazenamento em nível de bloco para Máquinas Virtuais do Azure.

Esses cinco tipos de armazenamento atendem a diferentes requisitos de arquitetura e abrangem quase todos os tipos de instalações de armazenamento de dados.

## Recursos de armazenamento

O armazenamento do Azure é flexível. Isso significa que você pode armazenar apenas alguns megabytes ou até mesmo petabytes de dados. Você não precisa pré-bloquear a capacidade. Ela aumenta e diminui automaticamente. Os consumidores precisam pagar apenas pelo uso real de armazenamento. Veja alguns dos principais benefícios do uso do Armazenamento do Azure:

- O armazenamento do Azure é seguro. Ele só pode ser acessado usando o protocolo SSL. Além disso, o acesso deve ser autenticado.
- O armazenamento do Azure fornece a facilidade de gerar um token de **assinatura de acesso seguro (SAS)** no nível da conta que pode ser usado pelos clientes de armazenamento para se autenticarem. Também é possível gerar tokens SAS de nível de serviço individuais para blobs, filas, tabelas e arquivos.
- Os dados salvos no armazenamento do Azure podem ser criptografados. Isso é conhecido como proteção de dados em repouso.
- A criptografia de disco do Azure é usada para criptografar o sistema operacional e os discos de dados em máquinas virtuais IaaS. **Criptografia do cliente (CSE)** e **Criptografia do serviço de armazenamento (SSE)**: ambas são usadas para criptografar dados no armazenamento do Azure. SSE é uma configuração do armazenamento do Azure que garante que os dados sejam criptografados durante a gravação no armazenamento e sejam descriptografados durante a leitura pelo mecanismo de armazenamento. Isso garante que nenhuma alteração na aplicação seja necessária para ativar a SSE. Na CSE, as aplicações cliente podem usar o SDK de armazenamento para criptografar os dados antes que eles sejam enviados e gravados no armazenamento do Azure. A aplicação cliente poderá descriptografá-los posteriormente durante a leitura. Isso fornece segurança para dados em trânsito e em repouso. A CSE depende de segredos do Azure Key Vault.
- O armazenamento do Azure é altamente disponível e durável. Isso significa que o Azure sempre mantém várias cópias das contas dessa solução. A localização e o número de cópias dependem da configuração de replicação.

O Azure fornece as seguintes configurações de replicação e opções de redundância de dados:

- **Armazenamento com redundância local (LRS):** dentro de um único local físico na região primária, haverá três réplicas dos seus dados de maneira síncrona. Do ponto de vista de cobrança, essa é a opção mais barata. No entanto, ela não é recomendada para soluções que exigem alta disponibilidade. O LRS fornece um nível de durabilidade de 99.999999999% para objetos em um determinado ano.
- **Armazenamento com redundância de zona (ZRS):** no caso do LRS, as réplicas eram armazenadas no mesmo local físico. No caso do ZRS, os dados serão replicados de forma síncrona nas zonas de disponibilidade na região primária. Como cada uma dessas zonas de disponibilidade é um local físico separado na região primária, o ZRS oferece melhor durabilidade e maior disponibilidade do que o LRS.
- **Armazenamento com redundância geográfica (GRS):** o GRS aumenta a alta disponibilidade, replicando sincronicamente três cópias de dados em uma única região primária usando LRS. Ele também copia os dados para um único local físico na região secundária.
- **Armazenamento com redundância de zona geográfica (GZRS):** isso é muito semelhante ao GRS, mas em vez de replicar dados em um único local físico na região primária, o GZRS replica de forma síncrona em três zonas de disponibilidade. Como discutimos no caso do ZRS, como as zonas de disponibilidade são locais físicos isolados na região primária, o GZRS tem melhor durabilidade e pode ser incluído em designs altamente disponíveis.
- **Armazenamento com redundância geográfica de acesso de leitura (RA-GRS) e armazenamento com redundância de zona geográfica e acesso de leitura:** os dados replicados para a região secundária por GZRS ou GRS não estão disponíveis para leitura ou gravação. Esses dados serão usados pela região secundária no caso do failover do datacenter primário. Os RA-GRS e os RA-GZRS seguem o mesmo padrão de replicação que o GRS e o GZRS, respectivamente; a única diferença é que os dados replicados para a região secundária via RA-GRS ou RA-GZRS podem ser lidos.

Agora que conhecemos as várias opções de armazenamento e conexão disponíveis no Azure, vamos saber mais sobre a arquitetura subjacente da tecnologia.

## Considerações sobre arquitetura para contas de armazenamento

As contas de armazenamento devem ser provisionadas na mesma região dos outros componentes da aplicação. Isso significaria usar o mesmo backbone de rede do datacenter sem haver cobrança de tarifas de rede.

Os serviços de armazenamento do Azure têm metas de escalabilidade para capacidade, taxa de transações e largura de banda associadas a cada um deles. Uma conta de armazenamento geral permite o armazenamento de 500 TB de dados. Se houver necessidade de armazenar mais de 500 TB de dados, várias contas de armazenamento devem ser criadas ou o armazenamento Premium deve ser usado.

O armazenamento geral é executado no máximo a 20.000 IOPS ou 60 MB de dados por segundo. Qualquer requisito para um IOPS mais alto ou dados gerenciados por segundo será limitado. Se isso também não for suficiente para suas aplicações do ponto de vista de performance, será preciso usar o armazenamento Premium ou várias contas de armazenamento. Para uma conta, o limite de escalabilidade para acessar tabelas é de até 20.000 (1 KB cada) entradas. A contagem de entidades inseridas, atualizadas, excluídas ou digitalizadas contribuirão para o destino. Uma única fila pode processar aproximadamente 2.000 mensagens (1 KB cada) por segundo, e cada uma das contagens de **AddMessage**, **GetMessage** e **DeleteMessage** será tratada como uma mensagem. Se esses valores não forem suficientes para sua aplicação, você deverá espalhar as mensagens em várias filas.

O tamanho das máquinas virtuais determina o tamanho e a capacidade dos discos de dados disponíveis. Embora as máquinas virtuais maiores tenham discos de dados com maior capacidade de IOPS, a capacidade máxima ainda será limitada a 20.000 IOPS e 60 MB por segundo. É importante notar que esses são os números máximos e, portanto, níveis geralmente mais baixos devem ser levados em consideração ao finalizar a arquitetura de armazenamento.

No momento da gravação, as contas de GRS oferecem um destino de largura de banda de 10 Gbps nos EUA para ingresso e 20 Gbps se o RA-GRS/GRS estiver habilitado. Quando se trata de contas de LRS, os limites estão no lado superior, em comparação com o GRS. Para contas de LRS, o ingresso é de 20 Gbps, e a saída é de 30 Gbps. Fora dos EUA, os valores são mais baixos: o destino de largura de banda é 10 Gbps e 5 Gbps para saída. Se houver um requisito de largura de banda maior, você poderá entrar em contato com o suporte do Azure, e eles poderão ajudar com outras opções.

As contas de armazenamento devem ser habilitadas para autenticação usando tokens SAS. Elas não devem permitir o acesso anônimo. Além disso, para o armazenamento de blobs, diferentes contêineres devem ser criados com tokens SAS separados gerados com base nos diferentes tipos e categorias de clientes que acessam esses contêineres. Esses tokens SAS devem ser regenerados periodicamente para garantir que as chaves não estejam em risco de serem adivinhadas ou decifradas. Você saberá mais sobre os tokens SAS e outras opções de segurança no Capítulo 8, Arquitetar aplicações seguras no Azure.

Geralmente, os blobs buscados para contas de armazenamento de blobs devem ser armazenados em cache. Podemos determinar se o cache é obsoleto ao comparar sua última propriedade modificada para buscar novamente o blob mais recente.

Contas de armazenamento oferecem recursos de simultaneidade para garantir que os mesmos arquivos e dados não sejam modificados simultaneamente por vários usuários. Elas incluem:

- **Simultaneidade otimista:** permite que vários usuários modifiquem os dados simultaneamente, mas, durante a gravação, verifica se o arquivo ou os dados foram alterados. Se foram, ela notificará os usuários para que refaçam a busca dos dados e atualizem novamente. Essa é a simultaneidade padrão para tabelas.
- **Simultaneidade pessimista:** quando uma aplicação tenta atualizar um arquivo, ela coloca um bloqueio que nega explicitamente qualquer atualização por outros usuários. Essa é a simultaneidade padrão para arquivos quando acessados usando o protocolo SMB.
- **O último autor ganha:** as atualizações não são restritas, e o último usuário atualiza o arquivo, independentemente do que foi lido inicialmente. Essa é a simultaneidade padrão para filas, blobs e arquivos (quando acessados usando REST).

A essa altura, você deve saber quais são os diferentes serviços de armazenamento e como eles podem ser utilizados em suas soluções. Na próxima seção, analisaremos os padrões de design e veremos como eles se relacionam com projetos arquitetônicos.

## Padrões de design de nuvem

Padrões de design são soluções comprovadas para problemas de design conhecidos. Eles são soluções reutilizáveis que podem ser aplicadas a problemas. Eles não são códigos ou designs reutilizáveis que possam ser incorporados como estão a uma solução. Eles são descrições e orientações documentadas para resolver um problema. Um problema pode se manifestar em contextos diferentes, e os padrões de design podem ajudar a resolvê-lo. O Azure fornece vários serviços, e cada um oferece funcionalidades e recursos específicos. O uso desses serviços é simples, mas criar soluções combinando vários serviços pode ser um desafio. Além disso, obter alta disponibilidade, superescalabilidade, confiabilidade, performance e segurança para uma solução não é uma tarefa simples.

Os padrões de design do Azure fornecem soluções prontas que podem ser adaptadas para problemas individuais. Eles nos ajudam a criar soluções altamente disponíveis, escalonáveis, confiáveis, seguras e centradas em performance no Azure. Embora haja muitos padrões e alguns deles sejam abordados em detalhes nos capítulos subsequentes, alguns dos padrões de mensagens, performance e escalabilidade são mencionados neste capítulo. Além disso, são fornecidos links para descrições detalhadas desses padrões. Esses padrões de design merecem um livro completo dedicado exclusivamente a eles. Eles foram mencionados aqui para que você saiba de sua existência e fornecer referências para mais informações.

## Padrões de mensagens

Os padrões de mensagens ajudam a conectar os serviços por meio de uma associação livre. Isso significa que os serviços nunca interagem diretamente uns com os outros. Em vez disso, um serviço gera e envia uma mensagem para um intermediário (geralmente uma fila), e qualquer outro serviço que tenha interesse nessa mensagem pode selecioná-la e processá-la. Não há comunicação direta entre o serviço remetente e o destinatário. Essa dissociação não só torna os serviços e a aplicação geral mais confiáveis, mas também mais robustos e tolerantes a falhas. Os destinatários podem receber e ler mensagens em sua própria velocidade.

As mensagens ajudam a criar padrões assíncronos. O serviço de mensagens envolve o envio de mensagens de uma entidade para outra. Essas mensagens são criadas e encaminhadas por um remetente, salvas em um armazenamento durável e finalmente consumidas pelos destinatários.

As principais preocupações de arquitetura abordadas pelos padrões de mensagens são:

- **Durabilidade:** as mensagens são salvas em um armazenamento durável, e as aplicações podem lê-las depois que elas são recebidas em caso de failover.
- **Confiabilidade:** as mensagens ajudam a implementar a confiabilidade, pois elas persistem no disco e nunca são perdidas.
- **Disponibilidade das mensagens:** as mensagens estão disponíveis para consumo por aplicações após a restauração da conectividade e antes do tempo de inatividade.

O Azure fornece filas e tópicos de barramento de serviço para implementar padrões de mensagens nas aplicações. O armazenamento de filas do Azure também pode ser usado para a mesma finalidade.

A escolha entre as filas de barramento de serviço e o armazenamento de filas do Azure é para decidir entre por quanto tempo a mensagem deve ser armazenada, o tamanho da mensagem, a latência e o custo. O barramento de serviço do Azure fornece suporte para mensagens de 256 KB, enquanto o armazenamento de filas fornece suporte para mensagens de 64 KB. O barramento de serviço do Azure pode armazenar mensagens por tempo ilimitado, enquanto o armazenamento de filas pode armazenar mensagens por sete dias. O custo e a latência são mais elevados com as filas de barramento de serviço.

Dependendo dos requisitos e das necessidades da aplicação, os fatores precedentes devem ser considerados antes de escolher a melhor fila. Na próxima seção, discutiremos diferentes tipos de padrões de mensagens.

## O padrão de consumidores concorrentes

Um consumidor único de mensagens funciona de forma síncrona, a menos que a aplicação implemente a lógica de leitura de mensagens de forma assíncrona. O padrão de consumidores concorrentes implementa uma solução na qual vários consumidores estão prontos para processar as mensagens recebidas e competem para processar cada mensagem. Isso pode gerar soluções altamente disponíveis e escaláveis. Esse padrão é escalável porque, com vários consumidores, é possível processar um maior número de mensagens em um período menor. É altamente disponível porque deveria haver pelo menos um consumidor para processar as mensagens, mesmo se alguns consumidores falharem.

Esse padrão deve ser usado quando cada mensagem é independente de outras. As mensagens por si só contêm todas as informações necessárias para um consumidor concluir uma tarefa. Esse padrão não deve ser usado quando há dependência entre as mensagens. O consumidor deve ser capaz de concluir as tarefas de forma isolada. Além disso, esse padrão é aplicável quando há demanda variável para os serviços. Mais consumidores podem ser adicionados ou removidos com base na demanda.

Uma fila de mensagens é necessária para implementar o padrão de consumidores concorrentes. Aqui, padrões de várias fontes passam por uma única fila, que está conectada a vários consumidores na outra extremidade. Esses consumidores devem excluir cada mensagem depois de ler para que elas não sejam processadas novamente:



Figura 3.10: o padrão de consumidores concorrentes

Consulte a documentação da Microsoft em <https://docs.microsoft.com/azure/architecture/patterns/competing-consumers> para saber mais sobre esse padrão.

## O padrão de fila de prioridade

Muitas vezes, é necessário priorizar algumas mensagens em detrimento de outras. Esse padrão é importante para aplicações que fornecem diferentes **acordos de nível de serviço (SLAs)** para os consumidores, que fornecem serviços com base em planos e assinaturas diferenciais.

As filas seguem o padrão de ordem de chegada. As mensagens são processadas em sequência. No entanto, com a ajuda do padrão de fila de prioridade, é possível acelerar o processamento de certas mensagens devido à sua maior prioridade. Existem várias maneiras de implementar isso. Se a fila permite atribuir prioridade e reordenar as mensagens com base na prioridade, até mesmo uma única fila é suficiente para implementar esse padrão:

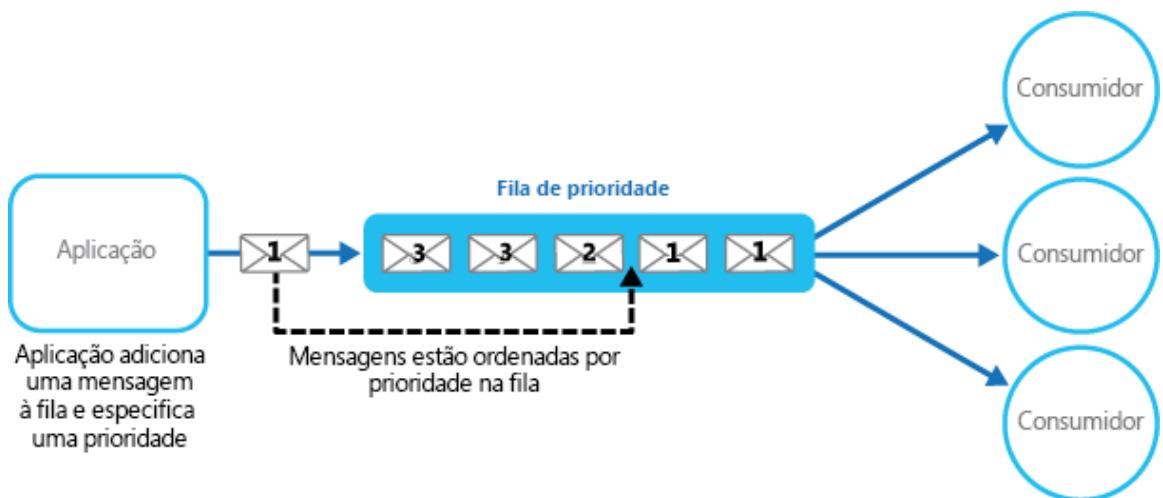


Figura 3.11: o padrão de fila de prioridade única

No entanto, se a fila não puder reordenar as mensagens, filas separadas podem ser criadas para prioridades diferentes, e cada fila pode ter consumidores separados associados a ela:

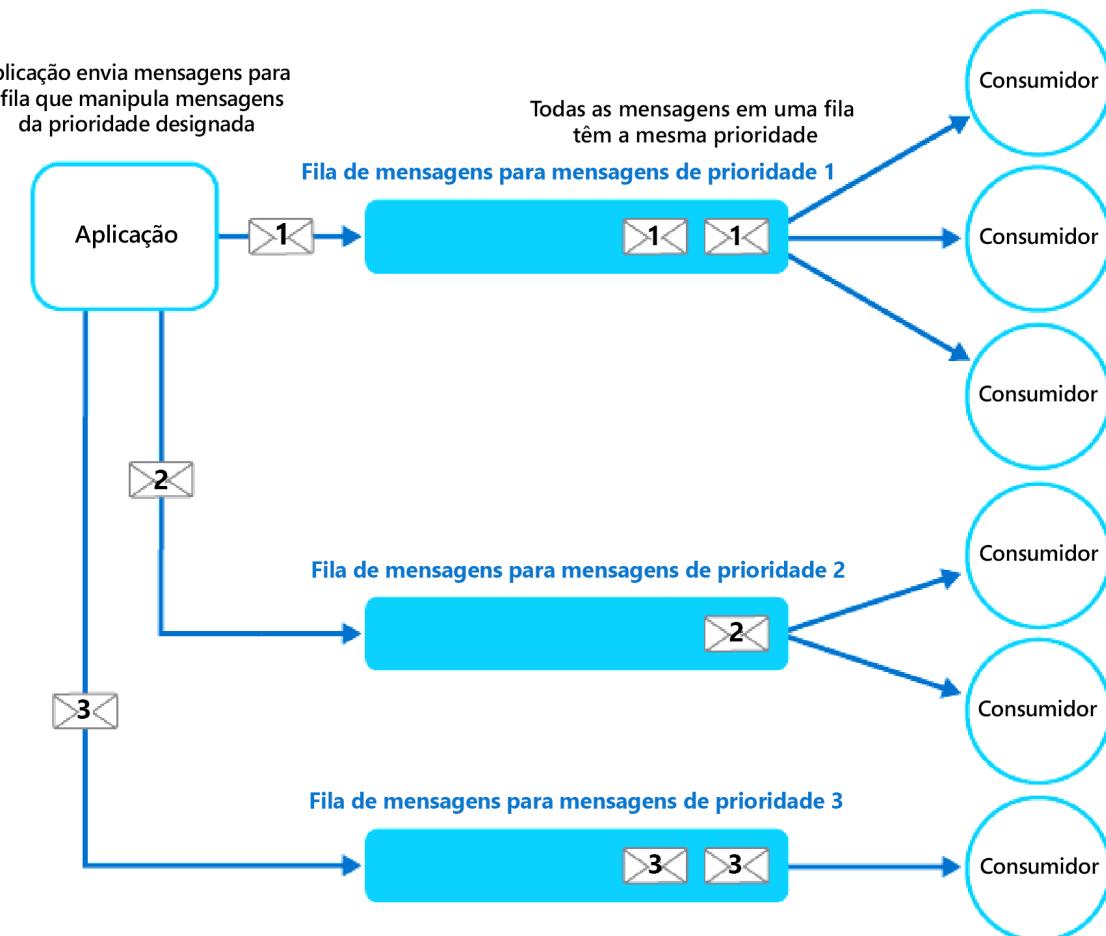


Figura 3.12: usando filas de mensagens separadas para diferentes prioridades

Na verdade, esse padrão pode usar o padrão de consumidores concorrentes para acelerar o processamento de mensagens de cada fila utilizando vários consumidores. Consulte a documentação da Microsoft em <https://docs.microsoft.com/azure/architecture/patterns/priority-queue> para ler mais sobre o padrão de fila de prioridade.

## O padrão de nivelamento de carga baseada em fila

O padrão de nivelamento de carga baseado em fila reduz o impacto dos picos na demanda sobre a disponibilidade e o estado de alerta de tarefas e serviços. Entre uma tarefa e um serviço, uma fila atuará como um buffer. Ela pode ser invocada para lidar com as cargas pesadas inesperadas que podem causar interrupção de serviço ou tempos limite. Esse padrão ajuda a abordar problemas de performance e confiabilidade. Para evitar que o serviço fique sobrecarregado, apresentaremos uma fila que armazenará uma mensagem até que seja recuperada pelo serviço. As mensagens serão retiradas da fila pelo serviço de maneira consistente e processadas.

A Figura 3.13 mostra como funciona o padrão de nivelamento de carga baseado em fila:

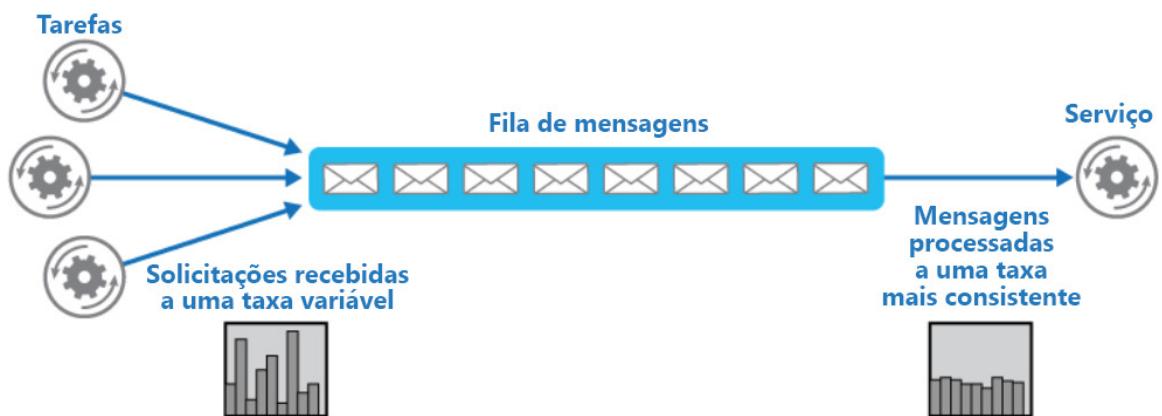


Figura 3.13: o padrão de nivelamento de carga baseada em fila

Embora esse padrão ajude a lidar com picos de demanda inesperada, ele não é a melhor opção quando você está arquitetando um serviço com latência mínima. Falando em latência, que é uma medição de performance, na próxima seção vamos nos concentrar nos padrões de performance e escalabilidade.

## Padrões de performance e escalabilidade

Performance e escalabilidade andam juntas. A performance é a medida da rapidez com que um sistema pode executar uma ação dentro de um intervalo de tempo de forma positiva. Por outro lado, a escalabilidade é a capacidade de um sistema lidar com carga inesperada sem afetar sua performance ou a rapidez com que ele pode ser expandido com os recursos disponíveis. Nesta seção, serão descritos alguns padrões de design relacionados à performance e escalabilidade.

## Padrão CQRS (Segregação de Responsabilidades de Consultas e Comandos)

O CQRS não é um padrão específico do Azure, mas um padrão geral que pode ser aplicado a qualquer aplicação. Isso melhora a performance geral e a capacidade de resposta de uma aplicação.

O CQRS é um padrão que segregá as operações que leem dados (consultas) das operações que atualizam dados (comandos) usando interfaces separadas. Isso significa que os modelos de dados utilizados para consultas e atualizações são diferentes. Os modelos podem, então, ser isolados, conforme mostrado na Figura 3.14, embora isso não seja um requisito absoluto.

Esse padrão deve ser usado quando há regras de negócios grandes e complexas executadas durante a atualização e a recuperação de dados. Além disso, esse padrão tem um caso de uso excelente em que uma equipe de desenvolvedores pode se concentrar no modelo de domínio complexo que faz parte do modelo de gravação, e outra equipe pode se concentrar no modelo de leitura e nas interfaces de usuário. Também é aconselhável usar esse padrão quando a proporção de leitura para gravação é distorcida. A performance das leituras de dados deve ser ajustada separadamente da performance das gravações de dados.

O CQRS não só melhora a performance de uma aplicação, mas também ajuda no design e na implementação de várias equipes. Devido à sua natureza de usar modelos separados, o CQRS não é adequado se você estiver usando ferramentas de geração de modelo e scaffolding:



Figura 3.14: o padrão CQRS

Consulte a documentação da Microsoft em <https://docs.microsoft.com/azure/architecture/patterns/cqrs> para saber mais sobre esse padrão.

## O padrão de fonte de evento

Como a maioria das aplicações trabalha com dados e como os usuários estão trabalhando com eles, a abordagem clássica da aplicação seria manter e atualizar o estado atual dos dados. Ler dados da origem, modificá-los e atualizar o estado atual com o valor modificado é a abordagem típica de processamento de dados. No entanto, há algumas limitações:

- Como as operações de atualização são feitas diretamente no armazenamento de dados, isso atrasa a performance geral e a capacidade de resposta.
- Se houver diversos usuários trabalhando e atualizando os dados, poderá haver conflitos, e algumas das atualizações relevantes poderão falhar.

A solução para isso é implementar o padrão de fornecimento de eventos, onde as alterações serão gravadas em um armazenamento somente acréscimo. Uma série de eventos será acionada pelo código da aplicação para o armazenamento de eventos, onde eles serão persistidos. Os eventos persistentes em um armazenamento de eventos atuam como um sistema de registro sobre o estado atual dos dados. Os consumidores serão notificados e poderão lidar com os eventos, se necessário, depois que eles forem publicados.

O padrão de fornecimento de eventos é mostrado na Figura 3.15:

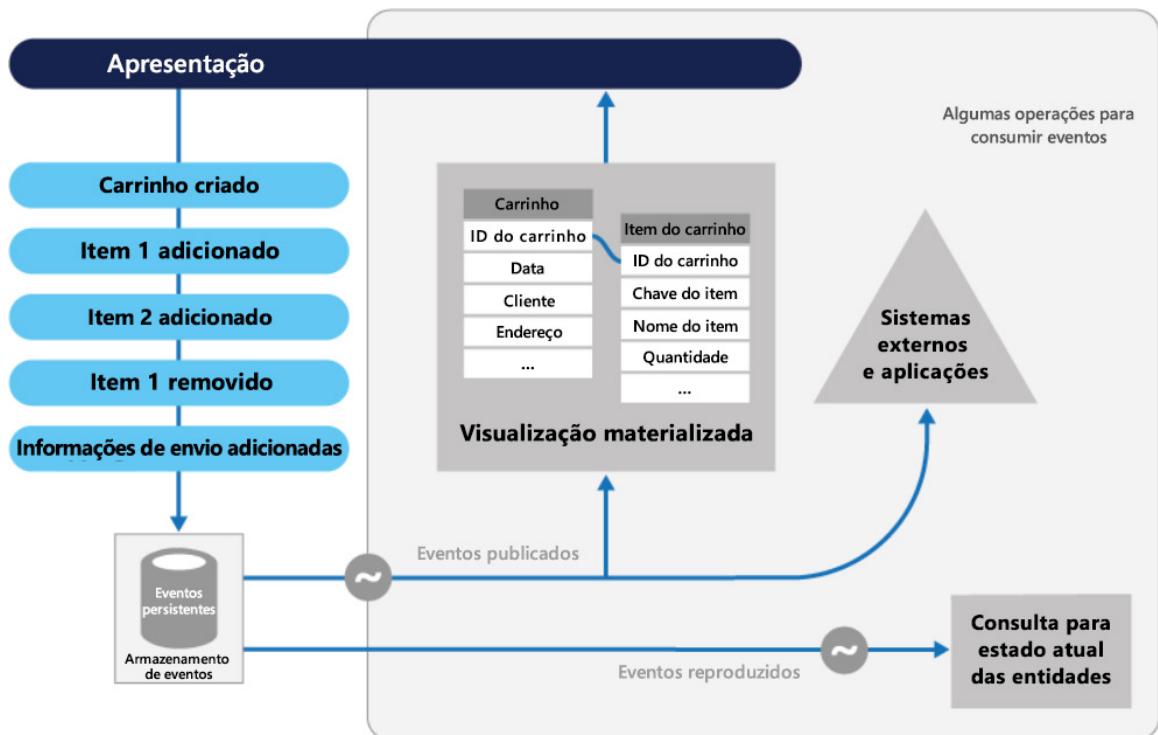


Figura 3.15: o padrão de fonte de evento

Mais informações sobre esse padrão estão disponíveis em <https://docs.microsoft.com/azure/architecture/patterns/event-sourcing>.

## O padrão de limitação

Às vezes, há aplicações com requisitos de SLA muito rigorosos em termos de performance e escalabilidade, independentemente do número de usuários que consomem o serviço. Nessas circunstâncias, é importante implementar o padrão de limitação porque ele pode limitar o número de solicitações que podem ser executadas. A carga sobre as aplicações não pode ser prevista com precisão em todas as circunstâncias. Quando a carga sobre uma aplicação aumenta, a limitação reduz a pressão sobre os servidores e serviços, controlando o consumo de recursos. A infraestrutura do Azure é um bom exemplo desse padrão.

Esse padrão deve ser usado quando o cumprimento do SLA é uma prioridade para as aplicações, para impedir que alguns usuários consumam mais recursos do que o alocado, otimizar picos e explosões de demanda e melhorar o custo do consumo de recursos. Esses são cenários válidos para aplicações criadas para serem implantadas na nuvem.

Pode haver várias estratégias para gerenciar a limitação em uma aplicação. A estratégia de limitação pode rejeitar novas solicitações depois que o limite é ultrapassado ou pode avisar ao usuário que a solicitação está na fila e terá a oportunidade de ser executada quando o número de solicitações for reduzido.

A Figura 3.16 ilustra a implementação do padrão de limitação em um sistema multilocatário, onde cada locatário é atribuído a um limite fixo de uso de recursos. Quando esse limite for ultrapassado, qualquer demanda adicional de recursos será restrita, mantendo assim recursos suficientes para outros locatários:

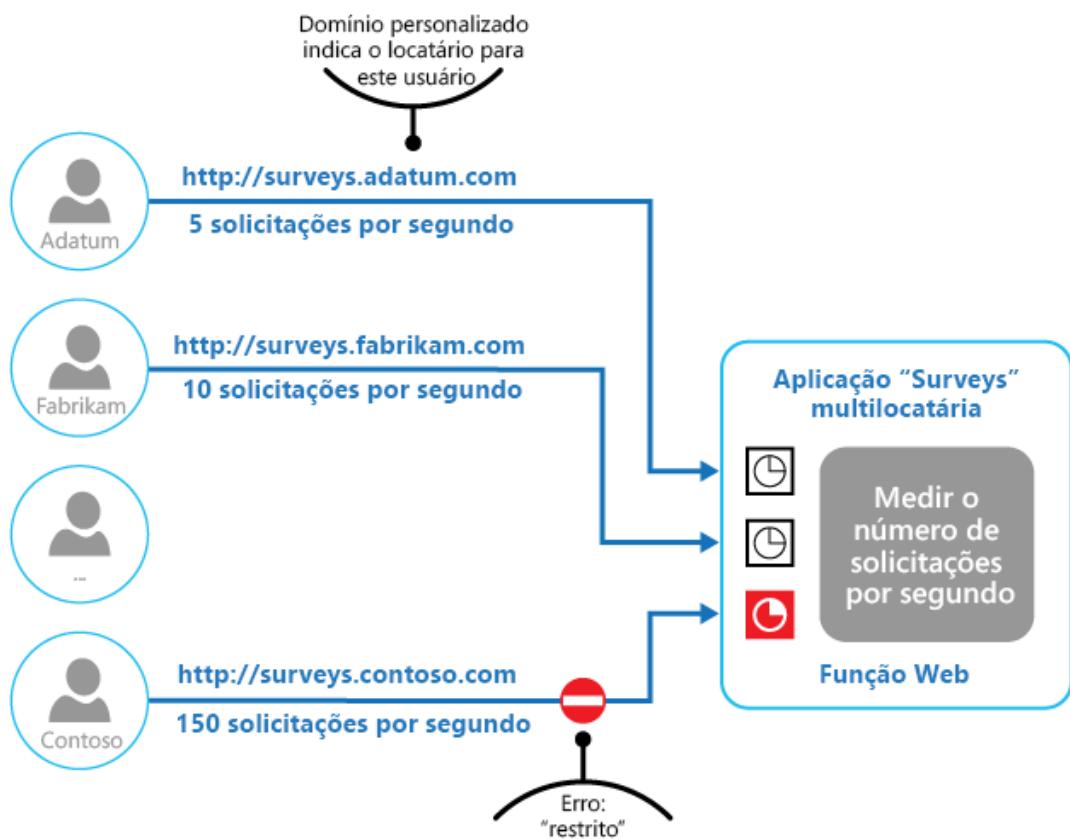


Figura 3.16: o padrão de limitação

Leia mais sobre esse padrão em <https://docs.microsoft.com/azure/architecture/patterns/throttling>.

## Padrão de repetição

O padrão de repetição é um padrão extremamente importante que torna aplicações e serviços mais resilientes a falhas temporárias. Imagine que você está tentando se conectar e usar um serviço, e ele não está disponível por algum motivo. Se o serviço for disponibilizado em breve, convém continuar tentando estabelecer uma conexão bem-sucedida. Isso tornará a aplicação mais robusta, tolerante a falhas e estável. No Azure, a maioria dos componentes funciona na internet, e a conexão de internet pode gerar falhas temporárias间断地. Como essas falhas podem ser corrigidas em poucos segundos, a falha de uma aplicação não deve ser permitida. A aplicação deve ser projetada de modo a poder tentar usar o serviço novamente várias vezes em caso de falha e parar de tentar quando houver êxito. Ela também pode determinar que há uma falha que levará tempo para ser corrigida.

Esse padrão deve ser implementado quando uma aplicação correr o risco de falhas temporárias quando interagir com um serviço remoto ou acessar um recurso remoto. Espera-se que essas falhas sejam passageiras e que a repetição de uma solicitação que falhou anteriormente possa ter êxito em uma tentativa subsequente.

O padrão de repetição pode adotar estratégias de repetição diferentes com base na natureza dos erros e da aplicação:

- **Repetir um número fixo de vezes:** isso indica que a aplicação tentará se comunicar com o serviço por um número fixo de vezes antes de determinar que houve uma falha e gerar uma exceção. Por exemplo, repetir três vezes a tentativa de conexão com outro serviço. Se a conexão for bem-sucedida dentro dessas três tentativas, toda a operação será bem-sucedida. Caso contrário, será gerada uma exceção.
- **Repetir com base na agenda:** indica que a aplicação tentará se comunicar com o serviço repetidamente por um número fixo de segundos ou minutos e aguardará um número fixo de segundos ou minutos antes de tentar novamente. Por exemplo, a aplicação tentará se conectar ao serviço a cada 3 segundos por 60 segundos. Se houver êxito na conexão dentro desse tempo, toda a operação será bem-sucedida. Caso contrário, será gerada uma exceção.
- **Postergar e atrasar a repetição:** indica que a aplicação tentará se comunicar com o serviço repetidamente com base no cronograma e continuará adicionando um atraso incremental nas tentativas subsequentes. Por exemplo, por um total de 60 segundos, a primeira nova tentativa ocorre após um segundo, a segunda ocorre dois segundos após a anterior, a terceira ocorre quatro segundos após a tentativa anterior e assim por diante. Isso reduz o número total de repetições.

A Figura 3.17 ilustra o padrão de repetição. A primeira solicitação recebe uma resposta HTTP 500, a segunda tentativa recebe novamente uma resposta HTTP 500 e, por fim, a solicitação é bem-sucedida e recebe HTTP 200 como a resposta:

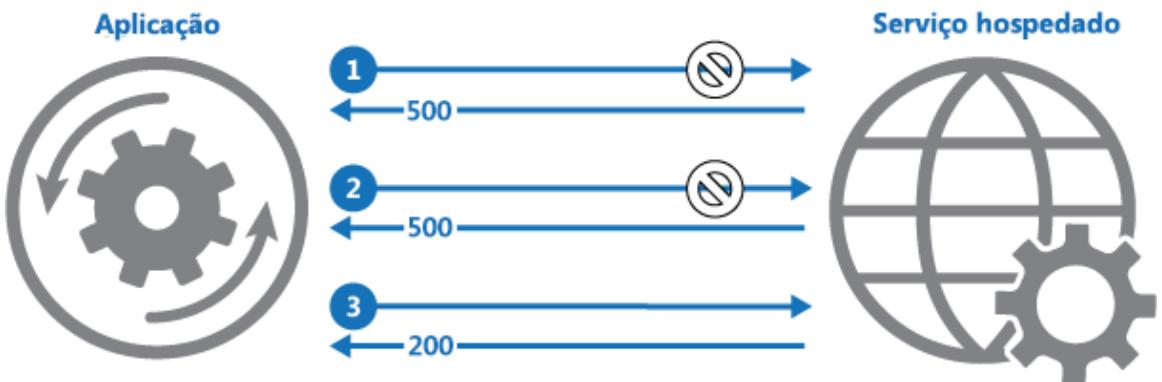


Figura 3.17: o padrão de repetição

Consulte esta documentação da Microsoft em <https://docs.microsoft.com/azure/architecture/patterns/retry> para saber mais sobre esse padrão.

### O padrão de disjuntor de circuito

Esse é um padrão extremamente útil. Imagine novamente que você está tentando se conectar e usar um serviço, e ele não está disponível por algum motivo. Se o serviço não for disponibilizado em breve, não adianta continuar tentando estabelecer a conexão. Além disso, manter outros recursos ocupados durante as tentativas desperdiça muitos recursos que poderiam ser usados em outro lugar.

O padrão de disjuntor de circuito ajuda a eliminar esse desperdício de recursos. Ele pode impedir que as aplicações tentem conectar-se repetidamente e usar um serviço que não esteja disponível. Ele também ajuda as aplicações a detectar se um serviço está ativo e funcionando novamente e permite que as aplicações se conectem a ele.

Para implementar o padrão de disjuntor de circuito, todas as solicitações para o serviço devem passar por um serviço que atua como um proxy para o serviço original. A finalidade desse serviço de proxy é manter uma máquina de estado e atuar como gateway para o serviço original. Esse serviço mantém três estados. Pode haver mais estados incluídos, dependendo dos requisitos da aplicação.

Os estados mínimos necessários para implementar esse padrão são:

- **Aberto:** indica que o serviço está inativo, e a aplicação é mostrada como uma exceção imediatamente, em vez de permitir que ele tente novamente ou espere um tempo limite. Quando o serviço está ativo novamente, o estado muda para Semi-aberto.
- **Fechado:** esse estado indica que o serviço é íntegro e que a aplicação pode se conectar a ele. Geralmente, um contador mostra o número de falhas antes de poder mudar para o estado Aberto.
- **Semiaberto:** em algum momento, quando o serviço estiver funcionando, esse estado permite que um número limitado de solicitações passe por ele. Esse estado é uma prova de fogo que verifica se as solicitações que passam são bem-sucedidas. Se as solicitações forem bem-sucedidas, o estado mudará de Semiaberto para Fechado. Esse estado também pode implementar um contador para permitir que um certo número de solicitações sejam bem-sucedidas antes de mudar para o estado Fechado.

Os três estados e suas transições são ilustrados na Figura 3.18:

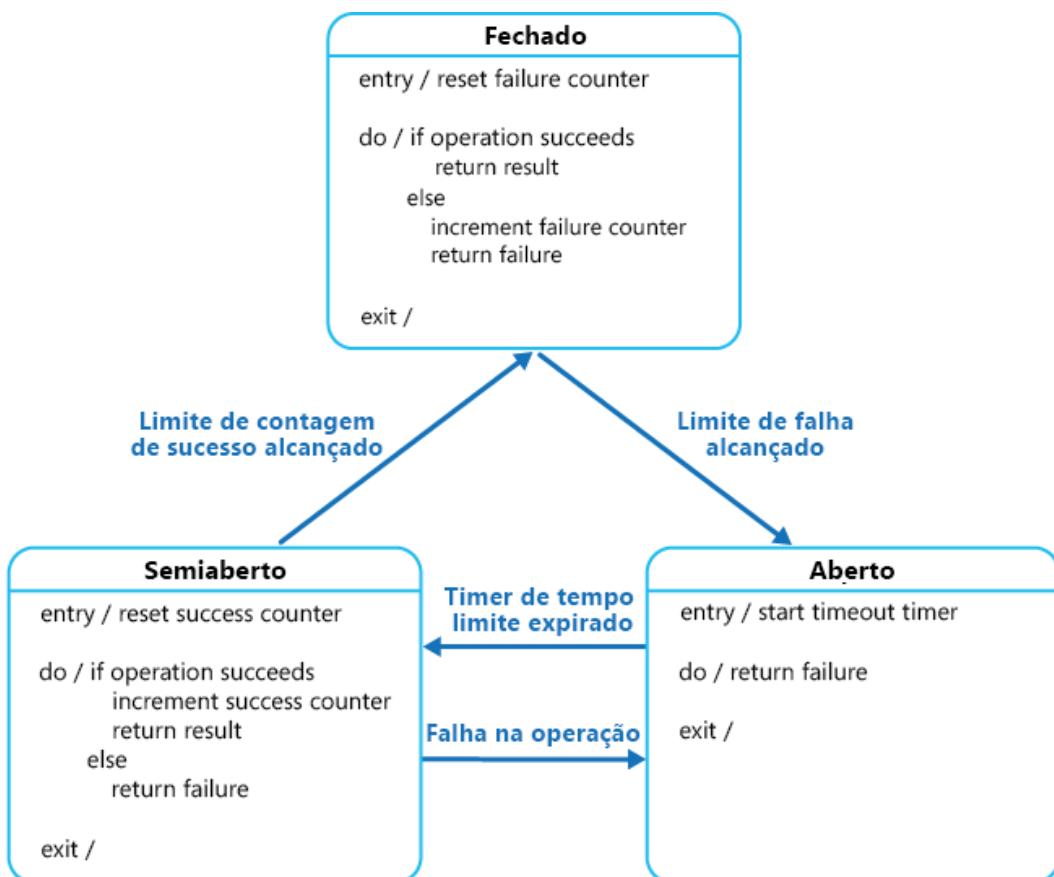


Figura 3.18: o padrão de disjuntor de circuito

Leia mais sobre esse padrão na documentação da Microsoft em <https://docs.microsoft.com/azure/architecture/patterns/circuit-breaker>.

Nesta seção, discutimos padrões de design que podem ser usados para arquitetar aplicações confiáveis, escaláveis e seguras na nuvem. No entanto, há outros padrões que você pode explorar em <https://docs.microsoft.com/azure/architecture/patterns>.

## Resumo

Há vários serviços disponíveis no Azure, e a maioria pode ser combinada para criar soluções reais. Este capítulo explicou os três serviços mais importantes fornecidos pelo Azure: regiões, armazenamento e redes. Eles formam a base da maioria das soluções implantadas em qualquer nuvem. Este capítulo forneceu detalhes sobre esses serviços e como a configuração e o provisionamento deles pode afetar as decisões de design.

Considerações importantes para armazenamento e redes foram detalhadas neste capítulo. As redes e o armazenamento fornecem muitas opções, e é importante escolher uma configuração adequada com base nos seus requisitos.

Por fim, alguns dos padrões de design importantes relacionados a mensagens, como consumidores concorrentes, fila de prioridade e nivelamento de carga, foram descritos. Padrões como CQRS e limitação foram ilustrados, e outros padrões, como repetição e disjuntor de circuito, também foram discutidos. Manteremos esses padrões como a linha de base quando implantarmos nossas soluções.

No próximo capítulo, discutiremos como automatizar as soluções que vamos arquitetar. À medida que avançamos no mundo da automação, todas as organizações querem eliminar a sobrecarga da criação de recursos um por um, algo muito exigente. Como a automação é a solução para isso, no próximo capítulo você aprenderá mais sobre ela.



# 4

## Automatizar a arquitetura no Azure

Todas as organizações desejam reduzir o esforço manual e o erro em suas atividades, e a automação desempenha um papel importante para trazer previsibilidade, padronização e consistência para a criação de um produto e as operações. A automação tem sido o foco de quase todos os **diretores de informações (CIO)** e diretores digitais para garantir que seus sistemas sejam altamente disponíveis, escaláveis, confiáveis e capazes de atender às necessidades de seus clientes.

A automação se tornou mais proeminente com o advento da nuvem porque novos recursos puderam ser provisionados rapidamente, sem a aquisição de recursos de hardware. Portanto, as empresas de nuvem desejam a automação em quase todas as suas atividades para reduzir o uso indevido, os erros, a governança, a manutenção e a administração.

Neste capítulo, avaliaremos a Automação do Azure como um serviço importante que fornece recursos de automação, juntamente com seus recursos que são diferenciais em comparação com outros serviços aparentemente semelhantes. Neste capítulo, vamos abordar o seguinte:

- O cenário da Automação do Azure
- O serviço Automação do Azure
- Recursos para serviços da Automação do Azure
- A escrita de runbooks da Automação do Azure
- Webhooks
- Hybrid Workers

Vamos começar com a Automação do Azure, um serviço de nuvem para automação de processos.

## Automação

A automação é necessária para o provisionamento, as operações, o gerenciamento e o desprovisionamento de recursos de TI em uma organização. A Figura 4.1 mostra uma visão mais detalhada sobre o que cada um desses casos de uso representa:

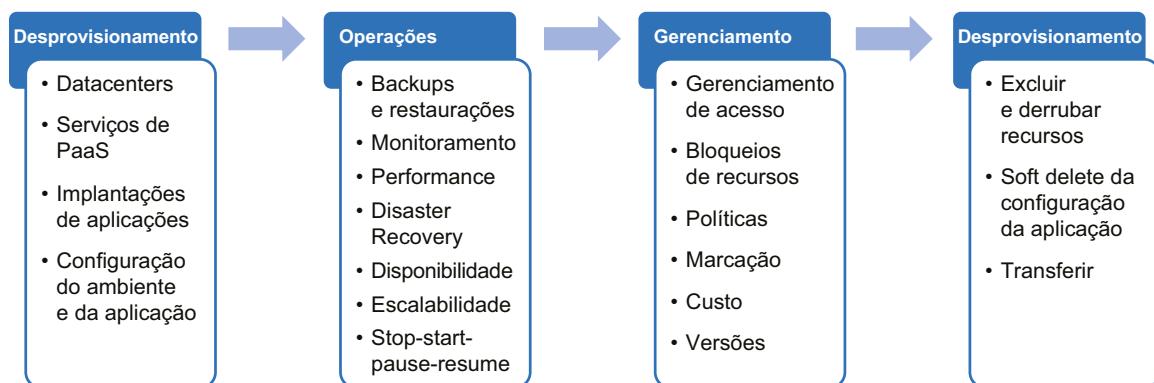


Figura 4.1: casos de uso de automação

Antes do advento da nuvem, os recursos de TI eram principalmente na infraestrutura local, e os processos manuais eram frequentemente usados para essas atividades. No entanto, como a adoção da nuvem aumentou, a automação recebeu maior foco e atenção. O principal motivo é que a agilidade e a flexibilidade da tecnologia de nuvem proporcionam uma oportunidade para provisionar, desprovisionar e gerenciar esses recursos rapidamente em uma pequena fração do tempo que costumava levar. Juntamente com essa flexibilidade e agilidade, os requisitos são mais previsíveis e consistentes com a nuvem, pois ela facilitou que as organizações criem recursos.

A Microsoft tem uma ótima ferramenta para automação de TI conhecida como System Center Orchestrator. É uma ótima ferramenta para automação para ambientes na infraestrutura local e na nuvem, mas é um produto, e não um serviço. Ele deve ser licenciado e implantado em servidores, e, em seguida, os runbooks podem ser executados para fazer alterações em ambientes na infraestrutura local e na nuvem.

A Microsoft percebeu que era necessária uma solução de automação que pudesse ser disponibilizada aos clientes como um serviço, em vez de ser comprada e implantada como um produto. A Automação do Azure entra em ação.

## Automação do Azure

O Azure fornece um serviço chamado **Automação do Azure**, que é essencial para a automação de processos, atividades e trabalhos, não só no Azure, como também na infraestrutura local. Usando a Automação do Azure, as organizações podem automatizar seus processos e trabalhos relacionados ao processamento, à destruição, às operações e aos gerenciamento dos recursos na nuvem, em ambientes de TI, em plataformas e em linguagens. Na Figura 4.2, podemos ver alguns recursos da Automação do Azure:

Entre nuvens	Entre ambientes	Entre plataformas	Entre linguagens
<ul style="list-style-type: none"> <li>• Azure</li> <li>• Outras nuvens</li> <li>• Qualquer combinação</li> </ul>	<ul style="list-style-type: none"> <li>• Nuvem</li> <li>• Na infraestrutura local</li> <li>• Híbrida</li> </ul>	<ul style="list-style-type: none"> <li>• Linux</li> <li>• Windows</li> </ul>	<ul style="list-style-type: none"> <li>• PowerShell</li> <li>• Python</li> <li>• Bash</li> </ul>

Figura 4.2: recursos da Automação do Azure

## Arquitetura da Automação do Azure

A Automação do Azure consiste em vários componentes, e cada um deles é completamente dissociado dos outros. A maior parte da integração acontece no nível do armazenamento de dados, e nenhum componente se comunica com o outro diretamente.

Quando uma conta da Automação é criada no Azure, ela é gerenciada por um serviço de gerenciamento. O serviço de gerenciamento é um único ponto de contato para todas as atividades na Automação do Azure. Todas as solicitações no portal, incluindo salvar, publicar e criar runbooks, para executar, parar, suspender, iniciar e testar são enviadas para o serviço de gerenciamento de automação, e o serviço grava os dados de solicitação no armazenamento de dados. Ele também cria um registro de trabalho no armazenamento de dados e, com base no status dos trabalhadores do runbook, o atribui a um trabalhador.

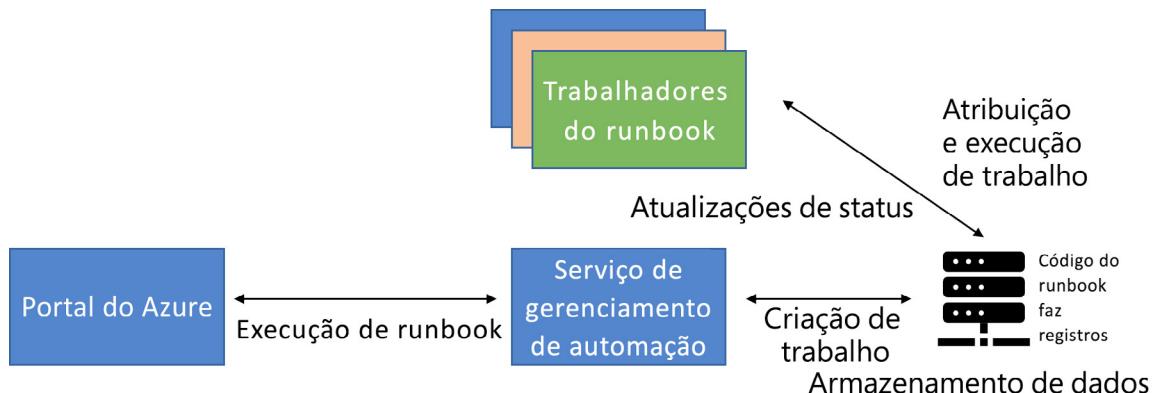


Figura 4.3: arquitetura da Automação do Azure

O trabalhador continua pesquisando o banco de dados para obter novos trabalhos atribuídos a ele. Depois que encontra uma atribuição de trabalho, ele obtém as informações do trabalho e começa a executar o trabalho usando seu mecanismo de execução. Os resultados são gravados novamente no banco de dados, lidos pelo serviço de gerenciamento e exibidos novamente no portal do Azure.

Os Hybrid Workers sobre os quais leremos mais adiante neste capítulo também são trabalhadores de runbook, embora não sejam mostrados na Figura 4.3.

A primeira etapa para começar a usar a Automação do Azure é criar uma nova conta. Depois que a conta for criada, todos os outros artefatos serão criados na conta.

A conta funciona como o principal recurso de nível superior que pode ser gerenciado usando grupos de recursos do Azure e seu próprio plano de controle.

A conta deve ser criada em uma região, e toda a automação nessa conta é executada em servidores nessa região.

É importante escolher a região com cuidado, preferencialmente perto de outros recursos do Azure que a conta da Automação integra ou gerencia, para reduzir o tráfego de rede e a latência entre as regiões.

A conta da Automação também oferece suporte a algumas contas de **execução**, que podem ser criadas com a conta da Automação. Como essas contas de execução são análogas a uma conta de serviço, as criamos principalmente para executar ações. Embora nos referimos a ela como conta de execução em geral, há dois tipos de conta de execução: um é chamado de conta de execução clássica, e o outro é simplesmente conta de execução, sendo que ambos são usados para conectar a assinaturas do Azure. A conta de execução clássica do Azure destina-se à conexão ao Azure usando a API de **Gerenciamento de Recursos do Azure**, e a conta de execução destina-se à conexão ao Azure usando a API de **Gerenciamento de Recursos do Azure (ARM)**.

As duas contas usam certificados para autenticação com o Azure. Essas contas podem ser criadas durante a criação da conta da Automação, ou você pode optar por criá-las em um estágio posterior no portal do Azure.

É recomendável criar essas contas de execução mais tarde, em vez de criá-las durante a criação da conta da Automação porque, se elas forem criadas durante a configuração da conta da Automação, a Automação gerará os certificados e as entidades de serviço nos bastidores com a configuração padrão. Se for necessário mais controle e configuração personalizada para essas contas de execução, como o uso de um certificado existente ou da entidade de serviço, as contas de execução devem ser criadas após a conta da Automação.

Depois que a conta da Automação for criada, ela fornecerá um painel por meio do qual vários cenários de automação podem ser habilitados.

Alguns dos cenários importantes que podem ser habilitados usando uma conta da Automação estão relacionados a:

- Automação de processos
- Gerenciamento de configuração
- Gerenciamento de atualizações

A automação envolve a escrita de scripts que são reutilizáveis e genéricos para que possam ser reutilizados em vários cenários. Por exemplo, um script de automação deve ser genérico o suficiente para iniciar e interromper qualquer VM em qualquer grupo de recursos em qualquer grupo de assinatura e gerenciamento. As informações de servidor de VM de codificação, juntamente com nomes de grupo de recursos, assinatura e grupo de gerenciamento, resultarão na criação de vários scripts semelhantes, e qualquer alteração em um resultará na alteração de todos os scripts. É melhor criar um único script para essa finalidade usando parâmetros e variáveis de script, e você deve garantir que os valores sejam fornecidos pelo executor desses artefatos.

Vamos analisar em detalhes cada um dos cenários mencionados anteriormente.

## Automação de processos

A automação de processos refere-se ao desenvolvimento de scripts que refletem processos do mundo real. A automação de processos consiste em várias atividades, em que cada atividade executa um trabalho discreto. Juntas, essas atividades formam um processo completo. As atividades podem ser executadas com base em se a atividade anterior foi executada com êxito ou não.

Há alguns requisitos que qualquer automação de processo exige da infraestrutura em que ela é executada. Alguns deles são os seguintes:

- A capacidade de criar fluxos de trabalho
- A capacidade de executar por longo período
- A capacidade de salvar o estado de execução quando o fluxo de trabalho não estiver concluído, que também é conhecido como definição do ponto de verificação e hidratação
- A capacidade de retomar a partir do último estado salvo, em vez de começar do início

O próximo cenário que vamos explorar é o gerenciamento de configuração.

## Gerenciamento de configuração

O gerenciamento de configuração refere-se ao processo de gerenciamento da configuração do sistema ao longo de seu ciclo de vida. A Configuração de Estado da Automação do Azure é o serviço de gerenciamento de configuração do Azure que permite que os usuários gravem, gerenciem e compilem a configuração do PowerShell DSC para nós de nuvem e datacenters na infraestrutura local.

A Configuração de Estado da Automação do Azure permite gerenciar VMs do Azure, VMs clássicas do Azure e máquinas físicas ou VMs (Windows/Linux) na infraestrutura local, além de fornecer suporte para VMs em outros provedores de nuvem.

Uma das maiores vantagens da Configuração de Estado da Automação do Azure é que oferece escalabilidade. Podemos gerenciar milhares de máquinas em uma única interface de gerenciamento central. Podemos atribuir configurações a máquinas com facilidade e verificar se elas estão em conformidade com a configuração desejada.

Outra vantagem é que a Automação do Azure pode ser usada como um repositório para armazenar as configurações do **Desired State Configuration (DSC)** e, quando necessário, elas podem ser usadas.

Na próxima seção, vamos falar sobre o gerenciamento de atualizações.

## Gerenciamento de atualizações

Como você já sabe, o gerenciamento de atualizações é de responsabilidade do cliente para gerenciar atualizações e patches quando se trata de IaaS. O recurso de gerenciamento de atualizações da Automação do Azure pode ser usado para automatizar ou gerenciar atualizações e patches para suas VMs do Azure. Há vários métodos pelos quais você pode habilitar o gerenciamento de atualizações em sua VM do Azure:

- Na sua conta da Automação
- Navegando no portal do Azure
- Em um runbook
- Em uma VM do Azure

Habilitá-lo em uma VM do Azure é o método mais fácil. No entanto, se você tiver muitas VMs e precisar habilitar o gerenciamento de atualizações, terá que considerar uma solução escalável, como um runbook ou em uma conta de automação.

Agora que os cenários estão claros, vamos explorar os conceitos relacionados à Automação do Azure.

## Conceitos relacionados à Automação do Azure

Agora você sabe que a Automação do Azure exige uma conta, que é chamada de conta da Automação do Azure. Antes de nos aprofundarmos, vamos examinar os conceitos relacionados à Automação do Azure. Entender o significado de cada um desses termos é muito importante, pois vamos usá-los ao longo deste capítulo. Vamos começar com o runbook.

### Runbook

Um runbook da Automação do Azure é uma coleção de instruções de script que representam uma única etapa na automação de processos ou uma automação de processos completa. É possível invocar outros runbooks de um runbook pai, e esses runbooks podem ser criados em várias linguagens de script. As linguagens que oferecem suporte à criação de runbooks são as seguintes:

- PowerShell
- Python 2 (no momento da escrita)
- Fluxos de trabalho do PowerShell
- PowerShell Gráfico
- Fluxos de trabalho do PowerShell Gráfico

A criação uma conta da Automação é muito fácil e pode ser feita no portal do Azure. Na folha **Todos os serviços**, você pode encontrar **Conta da Automação** ou pesquisá-la no portal do Azure. Como mencionado anteriormente, durante a criação, você terá a opção de criar uma conta de execução. A Figura 4.4 mostra as entradas necessárias para criar uma conta da Automação:

**Add Automation Account**

Name \*  ✓

Subscription \*  ✓

Resource group \*  ✓  
[Create new](#)

Location \*  ✓

Create Azure Run As account \* Yes No

! You have chosen not to create a Run As Account. Doing so might block the execution of some runbooks due to lack of access to required resources. [Click here to learn more about Run As accounts.](#)

Figura 4.4: criando uma conta da Automação

## Contas de execução

Por padrão, as contas da Automação do Azure, não têm acesso a nenhum recurso incluído em nenhuma assinatura do Azure, incluindo a assinatura em que estão hospedados. Uma conta precisa de acesso a uma assinatura do Azure e seus recursos a fim de gerenciá-las. Uma conta de execução é uma maneira de fornecer acesso a assinaturas e os recursos nelas.

Esse é um exercício opcional. Pode haver no máximo uma conta de execução para cada assinatura baseada no gerenciador de recursos e clássica. No entanto, uma conta da Automação pode precisar se conectar a várias assinaturas. Nesses casos, é recomendável criar recursos compartilhados para cada uma das assinaturas e usá-los em runbooks.

Depois de criar a conta da Automação, navegue até a exibição **Contas de execução** no portal e você verá que dois tipos de contas podem ser criados. Na Figura 4.5, você pode ver que a opção de criar uma **Conta de execução do Azure** e uma **Conta de execução clássica do Azure** está disponível na folha **Contas de execução**:

Figura 4.5: opções de conta de execução do Azure

Essas contas de execução podem ser criadas usando o portal do Azure, o PowerShell e a CLI. Para obter informações sobre como criar essas contas usando o PowerShell, acesse <https://docs.microsoft.com/azure/automation/manage-runas-account>.

No caso da conta de execução do ARM, este script cria uma nova entidade de serviço do Azure AD e um novo certificado e fornece permissões de RBAC de colaborador para a entidade de serviço recém-criada na assinatura.

## Trabalhos

O envio de uma solicitação de trabalho não está vinculado diretamente à execução da solicitação de trabalho devido à arquitetura dissociada da Automação do Azure. A vinculação entre eles é indireta usando um armazenamento de dados. Quando uma solicitação para executar um runbook é recebida pela Automação, ela cria um novo registro em seu banco de dados com todas as informações relevantes. Há outro serviço em execução em vários servidores, conhecido como Hybrid Runbook Worker, no Azure, que procura novas entradas adicionadas ao banco de dados para a execução de um runbook. Depois que vê um novo registro, ele bloqueia o registro para que nenhum outro serviço possa lê-lo e, em seguida, executa o runbook.

## Ativos

Os ativos da Automação do Azure referem-se a artefatos compartilhados que podem ser usados em runbooks. Eles são mostrados na *Figura 4.6*:

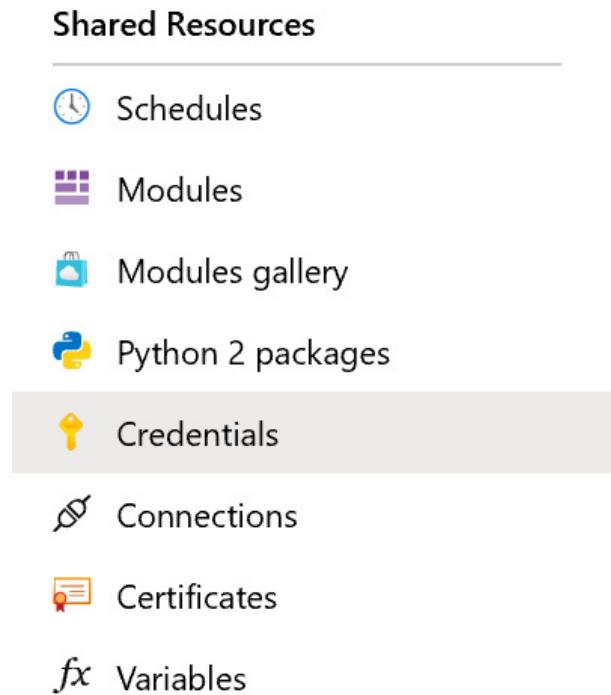


Figura 4.6: artefatos compartilhados na Automação do Azure

## Credenciais

As credenciais se referem aos segredos, como a combinação de nome de usuário/senha, que pode ser usados para se conectar a outros serviços de integração que precisam de autenticação. Essas credenciais podem ser usadas em runbooks usando o cmdlet **Get-AutomationPSCredential** do PowerShell, juntamente com o nome associado:

```
$myCredential = Get-AutomationPSCredential -Name 'MyCredential'
```

A sintaxe Python exige a importação do módulo **automationassets** e o uso da função **get\_automation\_credential** juntamente com o nome de credenciais associado:

```
import automationassets
cred = automationassets.get_automation_credential("credtest")
```

## Certificados

Os certificados referem-se ao certificado X.509 que pode ser adquirido das autoridades de certificação ou pode ser autoassinado. Os certificados são usados para fins de identificação na Automação do Azure. Cada certificado tem um par de chaves conhecidas como chaves privadas/públicas. A chave privada é usada para criar um ativo de certificado na Automação do Azure, e a chave pública deve estar disponível no serviço de destino. Usando a chave privada, a conta da Automação pode criar uma assinatura digital e anexá-la à solicitação antes de enviá-la para o serviço de destino. O serviço de destino pode obter os detalhes (o hash) da assinatura digital usando a chave pública já disponível e verificar a identidade do remetente da solicitação.

Os ativos de certificado armazenam informações de certificado e chaves na Automação do Azure. Esses certificados podem ser usados diretamente nos runbooks e também são usados pelos ativos da conexão. A próxima seção mostra a forma de consumir certificados em um ativo de conexão. O recurso de conexão principal do serviço do Azure usa uma impressão digital do certificado para identificar o certificado que deseja usar, enquanto outros tipos de conexão usam o nome do ativo de certificado para acessar o certificado.

Um ativo de certificado pode ser criado fornecendo um nome e fazendo upload de um certificado. É possível fazer upload de certificados públicos (arquivos .cer), bem como certificados privados (arquivos .pfx). A parte privada do certificado também tem uma senha que deve ser usada antes de acessar o certificado.

The screenshot shows the Azure portal interface for managing certificates in an automation account. On the left, the 'Certificates' blade is open under the 'bookaccount' automation account. It displays a search bar, a 'Refresh' button, and a table showing 'No certificates found.' On the right, a modal dialog titled 'Add a certificate' is displayed, prompting for a certificate name ('azureforarchitectsCertificate'), a description ('Certificate for Azure Automation Account'), a file to upload ('"azureforarchitects.pfx"'), and a password. A 'Create' button is at the bottom of the dialog.

Figura 4.7: adicionando um certificado à Automação do Azure

A criação de um certificado envolve fornecer um nome e uma descrição, fazer upload do certificado, fornecer uma senha (no caso de arquivos .pfx) e informar ao usuário se o certificado é exportável ou não.

Deve haver um certificado disponível para que esse ativo de certificado possa ser criado. Os certificados podem ser comprados de autoridades de certificação ou podem ser gerados. Os certificados gerados são conhecidos como certificados autoassinados. É sempre recomendável usar certificados de autoridades de certificação para ambientes importantes, como ambientes de produção. É bom usar certificados de autoassinatura para fins de desenvolvimento.

Para gerar um certificado autoassinado usando o PowerShell, use este comando:

```
$cert = New-SelfSignedCertificate -CertStoreLocation "Cert:\CurrentUser\my" -KeySpec KeyExchange -Subject "cn=azureforarchitects"
```

Isso criará um novo certificado no armazenamento de certificados do usuário atual em sua pasta pessoal. Como esse certificado também precisa ser carregado no ativo de certificado da Automação do Azure, ele deve ser exportado para o sistema de arquivos local, conforme mostrado na Figura 4.8:

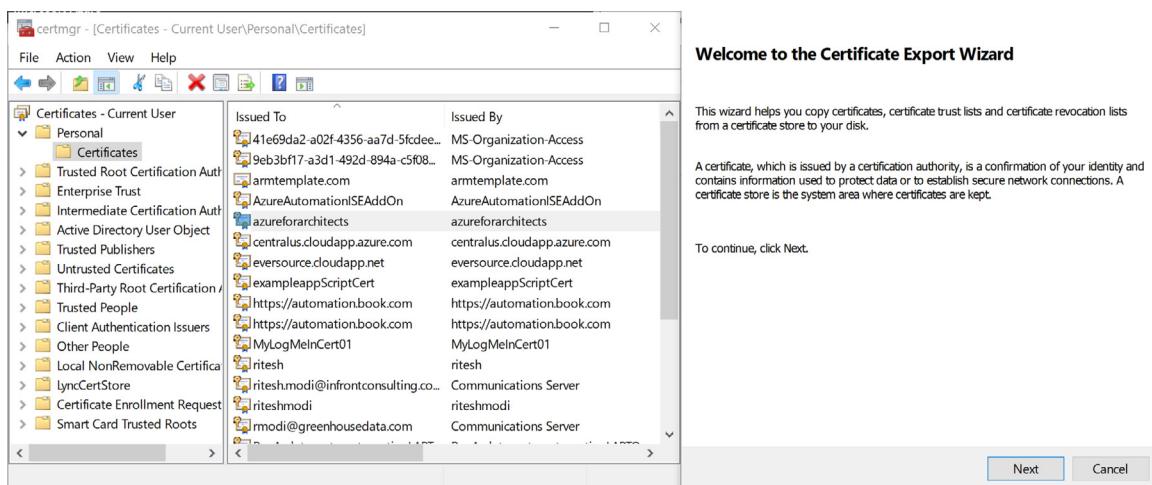


Figura 4.8: exportando o certificado

Ao exportar o certificado, a chave privada também deve ser exportada, portanto, a opção **Sim, exportar a chave privada** deve ser selecionada.

Selecione a opção **Troca de Informações Pessoais**, e o restante dos valores devem permanecer como os padrões.

Forneça uma senha e o nome do arquivo **C:\azureforarchitects.pfx**, e a exportação deve ser bem-sucedida.

A conexão com o Azure pode ser feita de várias maneiras. No entanto, a mais segura é por meio de um certificado. Uma entidade de serviço é criada no Azure usando o certificado. A entidade de serviço pode ser autenticada em relação ao uso do certificado. A chave privada do certificado está com o usuário, e a parte pública está com o Azure. Na próxima seção, uma entidade de serviço será criada usando o certificado criado nesta seção.

## Criar uma entidade de serviço usando credenciais de certificado

Uma entidade de serviço pode ser criada usando o portal do Azure, a CLI do Azure ou o Azure PowerShell. O script para criar uma entidade de serviço usando o Azure PowerShell está disponível nesta seção.

Depois de fazer logon no Azure, o certificado criado na seção anterior é convertido em codificação base64. Uma nova entidade de serviço, **azureforarchitects**, é criada, e a credencial de certificado é associada à entidade de serviço recém-criada. Por fim, a nova entidade de serviço recebe permissões de controle de acesso baseadas em função de colaborador na assinatura:

```
Login-AzAccount

$certKey = [system.Convert]::ToBase64String($cert.GetRawCertData())

$sp = New-AzADServicePrincipal -DisplayName "azureforarchitects"

New-AzADSpCredential -ObjectId $sp.Id -CertValue $certKey -StartDate
$cert.NotBefore -EndDate $cert.NotAfter

New-AzRoleAssignment -RoleDefinitionName contributor -ServicePrincipalName
$sp.ApplicationId

Get-AzADServicePrincipal -ObjectId $sp.Id

$cert.Thumbprint

Get-AzSubscription
```

Para criar um ativo de conexão, a ID da aplicação pode ser obtida usando o cmdlet **Get-AzADServicePrincipal**, e o resultado é mostrado na Figura 4.9:

```
ServicePrincipalNames : {http://azureforarchitects, ef52538d-9eb6-45e0-bf67-7f484b84cd25}
ApplicationId        : ef52538d-9eb6-45e0-bf67-7f484b84cd25
ObjectType          : ServicePrincipal
DisplayName         : azureforarchitects
Id                  : 15ee7335-9b96-4ae7-957f-62e4997e7b4d
Type                :
```

Figura 4.9: verificando a entidade de serviço

A impressão digital do certificado pode ser obtida usando a referência do certificado juntamente com **SubscriptionId**, que pode ser obtida usando o cmdlet **Get-AzSubscription**.

## Conexões

Os ativos de conexão são usados para criar informações de conexão para serviços externos. Nesse sentido, até mesmo o Azure é considerado um serviço externo. Os ativos de conexão têm todas as informações necessárias para se conectar com êxito a um serviço. Há três tipos de conexão fornecidos pela Automação do Azure e prontos para uso:

- Azure
- Certificado clássico do Azure
- Entidade de serviço do Azure

É recomendável usar a entidade de serviço do Azure para se conectar aos recursos do Azure Resource Manager e usar o certificado clássico do Azure para recursos clássicos do Azure. É importante observar que a Automação do Azure não fornece nenhum tipo de conexão para se conectar ao Azure usando credenciais, como nome de usuário e senha.

Os certificados clássicos do Azure e do Azure têm natureza semelhante. Ambos ajudam a se conectar aos recursos baseados em API do gerenciamento de serviços do Azure. Na verdade, a Automação do Azure cria uma conexão de certificado clássico do Azure ao criar uma conta de execução clássica.

A entidade do serviço do Azure é usada internamente por contas de execução para se conectar a recursos baseados no Azure Resource Manager.

Um novo ativo de conexão do tipo **AzureServicePrincipal** é mostrado na Figura 4.10. Ele precisa de:

- O nome da conexão. É obrigatório fornecer um nome.
- Uma descrição da conexão. Esse valor é opcional.
- Selecione um **Tipo** apropriado. É obrigatório selecionar uma opção; **AzureServicePrincipal** é selecionada para criar um ativo de conexão para todas as finalidades neste capítulo.
- **ApplicationId**, também conhecido como **ClientId**, é a ID da aplicação gerada durante a criação de uma entidade de serviço. A próxima seção mostra o processo de criação de uma entidade de serviço usando o Azure PowerShell. É obrigatório fornecer uma ID de aplicação.
- **TenantId** é o identificador exclusivo do locatário. Essas informações estão disponíveis no portal do Azure ou usando o cmdlet **Get-AzSubscription**. É obrigatório fornecer um identificador do locatário.
- **CertificateThumbprint** é o identificador do certificado. Esse certificado já deve estar carregado para a Automação do Azure usando o ativo do certificado. É obrigatório fornecer uma impressão digital do certificado.
- **SubscriptionId** é o identificador da assinatura. É obrigatório fornecer uma ID de assinatura.

Você pode adicionar uma nova conexão usando a folha **Conexões** na conta da Automação, conforme mostrado na Figura 4.10:

The screenshot shows the Azure Automation 'Connections' blade for an account named 'bookaccount'. On the left, there's a sidebar with 'Credentials', 'Connections' (which is selected), 'Certificates', 'Variables', 'Related Resources' (with 'Linked workspace', 'Event grid', and 'Start/Stop VM'), 'Account Settings', and 'Properties'. The main area shows a table with one row: 'No connections found.' Below the table is a 'New Connection' dialog. The 'Type' dropdown is set to 'AzureServicePrincipal'. The 'ApplicationId' field contains a placeholder 'XXXXXX-XXXXXX-XXXXXX-XXXXXX'. The 'TenantId' field contains a placeholder 'XXXXXXXX-XXXXXX-XXXXXX-XXXXXX'. The 'CertificateThumbprint' field contains a placeholder 'XXXXXXXXXXXXXXXXXXXX'. The 'SubscriptionId' field is empty. At the bottom right of the dialog is a 'Create' button.

Figura 4.10: adicionando uma nova conexão à conta da Automação

## Criação e execução do runbook

A Automação do Azure permite a criação de scripts de automação, conhecidos como runbooks. Vários runbooks podem ser criados usando o portal do Azure ou o PowerShell ISE. Eles também podem ser importados da **Galeria de runbooks**. A galeria pode ser pesquisada por funcionalidade específica, e todo o código é exibido no runbook.

Um runbook pode aceitar valores de parâmetro, assim como um script normal do PowerShell. O próximo exemplo leva um único parâmetro chamado **connectionName** do tipo **string**. É obrigatório fornecer um valor para esse parâmetro ao executar esse runbook:

```
param(  
    [parameter(mandatory=$true)]  
    [string] $connectionName  
)  
  
$connection = Get-AutomationConnection -name $connectionName  
$subscriptionid = $connection.subscriptionid  
$tenantid = $connection.tenantid  
$applicationid = $connection.applicationid  
$cretThumbprint = $connection.CertificateThumbprint  
  
Login-AzureRMAccount -CertificateThumbprint $cretThumbprint  
-ApplicationId $applicationid -ServicePrincipal -Tenant $tenantid  
  
Get-AzureRMVM
```

O runbook usa o cmdlet **Get-AutomationConnection** para fazer referência ao ativo de conexão compartilhada. O nome do ativo está no valor do parâmetro. Depois que a referência ao ativo de conexão for feita, os valores da referência de conexão serão preenchidos na variável **\$connection** e, posteriormente, eles serão atribuídos a várias outras variáveis.

O cmdlet **Login-AzureRMAccount** é autenticado com o Azure e fornece os valores obtidos do objeto de conexão. Ele usa a entidade de serviço criada anteriormente neste capítulo para autenticação.

Por fim, o runbook invoca o cmdlet **Get-AzureRMVm** para listar todas as VMs na assinatura.

Por padrão, a Automação do Azure ainda fornece módulos **AzureRM** para trabalhar com o Azure. Ela não instala módulos **Az** por padrão. Mais adiante, instalaremos um módulo **Az** manualmente na conta da Automação do Azure e usaremos cmdlets em runbooks.

## Runbooks pais e filhos

Os runbooks têm um ciclo de vida, desde a criação até a execução. Esses ciclos de vida podem ser divididos em status de criação e status de execução.

O ciclo de vida de criação é mostrado na Figura 4.11.

Quando um novo runbook é criado, ele tem o status **Novo** e, como é editado e salvo várias vezes, assume o status **Em edição**. Por fim, quando é publicado, o status é alterado para **Publicado**. Também é possível editar um runbook publicado e, nesse caso, ele voltará para o status **Em edição**.



Figura 4.11: ciclo de vida de criação

O ciclo de vida de execução é descrito a seguir.

O ciclo de vida começa com o início de uma solicitação de execução do runbook. Um runbook pode ser executado de várias maneiras:

- Manualmente no portal do Azure
- Usando um runbook pai como runbook filho
- Por meio de um webhook

Não importa como um runbook é iniciado; o ciclo de vida permanece o mesmo. Uma solicitação para executar o runbook é recebida pelo mecanismo da Automação. O mecanismo da Automação cria um trabalho e o atribui a um trabalhador do runbook. Atualmente, o runbook tem o status **Enfileirado**.

Há vários trabalhadores do runbook, e o escolhido seleciona a solicitação de trabalho e altera o status para **Iniciando**. Nesta fase, se houver algum problema de script e análise no script, o status será alterado para **Falha**, e a execução será interrompida.

Depois que a execução do runbook for iniciada pelo trabalhador, o status será alterado para **Em execução**. O runbook pode ter vários status diferentes quando está em execução.

O runbook mudará seu status para **Concluído** se a execução acontecer sem exceções não tratadas e de encerramento.

O runbook em execução pode ser interrompido manualmente pelo usuário e terá o status **Interrompido**.

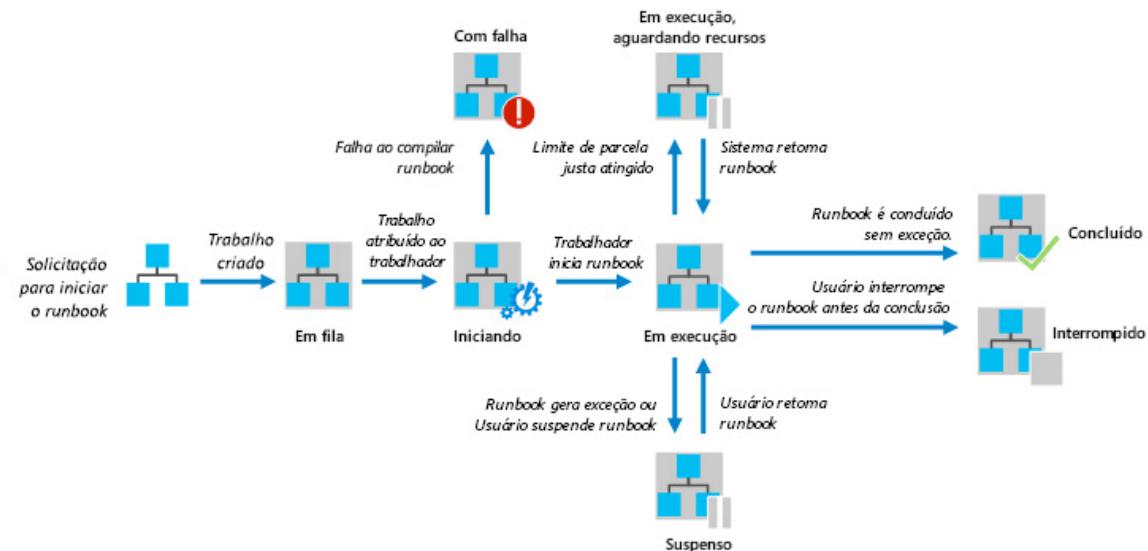


Figura 4.12: o ciclo de vida de execução para runbooks

O usuário também pode suspender e retomar a execução do runbook.

## Criar um runbook

Um runbook pode ser criado no portal do Azure, acessando o item do menu **Runbook** no painel de navegação à esquerda. Um runbook tem um nome e um tipo. O tipo determina a linguagem de script usada para criar o runbook. Já abordamos as possíveis linguagens e, neste capítulo, o PowerShell será usado principalmente para todos os exemplos.

A criação de um runbook do PowerShell é exatamente igual à criação de um script do PowerShell. Ele pode declarar e aceitar vários parâmetros: os parâmetros podem ter atributos como tipos de dados, que são obrigatórios (assim como quaisquer atributos de parâmetro do PowerShell). Ele pode invocar cmdlets do PowerShell cujos módulos estão disponíveis e já carregados e declarados e pode invocar funções e retornar a saída.

Um runbook também pode invocar outro runbook. Ele pode invocar um runbook filho em linha no processo e contexto originais ou em um processo e contexto separados.

A inovação de um runbook em linha é semelhante à invocação de um script do PowerShell. O próximo exemplo invoca um runbook filho usando a abordagem em linha:

```
. \ConnectAzure.ps1 -connectionName "azureforarchitectsconnection"
```

```
Get-AzSqlServer
```

No código anterior, vimos como o runbook **ConnectAzure** aceita um parâmetro chamado **connectionName**, e um valor apropriado é fornecido a ele. Esse runbook cria uma conexão com o Azure depois de ser autenticado com ele usando uma entidade de serviço. Confira a sintaxe para invocar o runbook filho. É muito semelhante à invocação de um script geral do PowerShell juntamente com parâmetros.

A próxima linha de código, **Get-AzVm**, obtém as informações relevantes do Azure e lista os detalhes da VM. Você observará que, embora a autenticação ocorra em um runbook filho, o cmdlet **Get-AzVm** é bem-sucedido e lista todas as VMs na assinatura porque o runbook filho é executado no mesmo trabalho que o runbook pai, e eles compartilham o contexto.

Como alternativa, um runbook filho pode ser invocado usando o cmdlet **Start-AzurermAutomationRunbook** fornecido pela Automação do Azure. Esse cmdlet aceita o nome da conta da Automação, o nome do grupo de recursos e o nome do runbook, juntamente com os parâmetros, conforme mencionado aqui:

```
$params = @{"connectionName"="azureforarchitectsconnection"}

$job = Start-AzurermAutomationRunbook 'bookaccount'
    -AutomationAccountName 'bookaccount'
    -Name 'ConnectAzure'
    -ResourceGroupName 'automationrg' -parameters $params

if($job -ne $null) {
    Start-Sleep -s 100
    $job = Get-AzureAutomationJob -Id $job.Id -AutomationAccountName
    'bookaccount'

    if ($job.Status -match "Completed") {
        $jobout = Get-AzureAutomationJobOutput 'bookaccount'
            -Id $job.Id
            -AutomationAccountName 'bookaccount'
            -Stream Output
        if ($jobout) {Write-Output $jobout.Text}
    }
}
```

O uso dessa abordagem cria um novo trabalho diferente do trabalho pai, e eles são executados em diferentes contextos.

## Usar módulos Az

Até agora, todos os exemplos usaram módulos **AzureRM**. Os runbooks anteriormente mostrados serão reescritos para usar cmdlets no módulo **Az**.

Como mencionado anteriormente, os módulos **Az** não são instalados por padrão. Eles podem ser instalados usando o item do menu **Galeria de módulos** na Automação do Azure.

Pesquise **Az** na galeria, e os resultados mostrarão vários módulos relacionados a ele. Se o módulo **Az** for selecionado para ser importado e instalado, ele lançará um erro informando que seus módulos dependentes não estão instalados e que eles devem ser instalados antes de instalar o módulo atual. O módulo pode ser encontrado na folha **Galeria de módulos** pesquisando **Az**, conforme mostrado na Figura 4.13:

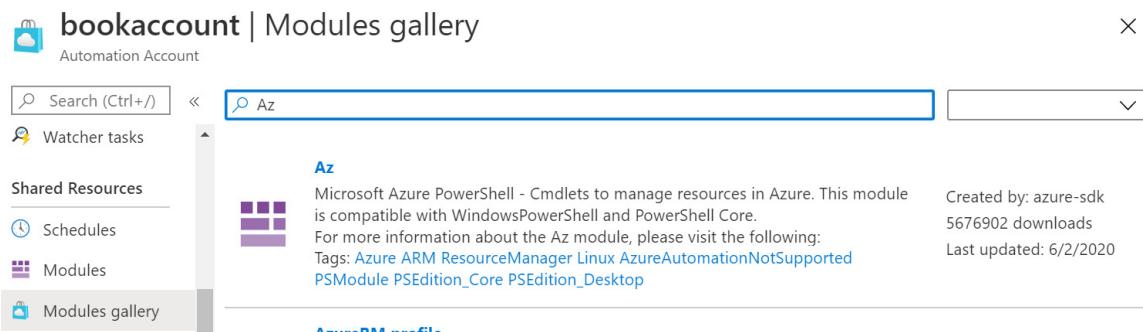


Figura 4.13: encontrando o módulo Az na folha Galeria de módulos

Em vez de selecionar o módulo **Az**, selecione **Az.Accounts** e importe o módulo seguindo o assistente, conforme mostrado na Figura 4.14:

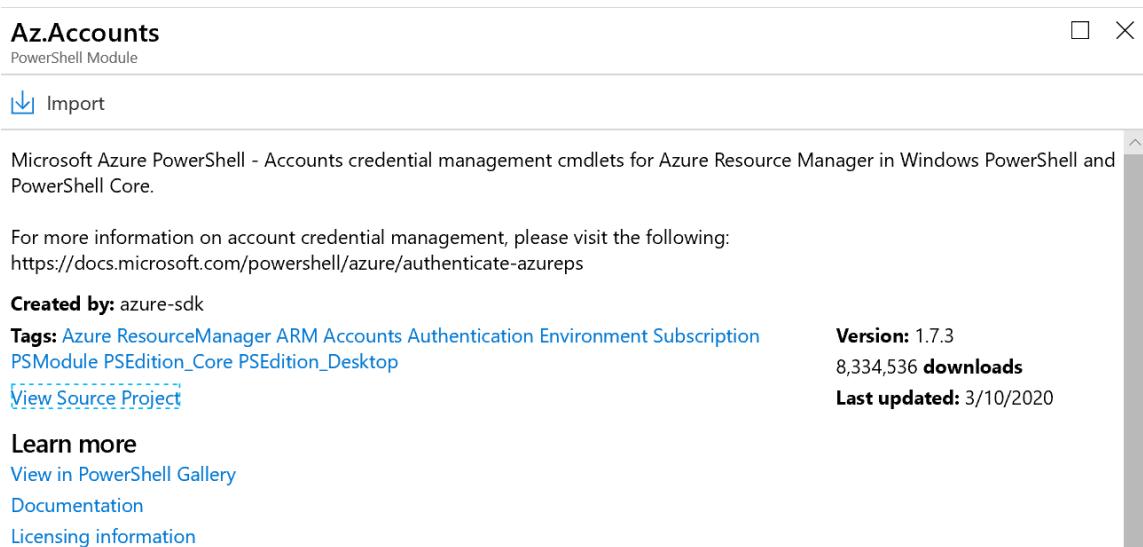


Figura 4.14: importando o módulo Az.Accounts

Depois de instalar **Az.Accounts**, o módulo **Az.Resources** pode ser importado. Os cmdlets relacionados a máquinas virtuais do Azure estão disponíveis no módulo **Az.Compute** e também podem ser importados usando o mesmo método que usamos para importar **Az.Accounts**.

Depois que esses módulos forem importados, os runbooks poderão usar os cmdlets fornecidos por esses módulos. O runbook **ConnectAzure** mostrado anteriormente foi modificado para usar o módulo **Az**:

```
param(  
    [parameter(mandatory=$true)]  
    [string] $connectionName  
)  
  
$connection = Get-AutomationConnection -name $connectionName  
$subscriptionid = $connection.subscriptionid  
$tenantid = $connection.tenantid  
$applicationid = $connection.applicationid  
$cretThumbprint = $connection.CertificateThumbprint  
  
Login-AzAccount -CertificateThumbprint $cretThumbprint  
-ApplicationId $applicationid -ServicePrincipal  
-Tenant $tenantid -SubscriptionId $subscriptionid  
  
Get-AzVm
```

As duas últimas linhas do código são importantes. Elas estão usando cmdlets **Az**, em vez de cmdlets **AzureRM**.

A execução desse runbook fornecerá resultados semelhantes a este:

The screenshot shows the Azure Runbook execution interface. At the top, it displays the navigation path: Home > bookaccount | Runbooks > ConnectAzure (bookaccount/ConnectAzure) > ConnectAzure 3/26/2020, 7:19 AM. Below this, the runbook title 'ConnectAzure' is shown with a timestamp '3/26/2020, 7:19 AM'. A 'Job' icon is present. Below the title, there are buttons for Resume, Stop, Suspend, Refresh, and a link to the source snapshot. The runbook details section shows the following information:

- Id:** 587f7695-fa49-4f99-8a9c-772f1a6c4788
- Status:** Completed
- Ran on:** localrunbookexecutionengine
- Ran As:** User
- Created:** 3/26/2020, 7:19:25 AM
- Last Update:** 3/26/2020, 7:20:05 AM
- Runbook:** ConnectAzure
- Source snapshot:** [View source snapshot](#)

Below the runbook details, there are tabs for Input, Output (which is selected), Errors, Warnings, All Logs, and Exception. The Output tab displays the command output and environment variables. The command output includes several GUIDs and a long string of environment variable names. The environments section lists several cloud regions.

```

9755ffce-e94b-4332-9be8-1ade15e78909
771f1cf4-b1ac-4f2e-ad21-de39ea201e7e
ef52538d-9eb6-45e0-bf67-7f484b84cd25
43A06770461183DE1725548BD76DA85B7FB9171A

Environments
-----
{[AzureChinaCloud, AzureChinaCloud], [AzureCloud, AzureCloud], [AzureGermanCloud, AzureGermanCloud], [AzureUSGovernme...}

ResourceGroupName : SPARK
Id : /subscriptions/9755ffce-e94b-4332-9be8-1ade15e78909/resourceGroups/SPARK/providers/Microsoft.Compute/virtualMachines/spark
VmId : 3a10b076-559d-41c2-b0c6-ab60674f2ae
Name : spark
Type : Microsoft.Compute/virtualMachines
Location : westeurope
LicenseType :
Tags :
AvailabilitySetReference :
DiagnosticsProfile :
Extensions :
HardwareProfile : Microsoft.Azure.Management.Compute.Models.HardwareProfile
InstanceView :
NetworkProfile : Microsoft.Azure.Management.Compute.Models.NetworkProfile
OSProfile : Microsoft.Azure.Management.Compute.Models.OSProfile
BillingProfile :
Plan :
ProvisioningState : Succeeded
StorageProfile : Microsoft.Azure.Management.Compute.Models.StorageProfile
DisplayHint : Compact
Identity :
Zones :
FullyQualifiedDomainName :
AdditionalCapabilities :
ProximityPlacementGroup :

```

Figura 4.15: o módulo Az.Accounts importado com êxito

Na próxima seção, trabalharemos com webhooks.

## Webhooks

Os webhooks ficaram famosos após o advento dos pontos de extremidade REST e das cargas de dados JSON. Os webhooks são um conceito importante e uma decisão arquitetônica na extensibilidade de qualquer aplicação. Os webhooks são espaços reservados que são deixados em áreas especiais de uma aplicação para que o usuário da aplicação possa preencher esses espaços reservados com URLs de ponto de extremidade com lógica personalizada. A aplicação invocará a URL do ponto de extremidade, passando automaticamente nos parâmetros necessários, e, em seguida, executará o login disponível nele.

Os runbooks da Automação do Azure podem ser invocados manualmente no portal do Azure. Eles também podem ser invocados usando cmdlets do PowerShell e a CLI do Azure. Há SDKs disponíveis em várias linguagens que são capazes de invocar runbooks.

Os webhooks são uma das formas mais poderosas de invocar um runbook. É importante observar que os runbooks que contêm a lógica principal nunca devem ser expostos diretamente como um webhook. Eles devem ser chamados usando um runbook pai, e o runbook pai deve ser exposto como um webhook. O runbook pai deve garantir que as verificações apropriadas sejam feitas antes de invocar o runbook filho principal.

A primeira etapa para criar um webhook é criar um runbook normalmente, como feito anteriormente. Depois que um runbook for criado, ele será exposto como um webhook.

Um novo runbook baseado no PowerShell chamado **exposedrunbook** é criado. Esse runbook assume um único parâmetro, **\$WebhookData**, do tipo de objeto. Ele deve ser chamado **verbatim**. Esse objeto é criado pelo tempo de execução da Automação do Azure e é fornecido ao runbook. O tempo de execução da Automação do Azure cria esse objeto depois de obter os valores do cabeçalho de solicitação HTTP e o conteúdo do corpo e preenche as propriedades **RequestHeader** e **RequestBody** deste objeto:

```
param(  
    [parameter(mandatory=$true)]  
    [object] $WebhookData  
  
)  
  
$webhookname = $WebhookData.WebhookName  
$headers = $WebhookData.RequestHeader  
$body = $WebhookData.RequestBody
```

```

Write-output "webhook header data"
Write-Output $webhookname
Write-output $headers.message
Write-output $headers.subject

$connectionname = (ConvertFrom-Json -InputObject $body)

./connectAzure.ps1 -connectionName $connectionname[0].name

```

As três propriedades importantes deste objeto são **WebhookName**, **RequestHeader** e **RequestBody**. Os valores são obtidos dessas propriedades e enviados para o fluxo de saída pelo runbook.

O cabeçalho e o conteúdo do corpo podem ser qualquer coisa que o usuário forneça ao invocar o webhook. Esses valores são preenchidos nas respectivas propriedades e são disponibilizados no runbook. No exemplo anterior, há dois cabeçalhos definidos pelo chamador, por exemplo, o cabeçalho de **mensagem** e **status**. O chamador também fornecerá o nome da conexão compartilhada para ser usada como parte do conteúdo do corpo.

Depois que o runbook for criado, ele deverá ser publicado para que um webhook possa ser criado. Depois de publicar o runbook, ao clicar no menu **Webhook** na parte superior, você iniciará o processo de criação de um novo webhook para o runbook, conforme mostrado na Figura 4.16:

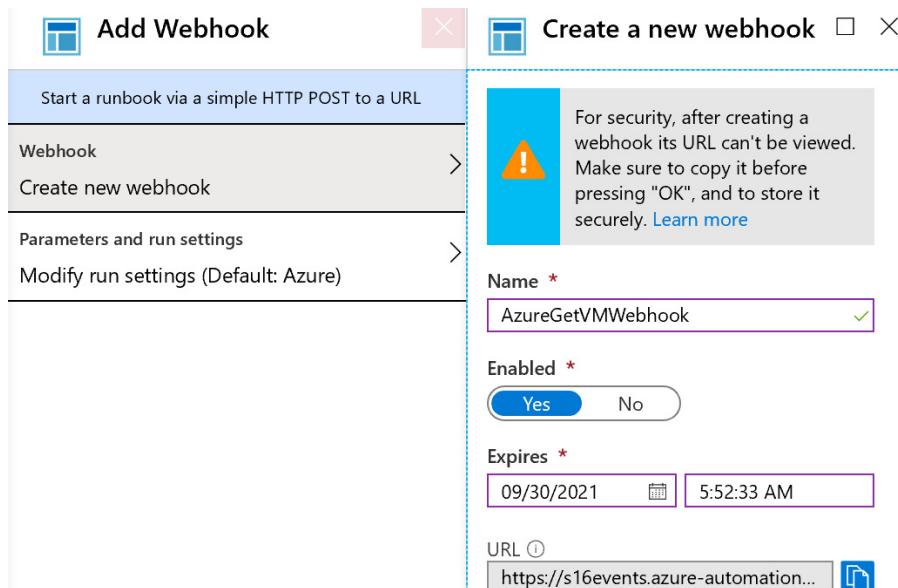


Figura 4.16: criando um webhook

Um nome para o webhook deve ser fornecido. Esse valor está disponível no runbook usando o parâmetro **WebhookData** com o nome de propriedade **WebhookName**.

O webhook pode estar no estado **habilitado** ou **desabilitado** e pode expirar em uma data e hora específicas. Ele também gera uma URL exclusiva para esse webhook e runbook. Essa URL deve ser fornecida a qualquer pessoa que queira invocar o webhook.

## Invocar um webhook

Os webhooks são invocados como solicitações HTTP usando o método **POST**. Quando um webhook é invocado, a solicitação HTTP tem como destino a Automação do Azure para iniciar um runbook. Ele cria o objeto **WebHookData**, preenchendo-o com o cabeçalho HTTP de entrada e os dados do corpo e cria um trabalho a ser escolhido por um trabalhador do runbook. Essa chamada usa a URL do webhook gerada na etapa anterior.

O webhook pode ser invocado usando o Postman, por qualquer código que tenha a capacidade de chamar um ponto de extremidade **REST** usando o método **POST**.

No próximo exemplo, o PowerShell será usado para invocar o webhook:

```
$uri = "https://s16events.azure-automation.net/
webhooks?token=rp0w93L60fAPYZQ4vryx1%2baN%2bS1Hz4F3qVdUaKUDzgM%3d"

$connection = @(
    @{
        name="azureforarchitectsconnection"
    }

)
$body = ConvertTo-Json -InputObject $connection
$header = @{
    subject="VMS specific to Ritesh";message="Get all virtual
    machine details"
}

$response = Invoke-WebRequest -Method Post -Uri $uri -Body $body -Headers
$header
$jobid = (ConvertFrom-Json ($response.Content)).jobids[0]
```

O código do PowerShell declara a URL para o webhook e cria o formato do corpo no JSON, com **nome** definido como **azureforarchitectsconnection** e um cabeçalho com dois pares nome-valor de cabeçalho: **assunto** e **mensagem**. Os dados do corpo e do cabeçalho podem ser obtidos no runbook usando o parâmetro **WebhookData**.

O cmdlet **invoke-webrequest** gera a solicitação no ponto de extremidade mencionado anteriormente usando o método **POST**, fornecendo o cabeçalho e o corpo.

A solicitação é de natureza assíncrona e, em vez da saída do runbook, o identificador do trabalho é retornado como uma resposta HTTP. Ele também está disponível no conteúdo de resposta. O trabalho é mostrado na Figura 4.17:

**exposedRunbook 3/26/2020, 6:25 AM**

Job

► Resume     Stop    || Suspend    Refresh

Id : 5854db0a-9bec-4aad-a47d-5749308d69ab

Status : Completed

Ran on : Azure

Ran As : User

**Input**   Output   Errors   Warnings   All Logs   Exception

Name

WEBHOOKDATA

Figura 4.17: verificando o trabalho

Ao clicar em **WEBHOOKDATA**, você verá os valores que chegaram no serviço de automação do runbook na solicitação HTTP:

**Input parameter**

WEBHOOKDATA

```
{"WebhookName": "AzureGetVMWebhook", "RequestBody": "[\r\n  \r\n    \"name\": \"azureforarchitectsconnection\"\r\n  ]", "RequestHeader": {"Host": "s16events.azure-automation.net", "User-Agent": "Mozilla/5.0", "message": "Get all virtual machine details", "subject": "VMS specific to Ritesh", "x-ms-request-id": "34da1397-f0d1-4ec4-b86a-fd705bfaaf3"}}
```

Figura 4.18: verificando a saída

Ao clicar no menu de saída, você verá a lista de VMs e SQL Server na assinatura.

Os próximos conceitos importantes na Automação do Azure são o Azure Monitor e os Hybrid Workers, que serão explicados nas próximas seções detalhadamente.

## Invocar um runbook do Azure Monitor

Os runbooks da Automação do Azure podem ser invocados como respostas aos alertas gerados no Azure. O Azure Monitor é o serviço central que gerencia logs e métricas entre recursos e grupos de recursos em uma assinatura. Você pode usar o Azure Monitor para criar novas definições e regras de alerta que, quando acionadas, podem executar runbooks da Automação do Azure. Elas podem invocar um runbook da Automação do Azure em seu formulário padrão ou um webhook que pode executar seu runbook associado. Essa integração entre o Azure Monitor e a capacidade de invocar runbooks possibilita várias oportunidades de automação para corrigir automaticamente o ambiente, expandir e reduzir os recursos de computação ou tomar medidas corretivas sem nenhuma intervenção manual.

Os alertas do Azure podem ser criados e configurados em recursos individuais e níveis de recursos, mas é sempre recomendável centralizar as definições de alerta para facilitar e melhorar a manutenção e a administração.

Vamos passar pelo processo de associar um runbook a um alerta e invocar o runbook como parte do alerta que está sendo gerado.

A primeira etapa é criar um novo alerta, conforme mostrado na Figura 4.19:

The screenshot shows the Azure Monitor Alerts interface. On the left, there's a sidebar with links: Overview, Activity log, Alerts (which is selected and highlighted in grey), Metrics, Logs, Service Health, and Workbooks. The main area has a header with 'Home > Monitor | Alerts' and a search bar. Below the header are buttons for 'New alert rule' (highlighted in yellow), 'Manage alert rules', 'Manage actions', 'View classic alerts', 'Refresh', and 'Provide feedback'. A message says 'Don't see a subscription? Open Directory + Subscription settings'. It shows a 'Subscription' dropdown set to 'RiteshSubscription' and a 'Resource group' dropdown set to '22 selected'. At the bottom, a message reads 'All is good! You have no alerts.'

Figura 4.19: criando uma regra de alerta

Selecione um recurso que deve ser monitorado e avaliado para geração de alertas. Um grupo de recursos foi selecionado na lista e habilita automaticamente todos os recursos no grupo de recursos. É possível remover as seleções de recursos do grupo de recursos:

**Select a resource**

Select the resource(s) you want to monitor. Available signal types for your selection will show up on the bottom right.

Filter by subscription \* ⓘ Filter by resource type ⓘ Filter by location ⓘ

RiteshSubscription Virtual machines All

Search to filter items...

Resource	Location
<input type="checkbox"/> RiteshSubscription	
<input type="checkbox"/> rg-harvestingclouds-infra101	East US
<input type="checkbox"/> vmAccounts101	East US
<input type="checkbox"/> spark	West Europe
<input type="checkbox"/> spark	West Europe
<input checked="" type="checkbox"/> visualstudio2017	West Europe
<input checked="" type="checkbox"/> visualstudio	West Europe

Figura 4.20: selecionando o escopo do alerta

Configure a condição e as regras que devem ser avaliadas. Selecione o nome do sinal **Desligar máquina virtual** depois de selecionar **Log de atividade** como **Tipo de sinal**:

**Configure signal logic**

Choose a signal below and configure the logic on the next screen to define the alert condition.

Signal type ⓘ Monitor service ⓘ

Activity Log All

Displaying 1 - 18 signals out of total 18 signals

Search by signal name

Signal name	↑↓	Signal type	↑↓	Monitor service	↑↓
All Administrative operations		Activity Log		Administrative	
Create or Update Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Delete Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Start Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
<b>Power Off Virtual Machine (Microsoft.Compute/virtualMachines)</b>		Activity Log		Administrative	
Redeploy Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Restart Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Deallocate Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Generalize Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Capture Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Run Command on Virtual Machine (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	
Convert Virtual Machine disks to Managed Disks (Microsoft.Compute/virtualM...)		Activity Log		Administrative	
Perform Maintenance Redeploy (Microsoft.Compute/virtualMachines)		Activity Log		Administrative	

Figura 4.21: selecionando o tipo de sinal

A janela resultante permitirá que você configure a **Condição/lógica do alerta**. Selecione **crítico** para **Nível de evento** e defina **Status** como **Bem-sucedido**:



Alert logic

Event Level ⓘ

Critical

Status ⓘ

Succeeded

Event initiated by ⓘ

\* (All services and users)

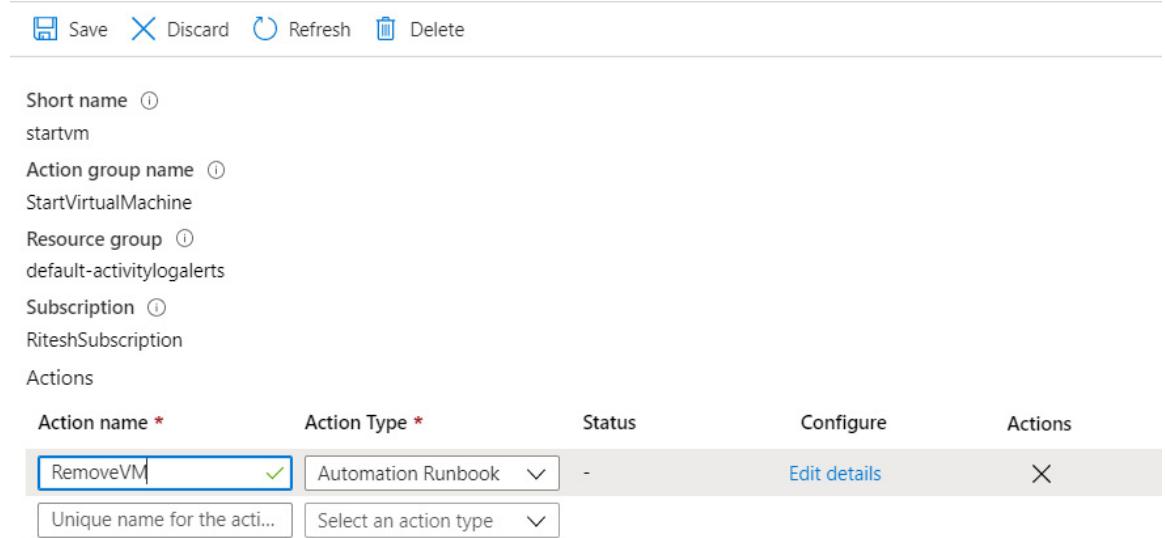
Condition preview

*Whenever the Activity Log has an event with Category='Administrative', Signal name='Power Off Virtual Machine (Microsoft.Compute/virtualMachines)', Level='critical', Status='succeeded'*

Figura 4.22: configurando a lógica do alerta

Depois de determinar a condição do alerta, temos a configuração mais importante, que define a resposta ao alerta invocando um runbook. Podemos usar **Grupos de ação** para configurar a resposta a um alerta. Ele fornece várias opções para invocar uma função do Azure, um webhook ou um runbook da Automação do Azure, bem como enviar emails e mensagens SMS.

Crie um grupo de ação fornecendo um nome, um nome abreviado, a assinatura de hospedagem, um grupo de recursos e um **Nome de ação**. Correspondendo ao **Nome da ação**, selecione a opção **Runbook de automação** como **Tipo de ação**:



Save Discard Refresh Delete

Short name ⓘ

startvm

Action group name ⓘ

StartVirtualMachine

Resource group ⓘ

default-activitylogalerts

Subscription ⓘ

RiteshSubscription

Actions

Action name *	Action Type *	Status	Configure	Actions
RemoveVM	Automation Runbook	-	Edit details	X
Unique name for the acti...	Select an action type			

[Azure Privacy Statement](#)

[Azure Alerts Pricing](#)

Figura 4.23 configurando o grupo de ação

A seleção de um runbook de automação abrirá outra folha para selecionar uma conta e um runbook apropriados da Automação do Azure. Vários runbooks estão disponíveis e prontos para uso, e um deles foi usado aqui:

**Configure Runbook** ×

Run runbook \*

Enabled  Disabled

Runbook source \* ⓘ

Built-in  User

Runbook \*

Remove VM

This runbook will remove (delete) the Azure virtual machine that triggered the alert.

Subscription \*

RiteshSubscription

Automation account \*

bookaccount

Enable the common alert schema \*

Yes  No

[Learn more about common alert schema](#)

Figura 4.24: criando o runbook

Por fim, forneça um nome e um grupo de recursos de hospedagem para criar um novo alerta.

Se a VM for desalocada manualmente, a condição do alerta será atendida e gerará um alerta:

Name	Severity	Monitor Condition	Alert State	Affected resource	Monitor Service	Signal Type	Fired time	Subscription
startVMalert	Sev4	<span style="color: orange;">⚠ Fired</span>	New	<span style="color: blue;">visualstudio</span>	ActivityLog Administr...	Log	5/10/2020, 8:32:12 PM	RiteshSubscription
startVMalert	Sev4	<span style="color: orange;">⚠ Fired</span>	New	<span style="color: blue;">visualstudio</span>	ActivityLog Administr...	Log	5/10/2020, 8:32:05 PM	RiteshSubscription

Figura 4.25: testando alertas

Se você verificar os detalhes da VM após alguns segundos, verá que a VM está sendo excluída:

Resource group (change)	: visualStudio2017	Azure Spot	: N/A
Status	: Deleting	Public IP address	: 52.174.123.111
Location	: West Europe	Private IP address	: 10.0.0.4
Subscription (change)	: RiteshSubscription	Public IP address (IPv6)	: -
Subscription ID	: 9755ffce-e94b-4332-9be8-1ade15e78909	Private IP address (IPv6)	: -
Computer name	: (not available)	Virtual network/subnet	: visualStudio2017-vnet/default
Operating system	: Windows	DNS name	: customerwrite.westeuropew.cloudapp.azure.com
Size	: Standard DS1 v2 (1 vcpus, 3.5 GiB memory)		
Tags (change)	: Click here to add tags		

Figura 4.26: verificando os resultados

## Hybrid Workers

Até agora, toda a execução dos runbooks foi principalmente na infraestrutura fornecida pelo Azure. Os trabalhadores do runbook são os recursos de computação do Azure que são provisionados pelo Azure com módulos e ativos apropriados implantados neles.

Todas as execuções de runbooks ocorrem nesta computação. No entanto, é possível que os usuários tragam sua própria computação e executem o runbook nessa computação fornecida pelo usuário, em vez da computação padrão do Azure.

Isso traz várias vantagens. Em primeiro lugar, toda a execução e seus logs são de propriedade do usuário, e o Azure não tem visibilidade disso. Em segundo lugar, a computação fornecida pelo usuário pode estar em qualquer nuvem, bem como na infraestrutura local.

### Adicionar um Hybrid Worker envolve várias etapas

- Em primeiro lugar, um agente precisa ser instalado na computação fornecida pelo usuário. A Microsoft fornece um script que pode fazer o download e configurar o agente automaticamente. Esse script está disponível em <https://www.powershellgallery.com/packages/New-OnPremiseHybridWorker/1.6>.

O script também pode ser executado no PowerShell ISE como um administrador no servidor que deve fazer parte do Hybrid Worker usando o seguinte comando:

```
Install-Script -Name New-OnPremiseHybridWorker -verbose
```

- Depois que o script for instalado, ele poderá ser executado juntamente com parâmetros relacionados aos detalhes da conta da Automação do Azure. Um nome também é fornecido para o Hybrid Worker. Se o nome já não existir, ele será criado. Se existir, o servidor será adicionado ao Hybrid Worker existente. É possível ter vários servidores em um único Hybrid Worker, e é possível ter vários Hybrid Workers também:

```
New-OnPremiseHybridWorker.ps1 -AutomationAccountName bookaccount  
-AAResourceGroupName automationrg '  
-HybridGroupName "localrunbookexecutionengine" '  
-SubscriptionID xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

- Depois que a execução terminar, navegue de volta para o portal para ver uma entrada para um Hybrid Worker, conforme mostrado na Figura 4.27:

The screenshot shows the Azure portal interface. At the top, there are 'Configure' and 'Refresh' buttons. Below them, two tabs are visible: 'User hybrid worker groups' (which is selected and highlighted with a blue dashed border) and 'System hybrid worker groups'. A search bar below the tabs contains the placeholder 'Search to filter items...'. The main content area displays a table with three columns: 'Group name', 'Number of workers', and 'Last registration time'. The table contains one row with the data: 'localrunbookexecutionengine', '1', and '3/26/2020, 7:05 AM' respectively.

Group name	Number of workers	Last registration time
localrunbookexecutionengine	1	3/26/2020, 7:05 AM

**Figura 4.27: verificando grupos do Hybrid Worker do usuário**

- Se, neste momento, um runbook do Azure for executado com uma dependência do módulo **Az** e um certificado personalizado carregado para o ativo do certificado, ele falhará com erros relacionados ao módulo **Az** e o certificado não será encontrado:

The screenshot shows the Azure portal interface. At the top, it says 'ConnectAzure 3/26/2020, 7:08 AM'. Below that are buttons for 'Resume', 'Stop', 'Suspend', and 'Refresh'. The main area shows a table with details about the job: Id (a8de935e-9745-43b5-b614-32b8483dac48), Status (Completed), Ran on (localrunbookexecutionengine), Ran As (User), Created (3/26/2020, 7:08:00 AM), Last Update (3/26/2020, 7:08:36 AM), Runbook (ConnectAzure), and Source snapshot (View source snapshot). Below the table, there are tabs for 'Input', 'Output', 'Errors' (which is selected and highlighted), 'Warnings', 'All Logs', and 'Exception'. Under the 'Errors' tab, it says '2 1'. A table lists the errors: the first error is 'No certificate was found in the certificate store with thumbprint 43A06770461183DE1725548BD76DA85B7FB9171A' (Time: 3/26/2020, 7:08:34 AM, Type: Error), and the second error is 'System.Management.Automation.CommandNotFoundException: The term 'Get-azvm' is not recognized as the name of a cmdlet, function, script file, or operable program.' (Time: 3/26/2020, 7:08:35 AM, Type: Error).

Time	Type	Details
3/26/2020, 7:08:34 AM	Error	No certificate was found in the certificate store with thumbprint 43A06770461183DE1725548BD76DA85B7FB9171A
3/26/2020, 7:08:35 AM	Error	System.Management.Automation.CommandNotFoundException: The term 'Get-azvm' is not recognized as the name of a cmdlet, function, script file, or operable program.

**Figura 4.28: verificando erros**

- Instale o módulo **Az** usando o seguinte comando no servidor:

```
Install-module -name Az -AllowClobber -verbose
```

Também é importante ter o certificado **.pfx** disponível neste servidor. O certificado anteriormente exportado deve ser copiado para o servidor e instalado manualmente.

- Após a instalação do módulo Az e do certificado, a nova execução do runbook no Hybrid Worker é mostrada na Figura 4.29 e deve mostrar a lista de VMs na assinatura:

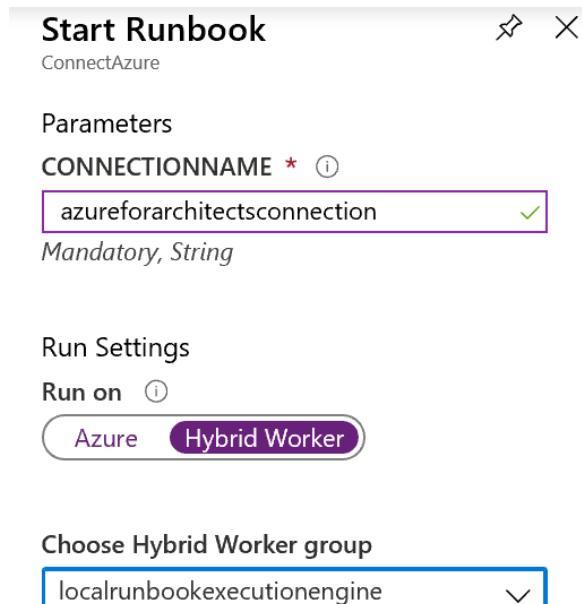


Figura 4.29: configurando um runbook para ser executado em um Hybrid Worker

Quando abordamos diferentes cenários, falamos sobre gerenciamento de configuração. Na próxima seção, abordaremos mais detalhadamente o gerenciamento de configuração com a Automação do Azure.

## Configuração de estado da Automação do Azure

A Automação do Azure fornece um servidor de pull do **Desired State Configuration (DSC)** juntamente com todas as contas da Automação do Azure. O servidor de pull pode armazenar scripts de configuração que podem ser extraídos por servidores em nuvens e na infraestrutura local. Isso significa que a Automação do Azure pode ser usada para configurar qualquer servidor hospedado em qualquer lugar do mundo.

O DSC precisa de um agente local nesses servidores, também conhecido como **gerenciador de configurações local (LCM)**. Ele deve ser configurado com o servidor de pull do DSC da Automação do Azure para que possa fazer o download da configuração necessária e configurar o servidor automaticamente.

A configuração automática pode ser programada para ser periódica (por padrão, é meia hora), e se o agente encontrar algum desvio na configuração do servidor em comparação com a disponível no script do DSC, ele corrigirá e trará o servidor de volta para o estado desejado e esperado.

Nesta seção, configuraremos um servidor hospedado no Azure, e o processo permanecerá o mesmo, independentemente de o servidor estar em uma nuvem ou na infraestrutura local.

A primeira etapa é criar uma configuração do DSC. Uma configuração de exemplo é mostrada aqui, e configurações complexas podem ser criadas da mesma forma:

```
configuration ensureiis {
    import-dscresource -modulename psdesiredstateconfiguration

    node localhost {
        WindowsFeature iis {
            Name = "web-server"
            Ensure = "Present"

        }
    }
}
```

A configuração é bastante simples. Ela importa o módulo do DSC de base **PSDesiredStateConfiguration** e declara uma configuração de nó único. Essa configuração não é associada a nenhum nó específico e pode ser usada para configurar qualquer servidor. A configuração deve configurar um servidor da Web do IIS e garantir que ele esteja presente em qualquer servidor ao qual seja aplicado.

Essa configuração ainda não está disponível no servidor de pull do DSC da Automação do Azure e, portanto, a primeira etapa é importar a configuração para o servidor de pull. Isso pode ser feito usando o cmdlet **Import-AzAutomationDscConfiguration** da conta da Automação, conforme mostrado a seguir:

```
Import-AzAutomationDscConfiguration -SourcePath "C:\Ritesh\ensureiis.ps1"
-AutomationAccountName bookaccount -ResourceGroupName automationrg -Force
-Published
```

Há algumas coisas importantes a serem observadas aqui. O nome da configuração deve corresponder ao nome de arquivo e deve conter apenas caracteres alfanuméricos e sublinhados. Uma convenção de nomenclatura recomendável é usar combinações de verbos/substantivos. Os cmdlets precisam do caminho do arquivo de configuração e dos detalhes da conta da Automação do Azure para importar o script de configuração.

Nesta fase, a configuração é visível no portal:

The screenshot shows the Azure portal interface for an Automation Account named 'bookaccount'. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration Management (with Inventory and Change tracking), and State configuration (DSC). The main area shows the 'State configuration (DSC)' blade. At the top, there's a search bar, an 'Add' button, and several filter buttons: 'Compose configuration', 'Refresh', and 'Reset filters'. Below that, tabs include 'Nodes', 'Configurations' (which is selected and highlighted with a blue border), 'Compiled configurations', and 'Gallery'. A 'Configuration' section with a search bar follows. The main table lists one configuration: 'ensureiis' with a 'Compiled Configuration Count' of 0 and a 'Last Modified' date of 3/26/2020, 8:34 AM.

Configuration	Compiled Configuration Count	Last Modified
ensureiis	0	3/26/2020, 8:34 AM

Figura 4.30: adicionando configuração

Depois que o script de configuração for importado, ele será compilado e armazenado no servidor de pull do DSC usando o cmdlet **Start-AzAutomationDscCompilationJob**, conforme mostrado a seguir:

```
Start-AzAutomationDscCompilationJob -ConfigurationName 'ensureiis'
-ResourceGroupName 'automationrg' -AutomationAccountName 'bookaccount'
```

O nome da configuração deve ser o mesmo que foi carregado recentemente, e a configuração compilada deve estar disponível agora na guia **Configurações compiladas**, conforme mostrado na Figura 4.31:

The screenshot shows the 'Compiled configurations' tab for the same 'bookaccount' Automation Account. The interface is similar to Figure 4.30, with the 'Compiled configurations' tab now selected. The table lists one compiled configuration: 'ensureiis' from 'ensureiis.localhost' with a node count of 0, created on 3/27/2020, 4:09 AM, and last modified on 3/27/2020, 4:09 AM.

Node Configuration	Configuration	Node Count	Created	Last Modified
ensureiis.localhost	ensureiis	0	3/27/2020, 4:09 AM	3/27/2020, 4:09 AM

Figura 4.31: listando configurações compiladas

É importante observar que a **Contagem de nós** na Figura 4.31 é 0. Isso significa que uma configuração de nó chamada `ensureiss.localhost` existe, mas não é atribuída a nenhum nó. A próxima etapa é atribuir a configuração ao nó.

Agora, temos uma configuração do DSC compilada disponível no servidor de pull do DSC, mas não há nós para gerenciar. A próxima etapa é integrar as VMs e associá-las ao servidor de pull do DSC. Isso é feito usando o cmdlet `Register-AzAutomationDscNode`:

```
Register-AzAutomationDscNode -ResourceGroupName 'automationrg'
    -AutomationAccountName 'bookaccount' -AzureVMLocation "west
    Europe" -AzureVMResourceGroup 'spark' -AzureVMName 'spark'
    -ConfigurationModeFrequencyMins 30 -ConfigurationMode 'ApplyAndAutoCorrect'
```

Esse cmdlet leva o nome do grupo de recursos para a VM e a conta da Automação do Azure. Ele também configura o modo de configuração e a propriedade **configurationModeFrequencyMins** do gerenciador de configurações local da VM. Essa configuração verificará e corrigirá automaticamente qualquer desvio da configuração aplicada a ele a cada 30 minutos.

Se **VMresourcegroup** não for especificado, o cmdlet tentará encontrar a VM no mesmo grupo de recursos que a conta da Automação do Azure e, se o valor do local da VM não for fornecido, ele tentará encontrar a VM na região da Automação do Azure. É sempre melhor fornecer valores para eles. Observe que esse comando só pode ser usado para VMs do Azure, pois ele solicita **AzureVMname** explicitamente. Para servidores em outras nuvens e na infraestrutura local, use o cmdlet `Get-AzAutomationDscOnboardingMetaconfig`.

Agora, uma nova entrada de configuração de nó também pode ser encontrada no portal, conforme mostrado na Figura 4.32:

Node	Status	Node configuration	Last seen	Version
spark	Compliant		3/27/2020, 4:38 AM	2.80.0.0

Figura 4.32: verificando o status do nó

As informações do nó podem ser obtidas da seguinte maneira:

```
$node = Get-AzAutomationDscNode -ResourceGroupName 'automationrg'
-AutomationAccountName 'bookaccount' -Name 'spark'
```

E uma configuração pode ser atribuída ao nó:

```
Set-AzAutomationDscNode -ResourceGroupName 'automationrg'
-AutomationAccountName 'bookaccount' -NodeConfigurationName 'ensureiis.
localhost' -NodeId $node.Id
```

Depois que a compilação for concluída, ela poderá ser atribuída aos nós. O status inicial é **Pendente**, conforme mostrado na Figura 4.33:

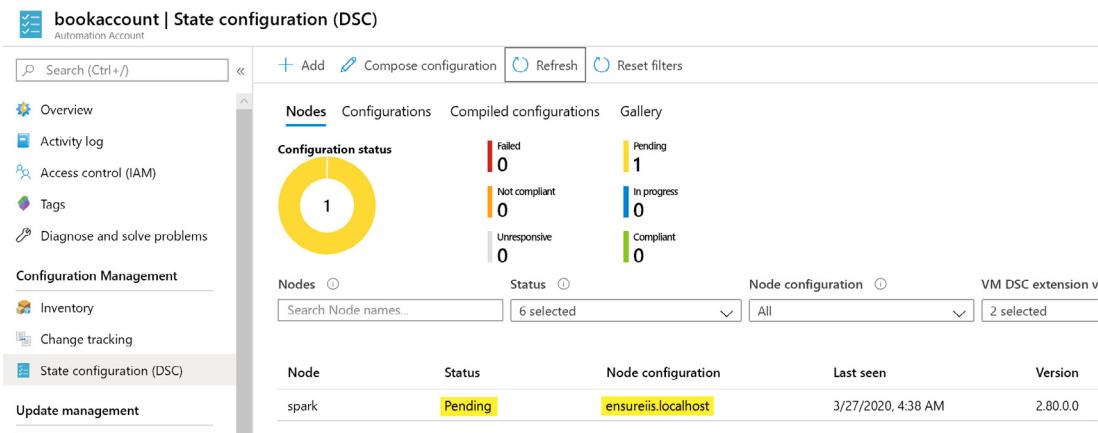


Figura 4.33: verificando o status do nó

Depois de alguns minutos, a configuração é aplicada ao nó, o nó torna-se **Compatível**, e o status se tornará **Concluído**:

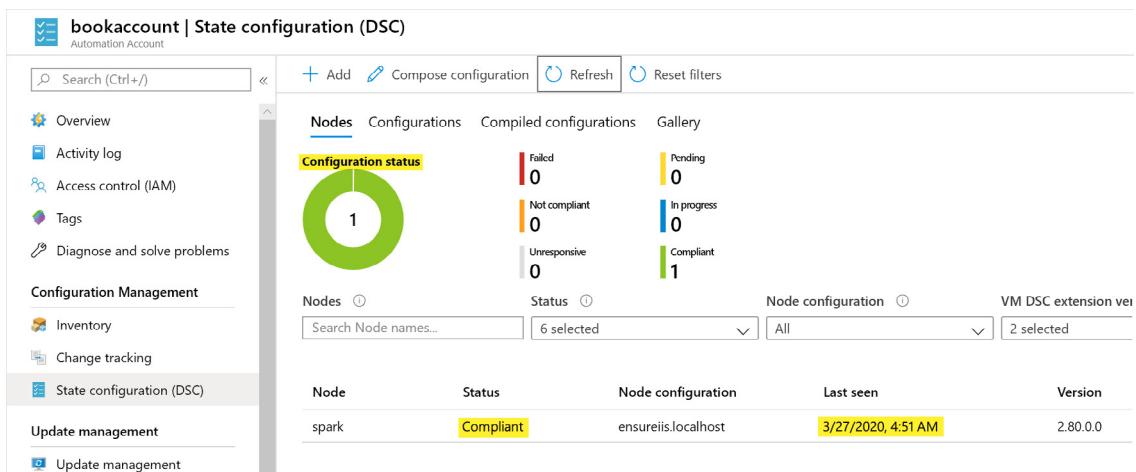


Figura 4.34: verificando se o nó é compatível

Mais tarde, ao fazer logon no servidor e verificar se o servidor Web (IIS) está instalado, você confirmará que ele está instalado, como pode ver na Figura 4.35:

PS C:\Users\citynextadmin> Get-WindowsFeature -Name web-server		
Display Name	Name	Install State
[X] Web Server (IIS)	Web-Server	Installed

Figura 4.35: verificando se o estado desejado foi alcançado

Na próxima seção, os preços da Automação do Azure serão abordados.

## Preços da Automação do Azure

Não há nenhum custo para a Automação do Azure se nenhum runbook for executado nela. O custo da Automação do Azure é cobrado por minuto pela execução de trabalhos de runbooks. Isso significa que, se o número total de minutos de execução de runbooks for 10.000, o custo da Automação do Azure será US\$ 0,002 por minuto multiplicado por 9.500, pois os primeiros 500 minutos são gratuitos.

Há outros custos envolvidos na Automação do Azure, dependendo dos recursos consumidos. Por exemplo, um servidor de pull do DSC não custa nada na Automação do Azure, nem a integração de VMs do Azure no servidor de pull. No entanto, se os servidores que não são do Azure são integrados, normalmente de outras nuvens ou da infraestrutura local, os cinco primeiros servidores são gratuitos e qualquer item além disso custa US\$ 6 por servidor por mês na região Oeste dos EUA.

Os preços podem variar de acordo com a região, e é sempre recomendável verificar os preços na página oficial de preços: <https://azure.microsoft.com/pricing/details/automation>.

Você pode perguntar: por que precisamos de uma conta da Automação quando podemos implantar aplicações sem servidor por meio do Azure Functions? Na próxima seção, exploraremos as principais diferenças entre a Automação do Azure e a automação sem servidor.

## Comparação com a automação sem servidor

A Automação do Azure e as tecnologias sem servidor do Azure, principalmente o Azure Functions, são bastante semelhantes e se sobrepõem em termos de funcionalidade. No entanto, esses são serviços separados com diferentes recursos e preços.

É importante entender que a Automação do Azure é um pacote completo para gerenciamento de configuração e automação de processos, enquanto o Azure Functions destina-se a implementar a funcionalidade de negócios.

A Automação do Azure é usada para automatizar os processos de provisionamento, desprovisionamento, gerenciamento e operações de gerenciamento de infraestrutura e configuração depois disso. Por outro lado, o Azure Functions destina-se à criação de serviços, implementando funcionalidades que podem fazer parte de microsserviços e outras APIs.

A Automação do Azure não se destina à escala ilimitada, e espera-se que a carga seja moderada, enquanto o Azure Functions pode lidar com tráfego ilimitado e escalar automaticamente.

Há uma série de ativos compartilhados, como conexões, variáveis e módulos, que podem ser reutilizados em runbooks na Automação do Azure. No entanto, não há um conceito compartilhado pronto para uso no Azure Functions.

A Automação do Azure pode gerenciar o estado intermediário por meio da definição do ponto de verificação e continuar a partir do último estado salvo, enquanto as funções do Azure geralmente não têm estado e não mantêm nenhum estado.

## Resumo

A Automação do Azure é um serviço importante no Azure e o único serviço de gerenciamento de configurações e automação de processos. Este capítulo abordou vários conceitos importantes relacionados à Automação do Azure e à automação de processos, incluindo ativos compartilhados, como conexão, certificados e módulos.

Ele abordou a criação de runbooks, incluindo a invocação de runbooks de formas diferentes, como relações entre pais e filhos, webhooks e uso do portal. O capítulo também abordou a arquitetura e o ciclo de vida dos runbooks.

Também examinamos o uso de Hybrid Workers e, no final do capítulo, exploramos o gerenciamento de configurações usando um servidor de pull do DSC e um gerenciador de configurações local. Por fim, fizemos comparações com outras tecnologias, como o Azure Functions.

No próximo capítulo, exploraremos a criação de políticas, bloqueios e marcas para implantações do Azure.





# 5

## Criar políticas, bloqueios e marcas para implantações do Azure

O Azure é uma plataforma de nuvem versátil. Os clientes podem não apenas criar e implantar suas aplicações, mas também gerenciar e governar seus ambientes ativamente. As nuvens geralmente seguem um paradigma pré-pago no qual um cliente assina e pode implantar praticamente qualquer coisa nelas. Desde uma máquina virtual básica até milhares de máquinas virtuais com **unidades de manutenção de estoque (SKUs)** maiores. O Azure não impedirá que nenhum cliente provisione os recursos que deseja provisionar. Em uma organização, pode haver um grande número de pessoas com acesso à assinatura do Azure da organização. Há uma necessidade de haver um modelo de governança em vigor para que somente os recursos necessários sejam provisionados por pessoas com o direito de criá-los. O Azure fornece funções de gerenciamento de recursos, como o **Controle de acesso baseado em função do Azure (RBAC)**, o Azure Policy, grupos de gerenciamento, Blueprints e bloqueios de recursos para gerenciar e fornecer governança para os recursos.

Outros aspectos importantes da governança incluem o gerenciamento de custos, uso e informações. A equipe de gerência de uma organização sempre quer se manter atualizada sobre o consumo e os custos de nuvem. Ela quer identificar a porcentagem de uso do custo total por parte de equipes, departamentos e unidades. Em suma, ela quer ter relatórios com base em várias dimensões de consumo e custo. O Azure fornece um recurso de marcação que pode ajudar a fornecer esse tipo de informação em tempo real.

Neste capítulo, abordaremos os seguintes tópicos:

- Grupos de gerenciamento do Azure
- Marcas do Azure
- Azure Policy
- Bloqueios do Azure
- Azure RBAC
- Azure Blueprints
- Implementação de recursos de governança do Azure

## Grupos de gerenciamento do Azure

Estamos começando com grupos de gerenciamento do Azure porque, na maioria das próximas seções, vamos fazer referência ou mencionar grupos de gerenciamento. Os grupos de gerenciamento atuam como um nível de escopo para que você atribua ou gerencie funções e políticas com eficiência. Os grupos de gerenciamento são muito úteis se você tem várias assinaturas.

Os grupos de gerenciamento atuam como um espaço reservado para organizar assinaturas. Você também pode ter grupos de gerenciamento aninhados. Se você aplicar uma política ou acesso no nível do grupo de gerenciamento, ele será herdado pelos grupos de gerenciamento subjacentes e pelas assinaturas. No nível da assinatura, essa política ou acesso será herdado por grupos de recursos e, por fim, pelos recursos.

A hierarquia de grupos de gerenciamento é mostrada aqui:

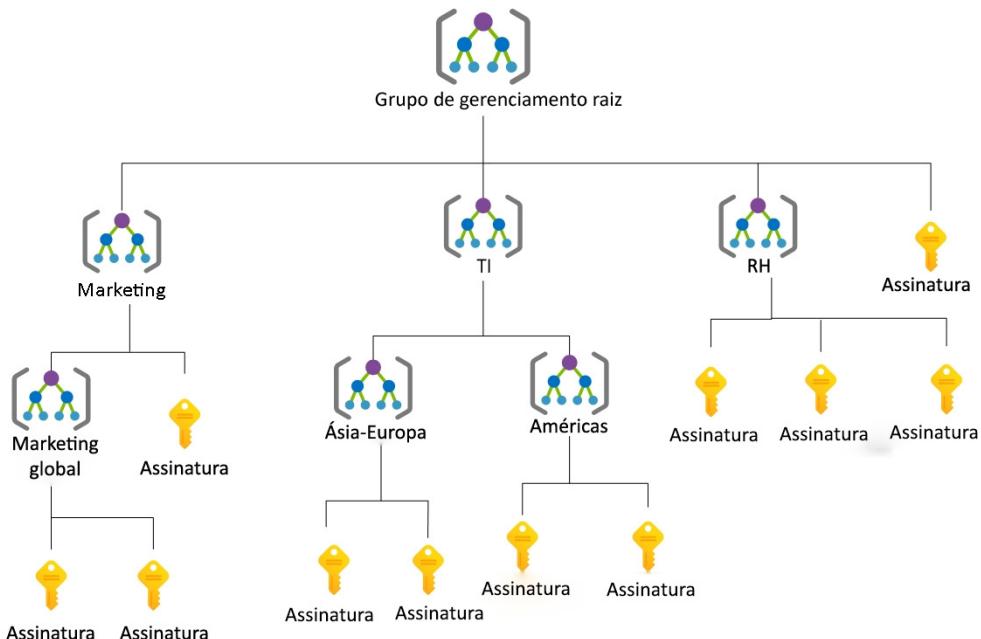


Figura 5.1: hierarquia de grupos de gerenciamento do Azure

Na Figura 5.1, estamos usando grupos de gerenciamento para separar as operações de diferentes departamentos, como marketing, TI e RH. Em cada um desses departamentos, há assinaturas e grupos de gerenciamento aninhados, o que ajuda a organizar os recursos em uma hierarquia para gerenciamento de políticas e acesso. Mais tarde, você verá como os grupos de gerenciamento são usados como um escopo para governança, gerenciamento de políticas e gerenciamento de acesso.

Na próxima seção, abordaremos as marcas do Azure, que desempenham outra função essencial no agrupamento lógico de recursos.

## Marcas do Azure

O Azure permite a marcação de recursos e grupos de recursos com pares nome-valor. A marcação ajuda na categorização e organização lógica dos recursos. O Azure também permite a marcação de 50 pares nome-valor para um grupo de recursos e seus recursos. Embora um grupo de recursos funcione como um contêiner ou um espaço reservado para recursos, a marcação de um grupo de recursos não significa a marcação dos recursos nele. Os grupos de recursos e os recursos devem ser marcados com base em seu uso, o que será explicado mais adiante nesta seção. As marcas são vinculadas a uma assinatura, um grupo de recursos ou um recurso. O Azure aceita qualquer par nome-valor e, por isso, é importante para uma organização definir os nomes e seus possíveis valores.

Mas por que a marcação é importante? Em outras palavras, quais problemas podem ser resolvidos usando a marcação? A marcação tem os seguintes benefícios:

- **Categorização de recursos:** uma assinatura do Azure pode ser usada por vários departamentos e funções em uma organização. É importante para a equipe de gerência identificar os proprietários de quaisquer recursos. A marcação ajuda na atribuição de identificadores a recursos que possam ser usados representar departamentos ou funções.
- **Gerenciamento de informações para recursos do Azure:** novamente, os recursos do Azure em uma assinatura podem ser provisionados por qualquer pessoa que tenha acesso a ela. As organizações desejam ter uma categorização adequada dos recursos em vigor para atender às políticas de gerenciamento de informações. Essas políticas podem ser baseadas no gerenciamento do ciclo de vida da aplicação, como o gerenciamento dos ambientes de desenvolvimento, teste e produção. Elas também podem ser baseadas no uso ou em quaisquer outras prioridades. Cada organização tem sua própria maneira de definir as categorias de informações, e o Azure atende a essa necessidade com as marcas.
- **Gerenciamento de custos:** a marcação no Azure pode ajudar a identificar recursos com base na sua categorização. As consultas podem ser executadas no Azure para identificar o custo por categoria, por exemplo. Por exemplo, o custo dos recursos no Azure para o desenvolvimento de um ambiente para o departamento financeiro e o departamento de marketing pode ser facilmente verificado. Além disso, o Azure também fornece informações de cobrança com base em marcas. Isso ajuda a identificar as taxas de consumo de equipes, departamentos ou grupos.

As marcas no Azure têm certas limitações, no entanto:

- O Azure permite que um máximo de 50 pares nome-valor de marca sejam associados a grupos de recursos.
- Marcas não são herdáveis. As marcas aplicadas a um grupo de recursos não se aplicam aos recursos individuais dentro dele. No entanto, é muito fácil esquecer de marcar recursos ao provisioná-los. O Azure Policy fornece o mecanismo a ser usado para garantir que as marcas sejam marcadas com o valor apropriado durante o tempo de provisionamento. Vamos considerar os detalhes dessas políticas mais adiante neste capítulo.

As marcas podem ser atribuídas a recursos e grupos de recursos usando o PowerShell, a CLI do Azure 2.0, modelos do Azure Resource Manager, o portal do Azure e as APIs REST do Azure Resource Manager.

Um exemplo de categorização de gerenciamento de informações usando as marcas do Azure é mostrado aqui:



Figura 5.2: categorização de gerenciamento de informações usando marcas do Azure

Neste exemplo, os pares nome-valor de **Departamento**, **Projeto**, **Ambiente**, **Proprietário**, **Aprovador**, **Mantenedor**, **Data de início**, **Data de desativação** e **Data de correção** são usados para marcar recursos. É muito fácil encontrar todos os recursos para uma marca específica ou uma combinação de marcas usando o PowerShell, a CLI do Azure ou as APIs REST: A próxima seção abordará maneiras de usar o PowerShell para atribuir marcas a recursos.

## Marcas com o PowerShell

As marcas podem ser gerenciadas usando o PowerShell, os modelos do Azure Resource Manager, o portal do Azure e as APIs REST. Nesta seção, o PowerShell será usado para criar e aplicar marcas. O PowerShell fornece um cmdlet para recuperar e anexar marcas a recursos e grupos de recursos:

- Para recuperar marcas associadas a um recurso usando o PowerShell, o cmdlet **Get-AzResource** pode ser usado:

```
(Get-AzResource -Tag @{"Environment"="Production"}).Name
```

- Para recuperar marcas associadas a um grupo de recursos usando o PowerShell, o seguinte comando pode ser usado:

```
Get-AzResourceGroup -Tag @{"Environment"="Production"}
```

- Para definir marcas para um grupo de recursos, o cmdlet **Update-AzTag** pode ser usado:

```
$tags = @{"Dept"="IT"; "Environment"="Production"}  
$resourceGroup = Get-AzResourceGroup -Name demoGroup  
New-AzTag -ResourceId $resourceGroup.ResourceId -Tag $tags
```

- Para definir marcas para um recurso, o mesmo cmdlet **Update-AzTag** pode ser usado:

```
$tags = @{"Dept"="Finance"; "Status"="Normal"}  
$resource = Get-AzResource -Name demoStorage -ResourceGroup demoGroup  
New-AzTag -ResourceId $resource.id -Tag $tags
```

- Você pode atualizar as marcas existentes usando o comando **Update-AzTag**. No entanto, você precisa especificar a operação como **Mesclagem** ou **Substituição**. A **Mesclagem** acrescentará as novas marcas que você está passando para as marcas existentes. No entanto, a operação **Substituição** substituirá todas as marcas antigas por novas. Veja a seguir um exemplo de atualização das marcas em um grupo de recursos sem substituir as existentes:

```
$tags = @{"Dept"="IT"; "Environment"="Production"}  
$resourceGroup = Get-AzResourceGroup -Name demoGroup  
Update-AzTag -ResourceId $resourceGroup.ResourceId -Tag $tags -Operation  
Merge
```

Agora, vamos ver as marcas com modelos do Azure Resource Manager.

## Marcas com modelos do Azure Resource Manager

Os modelos do Azure Resource Manager também fornece ajuda na definição de marcas para cada recurso. Eles podem ser usados para atribuir várias marcas a cada recurso, desta forma:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/  
deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "resources": [  
        {  
            "apiVersion": "2019-06-01",  
            "type": "Microsoft.Storage/storageAccounts",  
            "name": "[concat('storage', uniqueString(resourceGroup().id))]",  
            "location": "[resourceGroup().location]",  
            "tags": {  
                "Dept": "Finance",  
                "Environment": "Production"  
            },  
            "sku": {  
                "name": "Standard_LRS"  
            },  
            "kind": "Storage",  
            "properties": { }  
        }  
}
```

No exemplo anterior, duas marcas, **Departamento** e **Ambiente**, foram adicionados a um recurso de conta de armazenamento usando modelos do Azure Resource Manager.

## Marcar grupos de recursos versus recursos

É imprescindível que arquitetos decidam a arquitetura de informações e taxonomia para recursos e grupos de recursos do Azure. Eles devem identificar as categorias pelas quais os recursos serão classificados com base nos requisitos da consulta. No entanto, eles também devem identificar se as marcas devem ser anexadas a recursos individuais ou a grupos de recursos.

Se todos os recursos dentro de um grupo de recursos precisarem da mesma marca, é melhor marcar o grupo de recursos, embora as marcas não herdem os recursos no grupo de recursos. Se sua organização exigir que as marcas sejam passadas para todos os recursos subjacentes, você poderá considerar escrever um script do PowerShell para obter as marcas do grupo de recursos e atualizar as marcas para os recursos no grupo de recursos. É importante considerar as consultas nas marcas antes de decidir se as marcas devem ser aplicadas no nível do recurso ou do grupo de recursos. Se as consultas se relacionarem a tipos de recursos individuais em uma assinatura e em grupos de recursos, a atribuição de marcas a recursos individuais fará mais sentido. No entanto, se a identificação de grupos de recursos for suficiente para que suas consultas sejam eficazes, as marcas deverão ser aplicadas somente aos grupos de recursos. Se você estiver migrando recursos entre grupos de recursos, as marcas aplicadas no nível do grupo de recursos serão perdidas. Se você estiver migrando recursos, considere adicionar as marcas novamente.

## Azure Policy

Na seção anterior, falamos sobre a aplicação de marcas para implantações do Azure. As marcas são ótimas para organizar recursos; no entanto, há algo mais que não foi discutido: como as organizações garantem que as marcas sejam aplicadas para cada implantação? Deve haver uma imposição automática das marcas do Azure a recursos e grupos de recursos. Não há nenhuma verificação do Azure para garantir que marcas apropriadas sejam aplicadas a recursos e grupos de recursos. Isso não é apenas específico das marcas; aplica-se à configuração de qualquer recurso no Azure. Por exemplo, você pode querer restringir onde seus recursos podem ser provisionados geograficamente (somente para a região Leste dos EUA, por exemplo).

Talvez você já tenha percebido que esta seção é sobre a formulação de um modelo de governança no Azure. A governança é um elemento importante no Azure porque garante que todos que acessarem o ambiente do Azure estejam cientes dos processos e das prioridades organizacionais. Ela também ajuda a deixar os custos sob controle. Ela ajuda na definição das convenções organizacionais para o gerenciamento de recursos.

Cada política pode ser criada usando várias regras, e várias políticas podem ser aplicadas a uma assinatura ou a um grupo de recursos. Se as regras forem atendidas, as políticas poderão executar várias ações. Uma ação poderia ser negar uma transação em andamento, auditar (uma transação o que significa gravar em logs e permitir sua conclusão) ou acrescentar metadados a uma transação se eles estiverem ausentes.

As políticas podem estar relacionadas à convenção de nomenclatura de recursos, marcação de recursos, tipos de recursos que podem ser provisionados, localização de recursos ou qualquer combinação delas.

O Azure fornece várias políticas internas, e é possível criar políticas personalizadas. Há uma linguagem de política baseada em JSON que pode ser usada para definir políticas personalizadas.

Agora que você conhece a finalidade e o caso de uso do Azure Policy, vamos avançar e abordar políticas internas, a linguagem de políticas e políticas personalizadas.

## Políticas internas

O Azure fornece um serviço para criar políticas personalizadas. No entanto, ele também fornece algumas políticas prontas para uso que podem ser usadas para a governança. Essas políticas estão relacionadas a localizações permitidas, tipos de recursos permitidos e marcas. Mais informações sobre essas políticas internas podem ser encontradas em <https://docs.microsoft.com/azure/azure-resource-manager/resource-manager-policy>.

## Linguagem de políticas

As políticas no Azure usam a linguagem JSON para definir e descrever as políticas. Há duas etapas na adoção de políticas. A política deve ser definida e, em seguida, deve ser aplicada e atribuída. As políticas têm escopo e podem ser aplicadas no nível da assinatura, do grupo de recursos e do grupo de gerenciamento.

As políticas são definidas usando blocos **if...then** que são semelhantes a qualquer linguagem de programação popular. O bloco **if** é executado para avaliar as condições e, com base no resultado dessas condições, o bloco **then** é executado:

```
{  
  "if": {  
    <condition> | <logical operator>  
  },  
  "then": {  
    "effect": "deny | audit | append"  
  }  
}
```

As políticas do Azure não só permitem condições **if** simples, mas várias condições **if** podem ser associadas logicamente para criar regras complexas. Essas condições podem ser associadas usando operadores **AND**, **OR** e **NOT**:

- A sintaxe **AND** exige que todas as condições sejam verdadeiras.
- A sintaxe **OR** exige que uma das condições seja verdadeira.
- A sintaxe **NOT** inverte o resultado da condição.

A sintaxe **AND** é mostrada a seguir. Ela é representada pela palavra-chave **allOf**:

```
"if": {  
  "allOf": [  
    {  
      "field": "tags",  
      "containsKey": "application"  
    },  
    {  
      "field": "type",  
      "equals": "Microsoft.Storage/storageAccounts"  
    }  
  ]  
},
```

A sintaxe **OR** é mostrada em seguida. Ela é representada pela palavra-chave **anyOf**:

```
"if": {  
  "anyOf": [  
    {  
      "field": "tags",  
      "containsKey": "application"  
    },  
    {  
      "field": "type",  
      "equals": "Microsoft.Storage/storageAccounts"  
    }  
  ]  
},
```

A sintaxe **NOT** é mostrada a seguir. Ela é representada pela palavra-chave **not**:

```
"if": {  
  "not": [  
    {  
      "field": "tags",  
    }  
  ]  
},
```

```

    "containsKey": "application"
},
{
  "field": "type",
  "equals": "Microsoft.Storage/storageAccounts"
}
]
},

```

Na verdade, esses operadores lógicos podem ser combinados da seguinte maneira:

```

"if": {
  "allOf": [
    {
      "not": {
        "field": "tags",
        "containsKey": "application"
      }
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},

```

Isso é muito semelhante ao uso das condições **if** em linguagens de programação populares como C# e Node.js

```

If ("type" == "Microsoft.Storage/storageAccounts") {
  Deny
}

```

É importante observar que não há uma ação **allow**, embora haja uma ação **Deny**. Isto significa que as regras políticas devem ser escritas com a possibilidade de negação na mente. As regras devem avaliar as condições e **Negar** a ação se as condições forem atendidas.

## Campos permitidos

Os campos permitidos nas condições ao definir as políticas são os seguintes:

- **Nome:** o nome do recurso para aplicar a política. Isso é muito específico e aplicável a um recurso de acordo com seu uso.
- **Tipo:** o tipo de recurso, como **Microsoft.Compute/VirtualMachines**. Por exemplo, isso aplicaria a política a todas as instâncias de máquinas virtuais.
- **Localização:** a localização (ou seja, a região do Azure) de um recurso.
- **Marcas:** as marcas associadas a um recurso.
- **Aliases de propriedade:** propriedades específicas de recursos. Essas propriedades são diferentes para diferentes recursos.

Na próxima seção, você aprenderá mais sobre como proteger recursos em ambientes de produção.

## Bloqueios do Azure

Os bloqueios são mecanismos para interromper determinadas atividades nos recursos. O RBAC fornece direitos a usuários, grupos e aplicações em um determinado escopo. Há funções RBAC prontas para uso, como proprietário, colaborador e leitor. Com a função de colaborador, é possível excluir ou modificar um recurso. Como tais atividades podem ser impedidas apesar do usuário ter a função de colaborador? Insira bloqueios do Azure.

Os bloqueios do Azure podem ajudar de duas maneiras:

- Eles podem bloquear recursos de modo que eles não possam ser excluídos, mesmo que você tenha acesso de proprietário.
- Eles podem bloquear recursos de forma que não possam ser excluídos nem ter sua configuração modificada.

Os bloqueios normalmente são muito úteis para os recursos em ambientes de produção que não devem ser modificados nem excluídos acidentalmente.

Os bloqueios podem ser aplicados nos níveis da assinatura, do grupo de recursos, do grupo de gerenciamento e do recurso individual. Os bloqueios podem ser herdados entre assinaturas, grupos de recursos e recursos. A aplicação de um bloqueio no nível pai garantirá que esses recursos no nível filho também o herdarão. Os recursos adicionados posteriormente no sub-escopo também herdam a configuração de bloqueio por padrão. A aplicação de um bloqueio no nível do recurso também impedirá a exclusão do grupo de recursos que contenha o recurso.

Os bloqueios são aplicados somente a operações que ajudam no gerenciamento de um recurso, em vez daquelas que são internas a ele. Os usuários precisam das permissões de RBAC `Microsoft.Authorization/*` ou `Microsoft.Authorization/locks/*` para criar e modificar bloqueios.

Os bloqueios podem ser criados e aplicados usando o portal do Azure, o Azure PowerShell, a CLI do Azure, os modelos do Azure Resource Manager e APIs REST.

A criação de um bloqueio usando um modelo do Azure Resource Manager é feita da seguinte maneira:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "lockedResource": {  
            "type": "string"  
        }  
    },  
    "resources": [  
        {  
            "name": "[concat(parameters('lockedResource'), '/Microsoft.  
Authorization/myLock')]",  
            "type": "Microsoft.Storage/storageAccounts/providers/locks",  
            "apiVersion": "2019-06-01",  
            "properties": {  
                "level": "CannotDelete"  
            }  
        }  
    ]  
}
```

A seção de **recursos** do código de modelo do Azure Resource Manager consiste em uma lista de todos os recursos a serem provisionados ou atualizados no Azure. Há um recurso de conta de armazenamento, e a conta de armazenamento tem um recurso de bloqueio. Um nome para o bloqueio é fornecido usando a concatenação de cadeias dinâmicas, e o bloqueio que é aplicado é do tipo `CannotDelete`, o que significa que a conta de armazenamento está bloqueada para exclusão. A conta de armazenamento só pode ser excluída após a remoção do bloqueio.

A criação e a aplicação de um bloqueio a um recurso usando o PowerShell são feitas da seguinte forma:

```
New-AzResourceLock -LockLevel CanNotDelete -LockName LockSite '  
    -ResourceName examplesite -ResourceType Microsoft.Web/sites '  
    -ResourceGroupName exempleresourcegroup
```

A criação e a aplicação de um bloqueio a um grupo de recursos usando o PowerShell são feitas da seguinte forma:

```
New-AzResourceLock -LockName LockGroup -LockLevel CanNotDelete '  
    -ResourceGroupName exempleresourcegroup
```

A criação e a aplicação de um bloqueio a um recurso usando a CLI do Azure são feitas da seguinte forma:

```
az lock create --name LockSite --lock-type CanNotDelete \  
    --resource-group exempleresourcegroup --resource-name examplesite \  
    --resource-type Microsoft.Web/sites
```

A criação e a aplicação de um bloqueio a um grupo de recursos usando a CLI do Azure são feitas como a seguir:

```
az lock create --name LockGroup --lock-type CanNotDelete \  
    --resource-group exempleresourcegroup
```

Para criar ou excluir bloqueios de recursos, o usuário deve ter acesso às ações **Microsoft.Authorization/\*** ou **Microsoft.Authorization/locks/\***. Você também pode dar permissões granulares. Os proprietários e os administradores de acesso do usuário terão acesso à criação ou exclusão de bloqueios por padrão.

Se você quer saber quais são as palavras-chave de **Microsoft.Authorization/\*** e **Microsoft.Authorization/locks/\***, você saberá mais sobre elas na próxima seção.

Agora, vamos ver o Azure RBAC.

## Azure RBAC

O Azure fornece autenticação para seus recursos por meio do Azure Active Directory. Uma vez que uma identidade tenha sido autenticada, os recursos a que a identidade terá acesso deverão ser decididos. Isso é conhecido como autorização. A autorização avalia as permissões que foram oferecidas a uma identidade. Qualquer pessoa que tenha acesso a uma assinatura do Azure deve receber apenas as permissões necessárias para que seu trabalho específico possa ser realizado, e nada mais.

A autorização também é popularmente conhecida como RBAC. O RBAC no Azure se refere à atribuição de permissões a identidades em um escopo. O escopo pode ser um grupo de gerenciamento, uma assinatura, um grupo de recursos ou recursos individuais.

O RBAC ajuda na criação e na atribuição de permissões diferentes a identidades diferentes. Isso ajuda a segregar as tarefas dentro das equipes, em vez de todos terem todas as permissões. O RBAC ajuda a tornar as pessoas responsáveis somente pelo seu trabalho, pois outras talvez nem tenham o acesso necessário para realizá-lo. Deve-se observar que fornecer permissões a um escopo superior automaticamente garante que os recursos filho herdem essas permissões. Por exemplo, fornecer uma identidade com acesso de leitura a um grupo de recursos significa que a identidade terá acesso de leitura a todos os recursos dentro desse grupo também.

O Azure fornece três funções internas de uso geral. São eles:

- A função de proprietário, que tem acesso total a todos os recursos
- A função de colaborador, que tem acesso para leitura/gravação de recursos
- A função de leitor, que tem somente permissões de leitura sobre recursos

Há mais funções fornecidas pelo Azure, mas elas são específicas de recursos, como as funções de colaborador de rede e gerenciador de segurança.

Para obter todas as funções fornecidas pelo Azure para todos os recursos, execute o comando **Get-AzRoleDefinition** no console do PowerShell.

Cada definição de função tem determinadas ações permitidas e não permitidas. Por exemplo, a função de proprietário tem todas as ações permitidas e nenhuma ação é proibida:

```
PS C:\Users\riskaria> Get-AzRoleDefinition -Name "Owner"
Name          : Owner
Id            : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
IsCustom      : False
Description   : permite gerenciar tudo, incluindo o acesso a recursos.
Actions       : {*}
NotActions    : {}
DataActions   : {}
NotDataActions: {}
AssignableScopes : {/}
```

Cada função consiste em várias permissões. Cada recurso fornece uma lista de operações. As operações com o suporte de um recurso pode ser obtida usando o cmdlet **Get-AzProviderOperation**. Esse cmdlet obtém o nome do provedor e do recurso para recuperar as operações:

```
PS C:\Users\riskaria> Get-AzProviderOperation -OperationSearchString "Microsoft.Insights/*" | select Operation
```

Isso resultará na seguinte saída:

```
PS C:\Users\riskaria> Get-AzProviderOperation -OperationSearchString "Microsoft.Insights/*" | select Operation
```

## Operation

A saída mostrada aqui fornece todas as ações disponíveis no provedor de recursos **Microsoft. Insights** em seus recursos associados. Os recursos incluem **Métricas**, **Registro** e outros, enquanto as ações incluem **Leitura**, **Gravação** e outras.

Agora, vamos ver as funções personalizadas.

## Funções personalizadas

O Azure fornece inúmeras funções genéricas de saída prontas para uso, como proprietário, colaborador e leitor, bem como funções especializadas específicas de recursos, como colaborador de máquina virtual. Ter uma função de leitura atribuída a um usuário/grupo ou entidade de serviço significará que as permissões do leitor serão atribuídas ao escopo. O escopo pode ser um recurso, um grupo de recursos ou uma assinatura. Da mesma forma, um colaborador seria capaz de ler, bem como modificar o escopo atribuído. Um colaborador de máquina virtual seria capaz de modificar as configurações de máquina virtual, e não outras configurações de recursos. No entanto, há momentos em que as funções existentes podem não atender aos nossos requisitos. Nesses casos, o Azure permite a criação de funções personalizadas. Elas podem ser atribuídos a usuários, grupos e entidades de serviço e são aplicáveis a recursos, grupos de recursos e assinaturas.

As funções personalizadas são criadas por meio da combinação de várias permissões. Por exemplo, uma função personalizada pode consistir em operações de vários recursos. No próximo bloco de código, uma nova definição de função está sendo criada, mas, em vez de definir todas as propriedades manualmente, uma das funções existentes de "**Colaborador de máquina virtual**" é obtida porque quase coincide com a configuração da nova função personalizada. Evite usar o mesmo nome que as funções internas, pois isso criará conflito. Em seguida, a propriedade de ID é anulada, e um novo nome e descrição são fornecidos. O código também limpa todas as ações, adiciona algumas ações, adiciona um novo escopo após a limpeza do escopo existente e, por fim, cria uma nova função personalizada:

```
$role = Get-AzRoleDefinition "Virtual Machine Contributor"  
$role.Id = $null  
$role.Name = "Virtual Machine Operator"  
$role.Description = "Can monitor and restart virtual machines."  
$role.Actions.Clear()  
$role.Actions.Add("Microsoft.Storage/*/read")  
$role.Actions.Add("Microsoft.Network/*/read")  
$role.Actions.Add("Microsoft.Compute/*/read")  
$role.Actions.Add("Microsoft.Compute/virtualMachines/start/action")  
$role.Actions.Add("Microsoft.Compute/virtualMachines/restart/action")  
$role.Actions.Add("Microsoft.Authorization/*/read")  
$role.Actions.Add("Microsoft.Resources/subscriptions/resourceGroups/read")  
$role.Actions.Add("Microsoft.Insights/alertRules/*")  
$role.Actions.Add("Microsoft.Support/*")  
$role.AssivableScopes.Clear()  
$role.AssivableScopes.Add("/subscriptions/548f7d26-b5b1-468e-ad45-6ee12accf7e7")  
New-AzRoleDefinition -Role $role
```

Há um recurso de visualização disponível no portal do Azure que você pode usar para criar funções RBAC personalizadas do próprio portal do Azure. Você tem a opção de criar funções do zero, clonar uma função existente ou começar a escrever o manifesto JSON. A Figura 5.3 mostra a folha **Criar uma função personalizada**, que está disponível na seção IAM > +Adicionar:

Figura 5.3: criando funções personalizadas no portal do Azure

Isso simplifica o processo de criação de função personalizada.

## Como os bloqueios são diferentes do RBAC?

Os bloqueios não são iguais ao RBAC. O RBAC ajuda a permitir ou negar permissões para recursos. Essas permissões estão relacionadas à realização de operações como leitura, gravação e atualização nos recursos. Por outro lado, os bloqueios estão relacionados à reprovação de permissões para configurar ou excluir recursos.

Na próxima seção, abordaremos o Azure Blueprints, que ajuda com a orquestração de artefatos, como atribuições de função, atribuições de política e outros, que abordamos até agora.

## Azure Blueprints

Você estará familiarizado com a palavra "blueprint", que se refere ao plano ou projeto que é usado por um arquiteto para arquitetar uma solução. Da mesma forma, no Azure, os arquitetos de nuvem podem aproveitar o Azure Blueprints para definir um conjunto de recursos repetíveis do Azure que seguem as convenções, os processos e os padrões de uma organização.

O Blueprints permite orquestrar a implantação de vários recursos e outros artefatos, como:

- Atribuições de função
- Atribuições de política
- Modelos do Azure Resource Manager
- Grupos de recursos

Os objetos do Azure Blueprint são replicados para várias regiões e têm o suporte do Azure Cosmos DB. A replicação ajuda a fornecer acesso consistente aos recursos e a manter os padrões da organização, independentemente de qual região você esteja implantando.

O Azure Blueprints é composto de vários artefatos, e você pode encontrar a lista de artefatos com suporte aqui: <https://docs.microsoft.com/azure/governance/blueprints/overview#blueprint-definition>.

O Blueprints pode ser criado no portal do Azure, no Azure PowerShell, na CLI do Azure, nas APIs REST ou em modelos do ARM.

Na próxima seção, vamos ver um exemplo de como implementar recursos de governança do Azure. Serviços e recursos como RBAC, Azure Policy e bloqueios de recursos do Azure serão usados no exemplo.

## Um exemplo de implementação de recursos de governança do Azure

Nesta seção, vamos analisar um exemplo de implementação de arquitetura em uma organização fictícia que deseja implementar os recursos de governança e gerenciamento de custos do Azure.

### Contexto

**Company Inc** é uma empresa internacional que está implementando uma solução de mídias sociais em uma plataforma IaaS do Azure. Ela usa servidores Web e servidores de aplicativos implantados em redes e máquinas virtuais do Azure. O SQL do Azure Server age como o banco de dados de back-end.

### RBAC para Company Inc

A primeira tarefa é garantir que os proprietários de aplicações e as equipes apropriadas possam acessar seus recursos. Reconhece-se que cada equipe tem requisitos diferentes. Para fins de clareza, o SQL do Azure é implantado em um grupo de recursos separado em comparação aos artefatos da IaaS do Azure.

O administrador atribui as seguintes funções na assinatura:

Função	Atribuída a	Descrição
Proprietário	Administrador	Gerencia todos os grupos de recursos e a assinatura
Gerente de segurança	Administradores de segurança	Esta função permite que os usuários acessem a Central de Segurança do Azure e o status dos recursos.
Colaborador	Gerenciamento de infraestrutura	Gerencia máquinas virtuais e outros recursos.
Leitor	Desenvolvedores	Pode exibir os recursos, mas não pode modificá-los. Espera-se que os desenvolvedores trabalhem em seus ambientes de desenvolvimento/teste.

Tabela 5.1: diferentes funções com detalhes de acesso

## Azure Policy

A empresa deve implementar o Azure Policy para garantir que seus usuários sempre provisionem recursos de acordo com as diretrizes da empresa.

As políticas no Azure regem vários aspectos relacionados à implantação de recursos. As políticas também governarão as atualizações após a implantação inicial. Algumas das políticas que devem ser implementadas são fornecidas na seção a seguir.

### Implantações em determinados locais

Os recursos e implantações do Azure só podem ser executados para determinados locais escolhidos. Não seria possível implantar recursos em regiões fora da política. Por exemplo, as regiões permitidas são a Europa Ocidental e o Leste dos EUA. Não deve ser possível implantar recursos em qualquer outra região.

### Marcas de recursos e grupos de recursos

Todos os recursos do Azure, incluindo os grupos de recursos, obrigatoriamente terão marcas atribuídas a eles. As marcas incluirão, no mínimo, detalhes sobre o departamento, o ambiente, os dados de criação e o nome do projeto.

### Logs de diagnóstico e Application insights para todos os recursos

Todos os recursos implantados no Azure devem ter logs de diagnóstico e logs de aplicações habilitados sempre que possível.

## Bloqueios do Azure

Uma empresa deve implementar bloqueios do Azure para garantir que os recursos cruciais não sejam excluídos acidentalmente. Cada recurso que é crucial para o funcionamento de uma solução precisa ser bloqueado. Isso significa que até mesmo os administradores dos serviços em execução no Azure não têm a capacidade de excluir esses recursos. A única maneira de excluir um recurso é removê-lo primeiro.

Você também deve observar que:

Todos os ambientes de produção e pré-produção, além dos ambientes de desenvolvimento e teste, seriam bloqueados para exclusão.

Todos os ambientes de desenvolvimento e teste que têm instâncias únicas também seriam bloqueados para exclusão.

Todos os recursos relacionados à aplicação Web seriam bloqueados para exclusão para todos os ambientes de desenvolvimento à produção.

Todos os recursos compartilhados serão bloqueados para exclusão, independentemente do ambiente.

## Resumo

Neste capítulo, você aprendeu que a governança e o gerenciamento de custos estão entre algumas das principais prioridades para empresas que estão migrando para a nuvem. Ter uma assinatura pré-paga do Azure com um esquema pré-pago pode prejudicar o orçamento da empresa, pois qualquer um que tenha acesso à assinatura poderá provisionar quantos recursos quiser. Alguns recursos são gratuitos, mas outros são caros.

Você também aprendeu que é importante que as organizações mantenham o controle sobre seus custos de nuvem. As marcas ajudam na geração de relatórios de cobrança. Esses relatórios podem ser baseados em departamentos, projetos, proprietários ou qualquer outro critério. O custo e a governança são igualmente importantes. O Azure fornece bloqueios, políticas e RBAC para implementar a governança adequada. As políticas garantem que as operações de recursos possam ser negadas ou auditadas, os bloqueios garantem que os recursos não possam ser modificados nem excluídos, e o RBAC garante que os funcionários tenham as permissões corretas para realizar seus trabalhos. Com todos esses recursos, as empresas podem ter governança sólida e controle de custos para suas implantações do Azure.

No próximo capítulo, vamos abordar o gerenciamento de custos no Azure. Explicaremos diferentes métodos de otimização, gerenciamento de custos e APIs de cobrança.



# 6

# Gerenciamento de custos para soluções do Azure

No capítulo anterior, discutimos marcas, políticas e bloqueios e como eles podem ser aproveitados do ponto de vista da conformidade. As marcas nos permitem adicionar metadados aos nossos recursos; elas também nos ajudam no gerenciamento lógico de recursos. No portal do Azure, podemos filtrar recursos com base em marcas. Se presumirmos que há um grande número de recursos, o que é bastante comum nas empresas, a filtragem nos ajudará a gerenciar facilmente nossos recursos. Outro benefício das marcas é que elas podem ser usadas para filtrar relatórios de cobrança ou relatórios de uso em termos de marcas. Neste capítulo, vamos explorar o gerenciamento de custos para soluções do Azure.

O principal motivo pelo qual as corporações estão migrando para a nuvem é a economia de custos. Não há um custo inicial para ter uma assinatura do Azure. O Azure fornece uma assinatura pré-paga, em que a cobrança é baseada no consumo. O Azure mede o uso de recursos e fornece faturas mensais. Não há um limite superior para o consumo do Azure. Como estamos em uma nuvem pública, o Azure (como qualquer outro provedor de serviços) tem alguns limites rígidos e flexíveis sobre o número de recursos que podem ser implantados. Os limites flexíveis podem ser aumentados ao trabalhar com o Suporte do Azure. Alguns recursos têm um limite rígido. Os limites de serviço podem ser encontrados em <https://docs.microsoft.com/azure/azure-resource-manager/management/azure-subscription-service-limits> e o limite padrão varia de acordo com o tipo de assinatura que você tem.

É importante que as empresas mantenham um monitoramento sobre o consumo e o uso do Azure. Embora elas possam criar políticas para definir convenções e padrões organizacionais, também existe a necessidade de acompanhar os dados de cobrança e consumo. Além disso, elas devem aplicar as práticas recomendadas de consumo dos recursos do Azure para que o retorno seja maximizado. Para isso, os arquitetos precisam conhecer os recursos e as funcionalidades do Azure, seus custos correspondentes e a análise de custo/benefício de recursos e soluções.

Neste capítulo, abordaremos os seguintes tópicos:

- Detalhes das ofertas do Azure
- Cobrança
- Faturamento
- Uso e cotas
- APIs de Uso e Cobrança
- Calculadora de preços do Azure
- Práticas recomendadas para a otimização de custos

Vamos discutir cada um desses pontos.

## Detalhes das ofertas do Azure

O Azure tem diferentes ofertas que você pode comprar. Até agora, discutimos a pré-paga, mas há outras, como **Enterprise Agreements (EAs)**, Azure Sponsorship e Azure no CSP. Abordaremos cada uma delas, pois elas são muito importantes para a cobrança:

- **Pré-paga:** esta é uma oferta conhecida, em que os clientes pagam com base no consumo e as taxas estão disponíveis na documentação voltada para o público do Azure. Os clientes recebem uma fatura de uso todo mês da Microsoft e podem pagar por meio de cartão de crédito ou um método de pagamento de fatura.
- **EAs:** os EAs exigem um compromisso monetário com a Microsoft, o que significa que as organizações assinam um contrato com a Microsoft e prometem que usarão uma quantidade x de recursos do Azure. Se o uso ultrapassar a quantidade acordada, o cliente receberá uma fatura de excedente. Os clientes podem criar várias contas em um EA e ter várias assinaturas dentro dessas contas. Há dois tipos de EA: EA Direto e EA Indireto. Os clientes de EA Direto têm uma relação de cobrança direta com a Microsoft; por outro lado, com o EA Indireto, a cobrança é gerenciada por um parceiro. Os clientes de EA receberão ofertas e descontos melhores por causa do compromisso que firmam com a Microsoft. O EA é gerenciado por meio de um portal chamado portal do EA (<https://ea.azure.com>), e você precisa ter privilégios de inscrição para acessar esse portal.

- **Azure no CSP:** o Azure no CSP é onde um cliente entra em contato com um parceiro **Provedor de Soluções na Nuvem (CSP)**, e esse parceiro provisiona uma assinatura para o cliente. A cobrança será totalmente gerenciada pelo parceiro; o cliente não terá uma relação de cobrança direta com a Microsoft. A Microsoft fatura o parceiro e o parceiro fatura o cliente, adicionando sua margem.
- **Azure Sponsorship:** a Microsoft fornece um patrocínio a start-ups, ONGs e outras organizações sem fins lucrativos para usar o Azure. O patrocínio é para um prazo fixo e uma quantidade fixa de crédito. Se o prazo expirar ou o crédito for esgotado, a assinatura será convertida em uma assinatura pré-paga. Se as organizações quiserem renovar seu direito de patrocínio, terão que trabalhar com a Microsoft.

Acabamos de descrever algumas das ofertas do Azure. A lista completa está disponível em <https://azure.microsoft.com/support/legal/offer-details>, que inclui outras ofertas, como Azure for Students, Azure Pass e assinaturas de Desenvolvimento e Teste.

A seguir, vamos discutir a cobrança no Azure.

## Noções básicas de cobrança

O Azure é um utilitário de serviço que possui os seguintes recursos:

- Sem custos iniciais
- Sem taxas de cancelamento
- Cobrança por segundo, por minuto ou por hora, dependendo do tipo de recurso
- Pagamento com base no consumo conforme o uso

Nessas circunstâncias, é muito difícil estimar o custo inicial de consumo dos recursos do Azure. Cada recurso do Azure tem seu próprio modelo de custo e cobrança baseado em armazenamento, uso e intervalo de tempo. É muito importante que os departamentos de gerência, de administração e financeiro acompanhem o uso e os custos. O Azure fornece recursos de relatórios de cobrança e uso para ajudar a gerência superior e os administradores a gerar relatórios de custo e uso com base em vários critérios.

O portal do Azure fornece informações detalhadas sobre a cobrança e o uso por meio do recurso **Gerenciamento de Custos + Cobrança**, que pode ser acessado por meio da folha de navegação mestre, conforme mostrado na Figura 6.1:

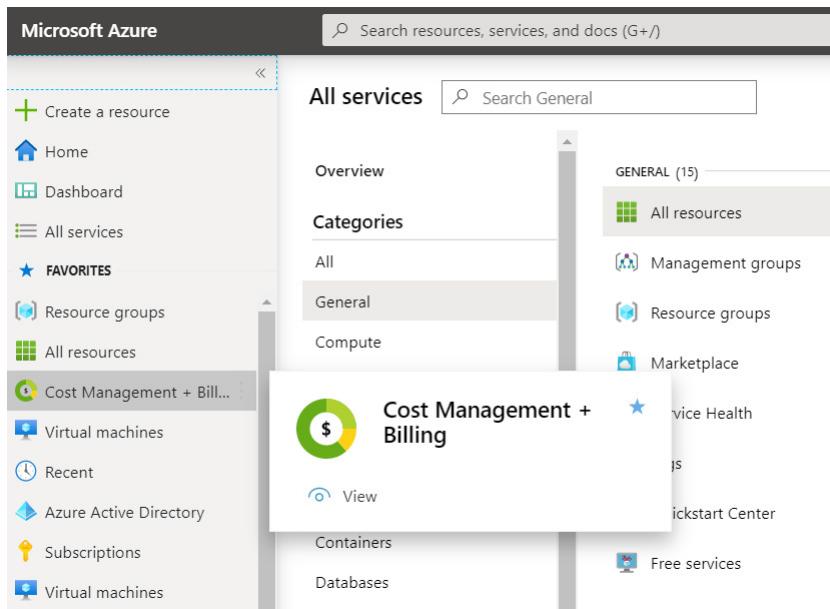


Figura 6.1: serviço Gerenciamento de Custos + Cobrança no portal do Azure

Observe que, se a sua cobrança for gerenciada por um CSP, você não terá acesso a esse recurso. Os clientes de CSP podem exibir seus custos no esquema pré-pago se fizerem a transição de suas assinaturas herdadas de CSP para o plano do Azure. Vamos discutir o plano do Azure e a plataforma de Comércio Moderno mais adiante no capítulo.

**Gerenciamento de Custos + Cobrança** mostra todas as assinaturas e os escopos de cobrança aos quais você tem acesso, conforme mostrado na Figura 6.2:

Subscription name	Subscription ID	Status	Last billed amount	Due date	Current Cost
Pay-As-You-Go Dev/Test		Active	Not available	Not available	₹0.00
Pay-As-You-Go		Active	Not available	Not available	₹0.00
Pay-As-You-Go		Active	Not available	Not available	₹0.00
[PROD] rithin.net		Active	Not available	Not available	₹231.45
Project Hawk Eye 360		Active	Not available	Not available	₹947.10
Pay-As-You-Go Dev/Test		Active	Not available	Not available	₹0.00

Figura 6.2: visão geral da cobrança de assinaturas de usuários

A seção **Gerenciamento de Custos** tem várias folhas, como:

- **Análise de custo** para analisar o uso de um escopo.
- **Orçamentos** para orçamentos de configurações.
- **Alertas de custo** para notificar os administradores quando o uso exceder um determinado limite.
- **Recomendações do Assistente** para obter orientações sobre como fazer possíveis economias. Vamos discutir o Assistente do Azure na última seção deste capítulo.
- **Exportações** para automatizar as exportações de uso para o Armazenamento do Azure.
- **Cloudyn**, que é uma ferramenta usada pelos parceiros CSP para analisar os custos, pois eles não têm acesso ao Gerenciamento de Custos.
- **Conectores do AWS** para conectar seus dados de consumo do AWS ao Gerenciamento de Custos do Azure.

As diferentes opções disponíveis no Gerenciamento de Custos do Azure são mostradas na Figura 6.3:

The screenshot shows the Azure Cost Management Overview page. The left sidebar includes links for Home, Cost Management + Billing, Overview, Go to subscription, Access control, Diagnose and solve problems, Cost analysis, Cost alerts, Budgets, Advisor recommendations, Cloudyn, and Connectors for AWS (Preview). The main content area has a title 'Cost Management: Pay-As-You-Go Dev/Test | Overview' and a scope filter set to 'Pay-As-You-Go Dev/Test'. It features three main sections: 'Analyze cloud costs' (with a description about breaking down and analyzing costs), 'Monitor with budgets' (with a description about creating budgets to control costs), and 'Optimize with recommendations' (with a description about viewing advisor recommendations to identify unused resources). Each section includes a small icon and a detailed description below it.

Figura 6.3: visão geral do Gerenciamento de Custos

Clicar no menu **Análise de custo** nesta folha fornece um painel interativo abrangente, usando o custo que pode ser analisado com diferentes dimensões e medidas:

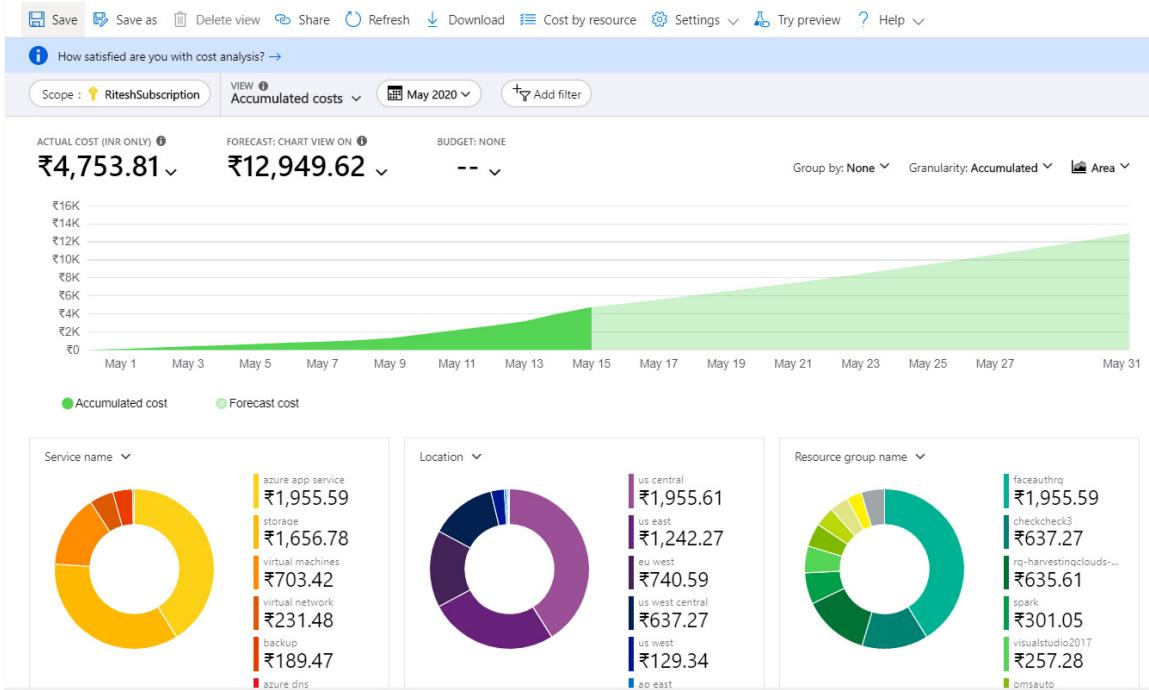


Figura 6.4: analisando os custos de assinatura por meio da opção Análise de custo

O painel não só mostra o custo atual, mas também prevê o custo e o divide com base em várias dimensões. **Nome do serviço**, **Localização** e **Nome do grupo de recursos** são fornecidas por padrão, mas podem ser alteradas para outras dimensões também. Sempre haverá um escopo associado a cada exibição. Alguns dos escopos disponíveis são conta de cobrança, grupo de gerenciamento, assinatura e grupo de recursos. Você pode mudar o escopo dependendo do nível que deseja analisar.

O menu **Orçamentos** à esquerda nos permite configurar o orçamento para um melhor gerenciamento de custos e fornece recursos de alerta caso o custo real exceda as estimativas de orçamento:

Dashboard > Cost Management + Billing > Cost Management: [PROD] rithin.net | Budgets >

**Cost Management: [PROD] rithin.net | Create budget**

Subscription

Search (Ctrl+ /) <>

**Create a budget** **Set alerts**

Overview

Go to subscription

Access control

Diagnose and solve problems

**Cost Management**

Cost analysis

Cost alerts

**Budgets** (selected)

Advisor recommendations

Cloudyn

**Settings**

Exports

Connectors for AWS (Preview)

Support + troubleshooting

New support request

Create a budget and set alerts to help you monitor your costs.

**Budget scoping**

The budget you create will be assigned to the selected scope. Use additional filters like resource groups to have your budget monitor with more granularity as needed.

Scope: [PROD] rithin.net [Change scope](#)

[Add filter](#)

**Budget Details**

Give your budget a unique name. Select the time window it analyzes during each evaluation period, its expiration date and the amount.

* Name	Enter a unique name		
* Reset period	Monthly		
* Creation date	2020	May	1
* Expiration date	2022	April	30

**Figura 6.5: criando um orçamento**

O Gerenciamento de Custos também nos permite buscar dados de custos de outras nuvens, como o AWS, nos painéis atuais, gerenciando assim os custos de várias nuvens em uma única folha e painel. No entanto, esse recurso está em versão preliminar no momento da redação. Esse conector se tornará passível de cobrança em 25 de agosto de 2020.

Você precisa preencher os detalhes da sua função no AWS e outros detalhes para obter as informações sobre custos, conforme mostrado na Figura 6.5. Se você não souber como criar a política e a função no AWS, consulte <https://docs.microsoft.com/azure/cost-management-billing/costs/aws-integration-set-up-configure#create-a-role-and-policy-in-aws>:

The screenshot shows the Azure Cost Management + Billing dashboard. On the left, there's a sidebar with links like 'Cost analysis', 'Cost alerts', 'Budgets', 'Advisor recommendations', 'Cloudyn', 'Exports', and 'Connectors for AWS (Preview)'. The main area has a heading 'Connect to your' and a button 'Add connector'. A 'Create connector' dialog box is open on the right, divided into 'Basics' and 'AWS properties' sections. The 'Basics' section includes fields for 'Display name' (aws-cost), 'Management Group' (mgmt), 'Billing' (Trial), 'Auto-Renew' (Off), and 'Subscription' ([PROD] rithin.net). The 'AWS properties' section includes 'Role ARN' (arn:awsiam::487610937256:role/acm-manager), 'External ID' (\*\*\*\*), and 'Report name' (432263259397). At the bottom of the dialog are 'Previous', 'Next', and 'Create' buttons.

Figura 6.6: criando um conector do AWS no Gerenciamento de Custos

Os relatórios de custos também podem ser exportados para uma conta de armazenamento de maneira programada.

Parte da análise de custo também está disponível na folha **Assinaturas**. Na seção **Visão geral**, você pode ver os recursos e seu custo. Além disso, há outro gráfico, em que você pode ver o gasto atual, a previsão e o crédito de saldo (se estiver usando uma assinatura baseada em crédito).

A Figura 6.7 mostra informações sobre custos:

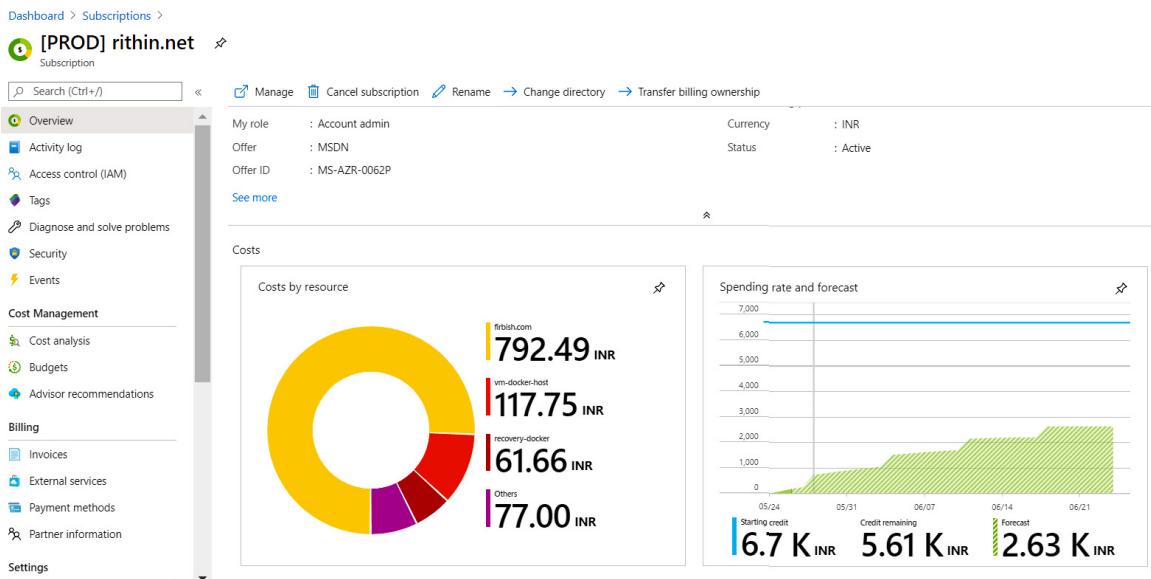


Figura 6.7: análise de custo para a assinatura

Ao clicar em qualquer um dos custos na Figura 6.7, você será redirecionado para a seção **Gerenciamento de Custos – Análise de Custo**. Há várias dimensões no Gerenciamento de Custos com as quais você pode agrupar os dados para análise. As dimensões disponíveis variarão com base no escopo selecionado. Algumas das dimensões comumente usadas são:

- Tipos de recursos
- Grupo de recursos
- Marca
- Localização do recurso
- ID do recurso
- Categoria do medidor
- Subcategoria do medidor
- Serviço

No início do capítulo, dissemos que as marcas podem ser usadas para o gerenciamento de custos. Por exemplo, digamos que você tenha uma marca chamada Departamento com valores de TI, RH e Finanças. A marcação adequada dos recursos ajudará você a entender os custos incorridos por cada departamento. Você também pode fazer o download do relatório de custos como um arquivo CSV, Excel ou PNG usando o botão **Download**.

Além disso, o Gerenciamento de Custos oferece suporte a várias exibições. Você pode criar seu próprio painel e salvá-lo. Os clientes de EA recebem o benefício adicional do conector de Gerenciamento de Custos ou do Power BI. Com o conector, os usuários podem obter as estatísticas de uso para o Power BI e criar visualizações.

Até agora, temos discutido como podemos acompanhar nosso uso com o Gerenciamento de Custos. Na próxima seção, exploraremos como o faturamento funciona para os serviços que usamos.

## Faturamento

O sistema de cobrança do Azure também fornece informações sobre faturas que são geradas mensalmente.

Dependendo do tipo de oferta, o método de faturamento pode variar. Para usuários pré-pagos, as faturas serão enviadas mensalmente para o administrador da conta. No entanto, para clientes de EA, a fatura será enviada para o contato na inscrição.

Ao clicar no menu **Faturas**, será exibida uma lista de todas as faturas geradas e ao clicar em qualquer uma das faturas, você verá detalhes sobre essa fatura. A Figura 6.8 mostra como as faturas são exibidas no portal do Azure:

Invoice ID	Billing Period	Invoice date	Amount	Status	Type	Download
G001290052	4/1/2020-4/30/2020	5/9/2020	INR 0.00	Paid	Azure Marketplace and Reserv...	
G001124632	3/1/2020-3/31/2020	4/9/2020	INR 0.00	Paid	Azure Marketplace and Reserv...	
G001009558	2/1/2020-2/29/2020	3/9/2020	INR 0.00	Paid	Azure Marketplace and Reserv...	

Figura 6.8: lista de faturas e seus detalhes

Há dois tipos de faturas: um é para serviços do Azure, como SQL, Máquinas Virtuais e Rede. O outro é para Azure Marketplace e Reservas. O Azure Marketplace fornece serviços de parceiros de diferentes fornecedores para clientes. Vamos falar sobre as Reservas do Azure posteriormente.

Por padrão, para uma assinatura pré-paga, o administrador da conta tem acesso às faturas. Se quiser, ele pode delegar acesso a outros usuários, como a equipe de finanças da organização, escolhendo a opção **Acessar a fatura** na Figura 6.8. Além disso, o administrador da conta pode escolher os endereços de email para os quais deseja enviar as cópias das faturas.

A opção **Fatura por Email** não está disponível para o Plano de Suporte agora. Como alternativa, você pode visitar o portal de Contas e fazer o download da fatura. A Microsoft está lentamente migrando desse portal e a maioria dos recursos estão sendo preferidos à medida que são integrados ao portal do Azure.

Até agora, discutimos assinaturas e como o faturamento é feito. Algo novo que foi introduzido pela Microsoft é o Comércio Moderno. Com essa nova experiência de comércio, o processo e a experiência de compra foram simplificados. Vamos ver em detalhes o Comércio Moderno e saber como ele é diferente da plataforma herdada que discutimos até agora.

## A experiência do Comércio Moderno

Se a sua organização já estiver trabalhando com a Microsoft, você saberá que existem vários contratos envolvidos para cada oferta, como Web Direct, EAs, CSP, **Contrato de Produtos e Serviços da Microsoft (MSPA)**, **Registros de Nuvem e Servidor (SCE)** etc. Juntamente com isso, cada um deles tem seu próprio portal; por exemplo, os EAs têm o portal do EA, o CSP tem o portal do Partner Center e o Licenciamento por Volume também tem seu próprio portal.

Cada oferta vem com um conjunto diferente de termos e condições, e os clientes precisam analisá-los sempre que fizerem uma compra. A transição de uma oferta para outra não é muito fácil, pois cada oferta tem um conjunto diferente de termos e condições. Vamos imaginar que você já tenha uma assinatura de EA e gostaria de convertê-la em uma assinatura de CSP; talvez seja necessário excluir alguns dos serviços de parceiros, pois eles não têm suporte no CSP. Para cada produto, cada oferta terá regras diferentes. Do ponto de vista do cliente, é muito difícil entender o que oferece suporte ao que e como as regras diferem.

Abordando esse problema, a Microsoft recentemente emitiu um novo contrato chamado **Contrato de Cliente da Microsoft (MCA)**. Ele atuará como os termos e condições básicos. Você poderá fazer alterações nele sempre que necessário quando se inscrever em um novo programa.

Para o Azure, haverá três programas **Go-To-Market (GTM)**:

- **Liderado pelo campo:** os clientes interagirão com a Equipe de Contas da Microsoft e a cobrança será gerenciada diretamente pela Microsoft. Eventualmente, isso substituirá os EAs.
- **Liderado pelo parceiro:** é equivalente ao programa Azure no CSP, em que um parceiro gerencia sua cobrança. Há diferentes parceiros em todo o mundo. Uma rápida pesquisa na Web ajudará você a encontrar os parceiros ao seu redor. Este programa substituirá o programa Azure no CSP. Como primeiro passo para o Comércio Moderno, um parceiro assinará um **Contrato de Parceiro da Microsoft (MPA)** com a Microsoft e fará a transição de seus clientes existentes, fazendo com que eles assinem o MCA. No momento da redação deste livro, muitos parceiros já tinham feito a transição de seus clientes para o Comércio Moderno, e a nova experiência de comércio está disponível em 139 países.
- **Self-service:** este será um substituto do Web Direct. Ele não requer nenhum envolvimento do parceiro ou da Equipe de Contas da Microsoft. Os clientes podem comprar diretamente em [microsoft.com](https://microsoft.com) e eles assinarão o MCA durante a compra.

No Azure, a cobrança será feita no Plano do Azure, e a cobrança sempre será alinhada com o mês do calendário. A compra de um Plano do Azure é muito semelhante à compra de qualquer outra assinatura. A diferença é que o MCA será assinado durante o processo.

O Plano do Azure pode hospedar várias assinaturas e atuará como um contêiner de nível raiz. Todo o uso está vinculado a um único Plano do Azure. Todas as assinaturas dentro do Plano do Azure atuarão como contêineres para hospedar serviços, como Máquinas Virtuais, Banco de Dados SQL e Rede.

Algumas das mudanças e avanços que pudemos observar após a introdução do Comércio Moderno são:

- Eventualmente, os portais serão preteridos. Por exemplo, os clientes de EA anteriores só podiam fazer o download das informações sobre uso da inscrição no portal do EA. Agora, a Microsoft fez a integração ao Gerenciamento de Custos do Azure com uma experiência mais avançada do que o portal do EA.
- Os preços serão estabelecidos em USD e cobrados na moeda local. Se a sua moeda não for USD, então a taxa de **intercâmbio (FX)** será aplicada e estará disponível na sua fatura. A Microsoft usa as taxas de intercâmbio da Thomson Reuters, e essas taxas serão atribuídas no primeiro dia de cada mês. Esse valor será consistente ao longo do mês, independentemente da taxa de mercado.
- Os clientes de CSP que fizerem a transição para o novo Plano do Azure poderão usar o Gerenciamento de Custos. O acesso ao Gerenciamento de Custos abre um novo mundo de rastreamento de custos, pois fornece acesso a todos os recursos nativos do Gerenciamento de Custos.

Todas as assinaturas que discutimos até agora eventualmente serão migradas para um Plano do Azure, que é o futuro do Azure. Agora que você entende as noções básicas do Comércio Moderno, vamos discutir outro tópico que desempenha um papel muito importante ao arquitetar soluções. A maioria dos serviços tem limites por padrão; alguns desses limites podem ser aumentados, enquanto alguns são rígidos. Ao arquitetar uma solução, precisamos garantir que haja uma cota ampla. O planejamento de capacidade é uma parte vital do design arquitetônico. Na próxima seção, você aprenderá mais sobre limites em assinaturas.

## Uso e cotas

Como mencionado na seção anterior, o planejamento de capacidade precisa ser uma das primeiras etapas ao arquitetar uma solução. Precisamos verificar se a assinatura tem cota suficiente para acomodar os novos recursos que estamos arquitetando. Caso contrário, durante a implantação, poderemos enfrentar problemas.

Cada assinatura possui um limite de cota para cada tipo de recurso. Por exemplo, pode haver no máximo 10 endereços IP públicos provisionados com uma conta Microsoft MSDN. Da mesma forma, todos os recursos têm um limite máximo padrão para cada tipo de recurso. Esses números de tipos de recursos para uma assinatura podem ser aumentados ao contatar o Suporte do Azure ou ao clicar no botão **Solicitar Aumento** na folha **Uso + Cota** na página **Assinatura**.

Considerando o número de recursos em cada região, será um desafio examinar a lista. O portal fornece opções para filtrar o conjunto de dados e procurar o que queremos. Na Figura 6.9, você pode ver que, se filtrarmos a localização para **EUA Central** e definirmos o provedor de recursos como **Microsoft.Storage**, poderemos confirmar quais cotas estão disponíveis para contas de armazenamento:

The screenshot shows the Azure portal's 'Usage + quotas' blade for a subscription named '[PROD] rithin.net'. The left sidebar lists various management options like 'Payment methods', 'Partner information', 'Programmatic deployment', 'Resource groups', 'Resources', and 'Policies'. The 'Usage + quotas' option is currently selected. The main area displays a table of quotas for 'Storage Accounts' under 'Microsoft.Storage' provider in 'Central US' location. The table shows a usage of 0% out of 250. At the top of the blade, there are dropdown menus for 'All service quotas', 'Provider' (set to 'Microsoft.Storage'), 'Location' (set to 'Central US'), and a 'Show all' button. A 'Request Increase' button is located in the top right. A message at the top states: 'You can use each Microsoft Azure resource up to its quota. Each subscription has separate quotas and usage is tracked per subscription. If you reach a quota cap, you can request an increase via Help+Support. [Learn more](#)'.

Quota	Provider	Location	Usage
Storage Accounts	Microsoft.Storage	Central US	0 % 0 of 250

Figura 6.9: uso e cota para uma localização e um provedor de recursos específicos

Você pode ver claramente na *Figura 6.9* que não criamos nenhuma conta de armazenamento na região EUA Central, e isso nos deixa com uma cota de 250 contas. Se a solução que estamos arquitetando exigir mais de 250 contas, precisaremos clicar em **Solicitar Aumento**, para entrar em contato com o Suporte do Azure.

Essa folha nos dá a liberdade de executar o planejamento de capacidade antes da implantação.

Ao filtrar o relatório, usamos o termo **provedor de recursos** e selecionamos **Microsoft Storage**. Na próxima seção, veremos em detalhes o que esse termo significa.

## Provedores de recursos e tipos de recursos

Se você estiver interagindo com o portal do Azure, filtrando serviços ou filtrando o relatório de uso de cobrança, talvez seja necessário trabalhar com provedores de recursos e tipos de recursos. Por exemplo, ao criar uma máquina virtual, você interage com o provedor de recursos **Microsoft.Compute** e o tipo de recurso **virtualMachines**. O botão **criar** em que você clica para criar as máquinas virtuais se comunica com o provedor de recursos por meio de uma API para fazer sua implantação. Isso é sempre denotado no formato **{resource-provider}/{resource-type}**. Portanto, o tipo de recurso da máquina virtual é **Microsoft.Compute/virtualMachines**. Em suma, os provedores de recursos ajudam a criar tipos de recursos.

É necessário que os provedores de recursos sejam registrados com uma assinatura do Azure. Os tipos de recursos não estarão disponíveis em uma assinatura se os provedores de recursos não estiverem registrados. Por padrão, a maioria dos provedores é registrada automaticamente; dito isso, haverá cenários em que devemos registrar de forma manual.

Para obter uma lista dos provedores que estão disponíveis, registrados e não registrados, e para registrar os provedores não registrados ou vice-versa, o painel mostrado na *Figura 6.10* poderá ser usado. Para essa operação, você precisa ter as funções necessárias atribuídas; as funções Proprietário ou Colaborador serão suficientes. A *Figura 6.10* mostra como é o painel:

The screenshot shows the Azure portal interface for the [PROD] rithin.net subscription. The left sidebar contains navigation links such as Dashboard, Subscription, Search (Ctrl+ /), External services, Payment methods, Partner information, Settings, Programmatic deployment, Resource groups, Resources, Usage + quotas, Policies, Management certificates, My permissions, Resource providers (which is selected and highlighted in grey), Deployments, Properties, Resource locks, Support + troubleshooting, and New support request. At the top right are buttons for Register, Unregister, Refresh, and a Filter by name... input field.

Provider	Status
Microsoft.RecoveryServices	Registered
Microsoft.AlertsManagement	Registered
Microsoft.Authorization	Registered
Microsoft.Automation	Registered
Microsoft.DomainRegistration	Registered
Microsoft.ResourceHealth	Registered
Microsoft.Web	Registered
Microsoft.Storage	Registered
microsoft.insights	Registered
Microsoft.Network	Registered
Microsoft.Compute	Registered
Microsoft.AAD	Registered
Microsoft.Advisor	Registered
Microsoft.Security	Registered

Figura 6.10: lista de provedores de recursos registrados e não registrados

Na seção anterior, discutimos como fazer o download de faturas e informações sobre uso. Se você precisar fazer o download dos dados programaticamente e salvá-los, poderá usar APIs. A próxima seção fala sobre as APIs de Cobrança do Azure.

## APIs de Uso e Cobrança

Embora o portal seja uma excelente maneira de encontrar informações sobre o uso, a cobrança e o faturamento manualmente, o Azure também fornece as seguintes APIs para recuperar detalhes de forma programática e criar relatórios e painéis personalizados. As APIs variam de acordo com o tipo de assinatura que você está usando. Como há muitas APIs, vamos compartilhar a documentação da Microsoft com cada API para que você possa explorar todas elas.

### APIs de Cobrança Empresarial do Azure

Os clientes de EA têm um conjunto dedicado de APIs disponíveis para trabalhar com dados de cobrança. As APIs a seguir usam a chave de API do portal do EA para autenticação; os tokens do Azure Active Directory não funcionarão com elas:

- **API de Saldo e Resumo:** como discutimos anteriormente, o EA funciona com um compromisso monetário, e é muito importante acompanhar saldo, excedente, ajustes de crédito e encargos do Azure Marketplace. Usando esta API, os clientes podem obter o saldo e o resumo de um período de cobrança.
- **API de Detalhes de Uso:** a API de Detalhes de Uso ajudará você a obter informações de uso diário sobre a inscrição com granularidade até o nível da instância. A resposta desta API será como o relatório de uso disponível para download no portal do EA.
- **API de Encargos de Lojas do Marketplace:** esta é uma API dedicada para extrair os encargos das compras no marketplace.
- **API de Tabela de Preços:** cada inscrição terá uma tabela de preços especial, e os descontos variarão de cliente para cliente. A API de Tabela de Preços pode obter a lista de preços.
- **API de Detalhes da Instância Reservada:** ainda não discutimos as Reservas do Azure, mas elas serão discutidas até o final deste capítulo. Usando esta API, você pode obter informações de uso sobre as reservas e uma lista das reservas na inscrição.

Aqui está o link para a documentação das APIs do EA: <https://docs.microsoft.com/azure/cost-management-billing/manage/enterprise-api>.

Agora vamos analisar as APIs de Consumo do Azure.

## APIs de Consumo do Azure

As APIs de Consumo do Azure podem ser usadas com assinaturas de EA e do Web Direct (com algumas exceções). Isso requer um token, que precisa ser gerado por meio da autenticação em relação ao Azure Active Directory. Como essas APIs também oferecem suporte ao EA, não confunda esse token com a chave de API do EA que mencionamos na seção anterior. Aqui estão as principais APIs que são autoexplicativas:

- API de Detalhes de Uso
- API de Encargos do Marketplace
- API de Recomendações de Reservas
- API de Resumo e Detalhes de Reservas

Os clientes de EA têm suporte adicional para as seguintes APIs:

- Tabela de preços
- Orçamentos
- Saldos

A documentação está disponível aqui: <https://docs.microsoft.com/azure/cost-management-billing/manage/consumption-api-overview>.

Além disso, há outro conjunto de APIs que podem ser usadas apenas por clientes do Web Direct:

- **API de Uso de Recursos do Azure:** esta API pode ser usada com uma assinatura de EA ou uma assinatura pré-paga para fazer o download dos dados de uso.
- **API de RateCard de Recursos do Azure:** só é aplicável ao Web Direct; os EAs não têm suporte. Os clientes do Web Direct podem usá-la para fazer o download de tabelas de preços.
- **API de Download de Faturas do Azure:** só é aplicável a clientes do Web Direct. Ela é usada para fazer o download de faturas programaticamente.

Os nomes podem parecer familiares, e a diferença é apenas o ponto de extremidade que estamos chamando. Para as APIs de Cobrança Empresarial do Azure, a URL começará com <https://consumption.azure.com> e, para as APIs de Consumo do Azure, a URL começará com <https://management.azure.com>. Você pode diferenciá-las dessa forma. Na próxima seção, você verá um novo conjunto de APIs que são usadas especificamente pelo Gerenciamento de Custos.

## APIs de Gerenciamento de Custos do Azure

Com a introdução do Gerenciamento de Custos do Azure, um novo conjunto de APIs está disponível para o cliente usar. Essas APIs são o alicerce do Gerenciamento de Custos, que usamos no portal do Azure anteriormente. As principais APIs são:

- **API de Uso de Consultas:** esta é a mesma API usada pela Análise de Custo no portal do Azure. Podemos personalizar a resposta com o que precisamos usando uma carga. É muito útil quando queremos um relatório personalizado. O intervalo de datas não pode exceder 365 dias.
- **API de Orçamentos:** os orçamentos são outro recurso do Gerenciamento de Custos do Azure, e esta API nos permite interagir com os orçamentos programaticamente.
- **API de Previsão:** pode ser usada para obter uma previsão de um escopo. A API de Previsão está disponível apenas para clientes de EA no momento.
- **API de Dimensões:** anteriormente, quando estávamos discutindo o Gerenciamento de Custos, dissemos que ele oferece suporte a várias dimensões com base no escopo. Se você quiser obter a lista de dimensões com suporte baseadas em um escopo específico, poderá usar esta API.
- **API de Exportações:** outro recurso do Gerenciamento de Custos é que podemos automatizar a exportação de relatórios para uma conta de armazenamento. A API de Exportações pode ser usada para interagir com a configuração de exportação, como o nome da conta de armazenamento, a personalização, a frequência e assim por diante.

Confira a documentação oficial em <https://docs.microsoft.com/rest/api/cost-management>.

Como o Comércio Moderno está expandindo o MCA, há um novo conjunto de APIs que podem ser exploradas aqui: <https://docs.microsoft.com/rest/api/billing>.

Você pode ter notado que não mencionamos o CSP em nenhum desses cenários. No CSP, o cliente não tem acesso à cobrança, pois ela é gerenciada por um parceiro, portanto, as APIs não são expostas. No entanto, uma transição para um Plano do Azure permitiria que o cliente de CSP usasse as APIs de Gerenciamento de Custos do Azure para ver as taxas de varejo.

Qualquer linguagem de programação ou script pode usar essas APIs e combiná-las para criar soluções completas e abrangentes de cobrança. Na próxima seção, vamos nos concentrar na calculadora de preços do Azure, que ajudará o cliente ou o arquiteto a entender o custo de uma implantação.

## Calculadora de preços do Azure

O Azure oferece uma calculadora de custos para que os usuários e clientes estimem seu custo e uso. Essa calculadora está disponível em <https://azure.microsoft.com/pricing/calculator>.

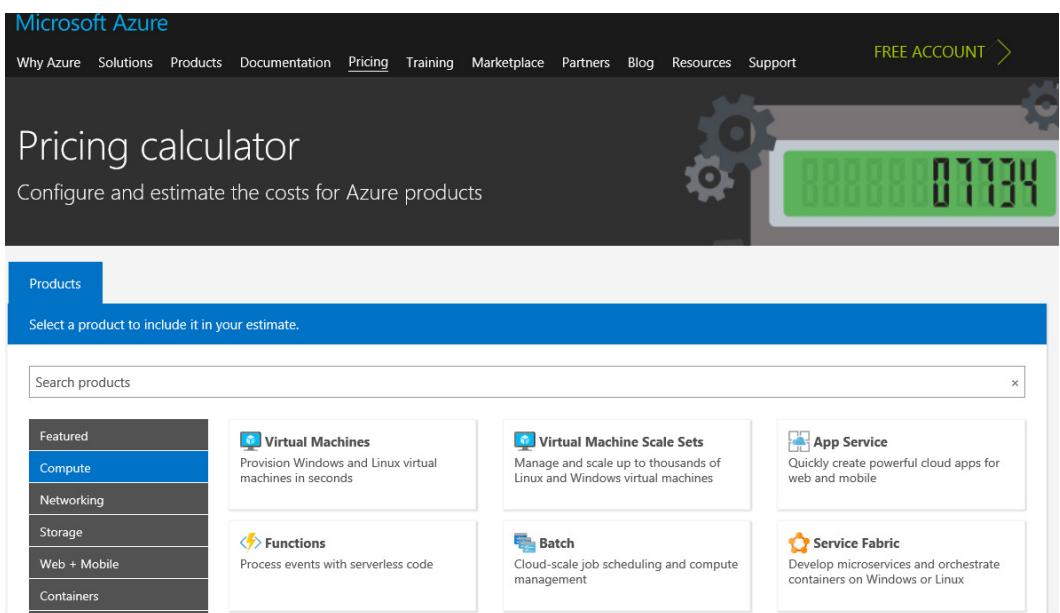


Figura 6.11: calculadora de preços do Azure

Os usuários podem selecionar vários recursos no menu à esquerda, e eles serão adicionados à calculadora. No exemplo a seguir, uma máquina virtual é adicionada. Uma configuração adicional referente a região, sistema operacional, tipo, camada, tamanho de instância, número de horas e contagem de máquinas virtuais deve ser fornecida para calcular os custos:

#### Your Estimate

[Virtual Machines](#) 1: D1: 1 cores, 3.5 GB RAM, 50 GB disk

	Virtual Machines	
REGION:	OPERATING SYSTEM:	TYPE:
West US	Windows	(OS Only)
TIER:	<input type="checkbox"/> ADD MANAGED DISKS	
Standard		
INSTANCE:	D1: 1 Core(s), 3.5 GB RAM, 50 GB Disk, \$ 0.140/hour	
1	744	= \$ 104.16
Virtual machines	Hours	

Figura 6.12: fornecer detalhes de configuração para calcular custos de recursos

Da mesma forma, o custo do Azure Functions, que está relacionado ao tamanho de memória, ao tempo de execução e às execuções por segundo das máquinas virtuais, é mostrado na Figura 6.13:

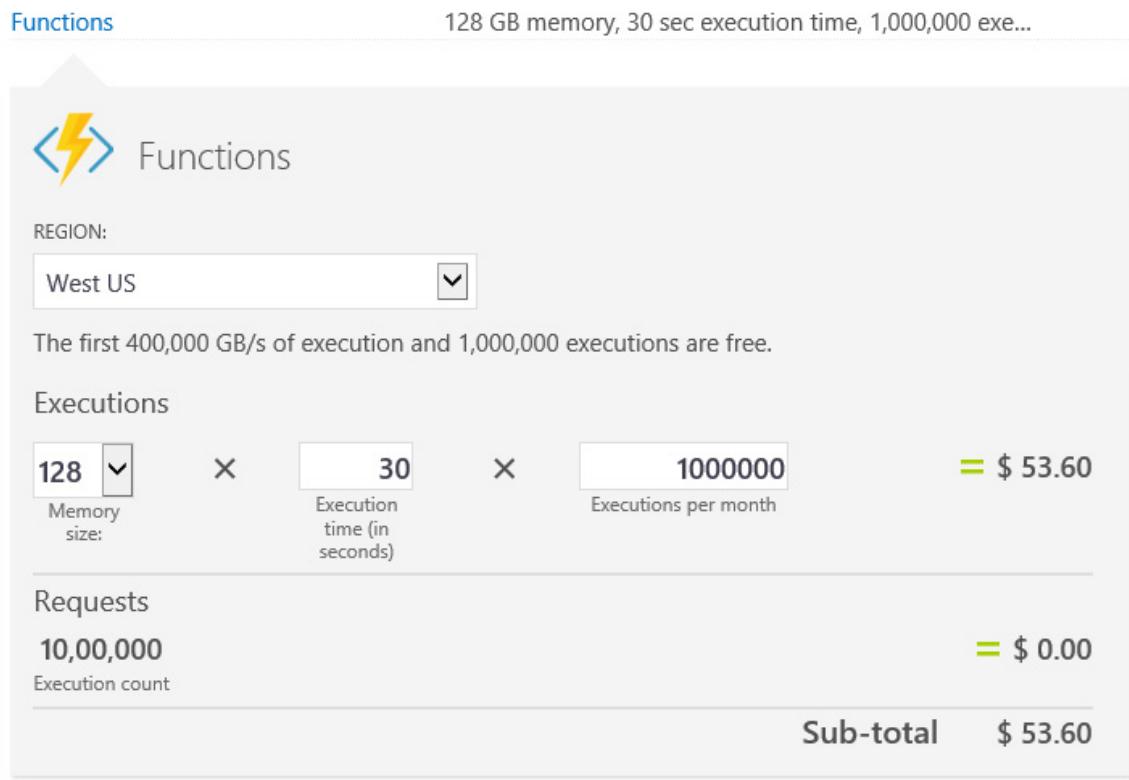


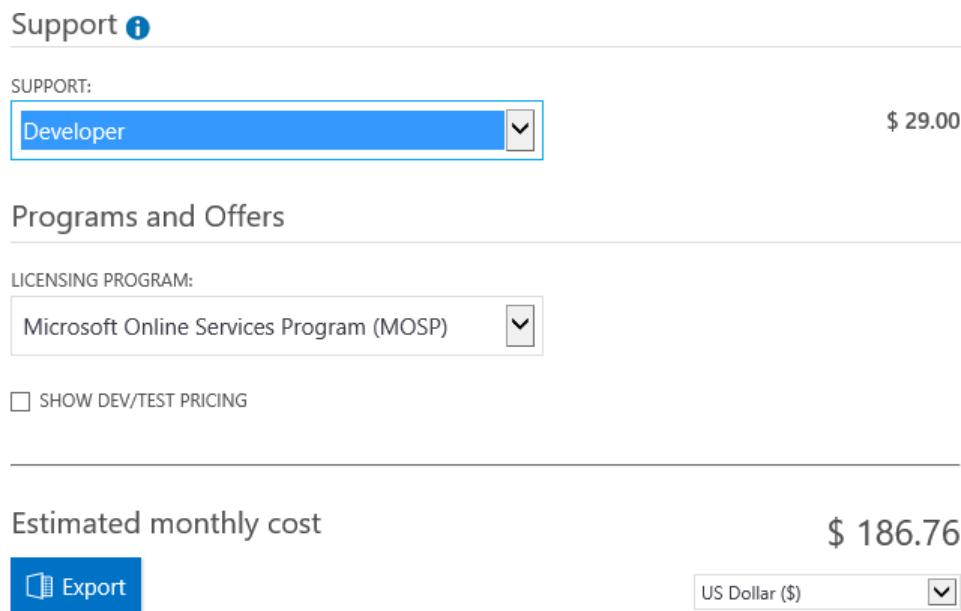
Figura 6.13: calculando os custos do Azure Functions

O Azure fornece diferentes níveis e planos de suporte:

- Suporte padrão: gratuito
- Suporte Developer: US\$ 29 por mês
- Suporte Standard: US\$ 300 por mês
- Professional Direct: US\$ 1.000 por mês

Para ver uma comparação completa dos planos de suporte, consulte <https://azure.microsoft.com/support/plans>.

Você pode selecionar o suporte que deseja, dependendo de seus requisitos. Por fim, o custo total estimado é exibido:



**Figura 6.14:** estimativa de custo para o plano de suporte selecionado

É importante que os arquitetos compreendam todos os recursos do Azure usados na arquitetura e na solução. O sucesso da calculadora do Azure depende dos recursos selecionados e de como eles são configurados. Qualquer informação incorreta levaria a estimativas tendenciosas e erradas, que seriam diferentes da cobrança real.

Chegamos à última seção deste capítulo. Abordamos as noções básicas de cobrança e, agora, é hora de aprender as práticas recomendadas. Seguir as práticas recomendadas ajudará você a alcançar a otimização de custos.

## Práticas recomendadas

Os arquitetos precisam compreender sua arquitetura e os componentes do Azure utilizados. Com base em monitoramento ativo, auditorias e uso, eles devem determinar a melhor oferta da Microsoft em termos de SKU, tamanho e recursos. Esta seção detalhará algumas das práticas recomendadas a serem adotadas de uma perspectiva de otimização de custos.

## Governança do Azure

A Governança do Azure pode ser definida como um conjunto de processos ou mecanismos que podem ser aproveitados para manter o controle total sobre os recursos implantados no Azure. Estes são alguns dos pontos principais:

- Configurar uma convenção de nomenclatura para todos os tipos de recursos e grupos de recursos. Garantir que a convenção de nomenclatura seja seguida de forma consistente e abrangente em todos os recursos e grupos de recursos. Isso pode ser feito estabelecendo políticas do Azure.

- Configurar uma organização lógica e várias taxonomias para recursos, grupos de recursos e assinaturas aplicando marcas a eles. As marcas categorizam os recursos e também podem ajudar a avaliar os custos de diferentes perspectivas. Isso pode ser feito estabelecendo políticas do Azure, e várias políticas podem ser combinadas em iniciativas. Essas iniciativas podem ser aplicadas e, por sua vez, aplicarão todas as políticas para verificações e relatórios de conformidade.
- Usar o Azure Blueprints em vez de modelos do ARM diretamente. Isso garantirá que a implantação de novos ambientes, recursos e grupos de recursos possa ser padronizada de acordo com os padrões corporativos, incluindo convenções de nomenclatura e o uso de marcas.

## Práticas recomendadas de computação

A computação refere-se a serviços que ajudam na execução de serviços. Algumas das práticas recomendadas de computação que os arquitetos do Azure devem seguir para a utilização ideal de recursos e a eficiência de custos são as seguintes:

- Aproveitar o Assistente do Azure para ver as opções disponíveis a fim de economizar custos nas suas máquinas virtuais e descobrir se uma máquina virtual é subutilizada. O Assistente usa padrões de machine learning e inteligência artificial para analisar seu uso e fornecer recomendações. Essas recomendações desempenham um papel importante na otimização de custos.
- Usar **Instâncias Reservadas (RI)** do Azure. As RIs podem proporcionar possíveis economias nos custos de computação pagando o custo da máquina virtual com antecedência ou mensalmente. O prazo da RI pode ser de um ano ou três anos. Se você comprar uma RI, isso reduzirá o custo de computação e somente os encargos de disco, rede e licença (se houver) para uma máquina virtual serão exibidos. Se você tiver cinco máquinas virtuais, poderá optar por cinco RIs para suprimir completamente o custo de computação. As RIs automaticamente procuram máquinas virtuais com o SKU correspondente e se anexam a elas. As possíveis economias podem variar de 20% a 40%, dependendo do tamanho da máquina virtual.
- Com o **Benefício Híbrido do Azure (AHUB)**, você pode usar suas próprias licenças do Windows Server ou do SQL para reduzir o custo da licença. A combinação das RIs e do AHUB pode proporcionar uma economia imensa.
- Escolher o melhor local para seus serviços de computação, como as máquinas virtuais. Escolha um local onde todos os recursos do Azure estejam reunidos na mesma região. Isso evitará o tráfego de saída.
- Escolher o tamanho ideal para as máquinas virtuais. Uma máquina virtual grande tem um custo mais alto do que uma pequena, e talvez uma máquina virtual grande não seja necessária.

- Redimensionar as máquinas virtuais em períodos de alta demanda e reduzir seu tamanho quando a demanda diminuir. O Azure lança novos SKUs com bastante frequência. Se um novo tamanho for mais adequado às suas necessidades, ele deverá ser usado.
- Desligar os serviços de computação fora do horário comercial ou quando eles não forem necessários. Isso é aplicável a ambientes que não sejam de produção.
- Desalocar máquinas virtuais em vez de desligá-las. Isso liberará todos os recursos, e o medidor de consumo será interrompido.
- Usar laboratórios de desenvolvimento/teste para fins de desenvolvimento e testes. Eles fornecem políticas e recursos de desligamento automático e de inicialização automática.
- Com conjuntos de escalas de máquinas virtuais, começar com um número pequeno de máquinas virtuais e expandir quando a demanda aumentar.
- Escolher o tamanho correto (pequeno, médio ou grande) para os gateways de aplicações. Eles têm o suporte de máquinas virtuais e podem ajudar a reduzir custos.
- Escolher um gateway de aplicação da camada básica se o firewall da aplicação Web não for necessário.
- Escolher as camadas corretas para os gateways de VPN (VPN básica, padrão, de alta performance e de ultraperformance).
- Reduzir o tráfego de rede entre as regiões do Azure.
- Usar um balanceador de carga com um IP público para acessar várias máquinas virtuais em vez de atribuir um IP público a cada máquina virtual.
- Monitorar as máquinas virtuais e calcular suas métricas de uso e performance. Com base nesses cálculos, determine se deseja expandir ou reduzir suas máquinas virtuais. Isso pode resultar em downsizing e na redução do número de máquinas virtuais.

Considerar essas práticas recomendadas durante a arquitetura inevitavelmente resultará em economia de custos. Agora que abordamos a computação, vamos adotar uma abordagem semelhante para o armazenamento.

## Práticas recomendadas de armazenamento

Como estamos hospedando nossas aplicações na nuvem, o Armazenamento do Azure será usado para armazenar os dados relacionados a essas aplicações. Se não seguirmos as práticas certas, as coisas podem dar errado. Algumas das práticas recomendadas de armazenamento são:

- Escolher um tipo de redundância de armazenamento adequado (GRS, LRS, RA-GRS). GRS tem um custo mais alto do que LRS.

- Arquivar dados de armazenamento na camada de acesso esporádico ou na camada de acesso aos arquivos. Mantenha os dados acessados com frequência na camada de acesso frequente.
- Remover blobs que não são necessários.
- Excluir explicitamente os discos do sistema operacional das máquinas virtuais após excluir as máquinas virtuais, caso elas não sejam necessárias.
- As contas de armazenamento devem ser medidas com base em suas operações de tamanho, gravação, leitura, listagem e contêiner.
- Preferir discos standard em vez de discos premium; usar discos premium somente se a sua empresa exigir.
- Usar a CDN e o armazenamento em cache para arquivos estáticos em vez de buscá-los no armazenamento toda vez.
- O Azure oferece capacidade reservada para economizar custos em dados de blob.

Manter essas práticas recomendadas acessíveis ajudará você a arquitetar soluções de armazenamento econômicas. Na próxima seção, vamos discutir as práticas recomendadas envolvidas na implantação de serviços de PaaS.

## Práticas recomendadas de PaaS

Há muitos serviços de PaaS oferecidos pelo Azure e, se forem configurados incorretamente, você pode acabar com encargos inesperados na fatura. Para evitar esse cenário, você pode aproveitar as seguintes práticas recomendadas:

- Escolher uma camada de SQL do Azure adequada (Básica, Standard, Premium RS ou Premium) e os níveis de performance adequados.
- Escolher adequadamente entre um banco de dados individual e um banco de dados elástico. Se houver muitos bancos de dados, é mais econômico usar bancos de dados elásticos do que bancos de dados individuais.
- Rearquitetar sua solução para usar soluções de PaaS (sem servidor ou microsserviços com contêineres) em vez de soluções de IaaS. Essas soluções de PaaS reduzem os custos de manutenção e estão disponíveis em um consumo por minuto. Se você não consumir esses serviços, não haverá nenhum custo, apesar de seu código e seus serviços estarem sempre disponíveis.

Há otimizações de custos específicas de recursos, e não é possível abordar todas elas em apenas um capítulo. É recomendável ler a documentação relacionada ao custo e uso de cada recurso.

## Práticas recomendadas gerais

Até agora, analisamos as práticas recomendadas específicas de serviços e vamos encerrar esta seção com algumas diretrizes gerais:

- O custo dos recursos é diferente entre regiões. Experimente uma região alternativa, desde que não crie problemas de performance ou latência.
- Os EAs oferecem descontos melhores do que outras ofertas. Você pode falar com a Equipe de Contas da Microsoft e ver quais benefícios pode obter ao se inscrever em um EA.
- Se os custos do Azure puderem ser pré-pagos, você receberá descontos para todos os tipos de assinatura.
- Excluir ou remover recursos não utilizados. Descubra os recursos que são subutilizados e reduza seu SKU ou tamanho. Se eles não forem necessários, exclua-os.
- Usar o Assistente do Azure e levar suas recomendações a sério.

Como mencionado anteriormente, essas são algumas diretrizes genéricas e, à medida que você arquitetar mais soluções, poderá criar um conjunto de práticas recomendadas para você. Mas, para começar, você pode considerar essas. De qualquer forma, cada componente no Azure tem suas próprias práticas recomendadas e consultar a documentação enquanto você está arquitetando ajudará a criar uma solução econômica.

## Resumo

Neste capítulo, aprendemos a importância do gerenciamento de custos e da administração ao trabalhar em um ambiente de nuvem. Também abordamos as várias opções de preços do Azure e os vários recursos de otimização de preços que o Azure oferece. O gerenciamento de custos de um projeto é fundamental principalmente porque a despesa mensal pode ser muito baixa, mas pode aumentar se os recursos não forem monitorados de forma periódica. Os arquitetos de nuvem devem projetar suas aplicações de forma econômica. Eles devem usar os recursos do Azure, os SKUs, as camadas e os tamanhos adequados e saber quando iniciar, parar, escalar verticalmente e horizontalmente, transferir dados e muito mais. O gerenciamento de custos adequado garantirá que as despesas reais correspondam às despesas orçamentárias.

No próximo capítulo, analisaremos vários recursos do Azure relacionados a serviços de dados, como SQL do Azure, Cosmos DB e fragmentação.



# 7

## Soluções OLTP do Azure

O Azure fornece os serviços **Infraestrutura como Serviço (IaaS)** e **Plataforma como Serviço (PaaS)**. Esses tipos de serviços fornecem diferentes níveis e controles sobre armazenamento, computação e redes às organizações. Armazenamento é o recurso usado ao trabalhar com o armazenamento e a transmissão de dados. O Azure fornece muitas opções para armazenar dados, como Armazenamento de Blobs do Azure, Armazenamento de tabelas, Cosmos DB, Banco de Dados SQL do Azure, Azure Data Lake Storage e muito mais. Embora algumas dessas opções sejam destinadas a armazenamento, análise e apresentação de big data, há outras que são destinadas a aplicações que processam transações. O SQL do Azure é o recurso principal no Azure que funciona com dados transacionais.

Este capítulo se concentrará em vários aspectos do uso de repositórios de dados transacionais, como Banco de Dados SQL do Azure e outros bancos de dados open source normalmente usados em sistemas de **Processamento de Transações Online (OLTP)**, e abordará os seguintes tópicos:

- Aplicações OLTP
- Bancos de dados relacionais
- Modelos de implantação
- Banco de dados SQL do Azure
- Instância Única
- Pools elásticos
- Instância Gerenciada
- Cosmos DB

Iniciaremos este capítulo analisando o que são aplicações OLTP e listando os serviços OLTP do Azure e seus casos de uso.

## Aplicações OLTP

Como mencionado anteriormente, as aplicações OLTP são aplicações que ajudam no processamento e gerenciamento de transações. Algumas das implementações OLTP mais prevalentes podem ser encontradas em vendas no varejo, sistemas de transações financeiras e entrada de ordens. Essas aplicações executam captura, processamento, recuperação, modificação e armazenamento de dados. No entanto, não param por aqui. As aplicações OLTP tratam essas tarefas de dados como transações. As transações têm algumas propriedades importantes e as aplicações OLTP são responsáveis por essas propriedades. Essas propriedades são agrupadas com o acrônimo **ACID**. Vamos discuti-las em detalhes:

- **Atomicidade:** esta propriedade informa que uma transação deve consistir em instruções e que todas as instruções devem ser concluídas com êxito ou nenhuma instrução deve ser executada. Se várias instruções forem agrupadas, elas formarão uma transação. Atomicidade significa que cada transação é tratada como a menor unidade única de execução que é concluída com êxito ou falha.
- **Consistência:** esta propriedade se concentra no estado dos dados em um banco de dados. Ela determina que qualquer alteração no estado deve ser completa e baseada nas regras e restrições do banco de dados, e que atualizações parciais não devem ser permitidas.

- **Isolamento:** esta propriedade informa que pode haver várias transações simultâneas executadas em um sistema e que cada transação deve ser tratada isoladamente. Uma transação não deve saber sobre a outra ou interferir em nenhuma outra transação. Se as transações tiverem que ser executadas em sequência, no final, o estado dos dados deverá ser o mesmo que antes.
- **Durabilidade:** esta propriedade informa que os dados devem ser persistentes e estar disponíveis, mesmo após uma falha, depois de confirmados no banco de dados. Uma transação confirmada torna-se um fato.

Agora que você sabe o que são aplicações OLTP, vamos discutir a função dos bancos de dados relacionais nessas aplicações.

## Bancos de dados relacionais

As aplicações OLTP geralmente dependem de bancos de dados relacionais para gerenciamento e processamento de suas transações. Os bancos de dados relacionais geralmente são fornecidos em um formato tabular, que consiste em linhas e colunas. O modelo de dados é convertido em várias tabelas, em que cada tabela é conectada a outra tabela (com base em regras) usando relacionamentos. Esse processo também é conhecido como normalização.

Há vários serviços no Azure que oferecem suporte a aplicações OLTP e à implantação de bancos de dados relacionais. Na próxima seção, vamos analisar os serviços no Azure que estão relacionados a aplicações OLTP.

## Serviços de nuvem do Azure

Uma pesquisa por **sql** no portal do Azure fornece vários resultados. Marquei alguns deles para mostrar os recursos que podem ser usados diretamente para aplicações OLTP:

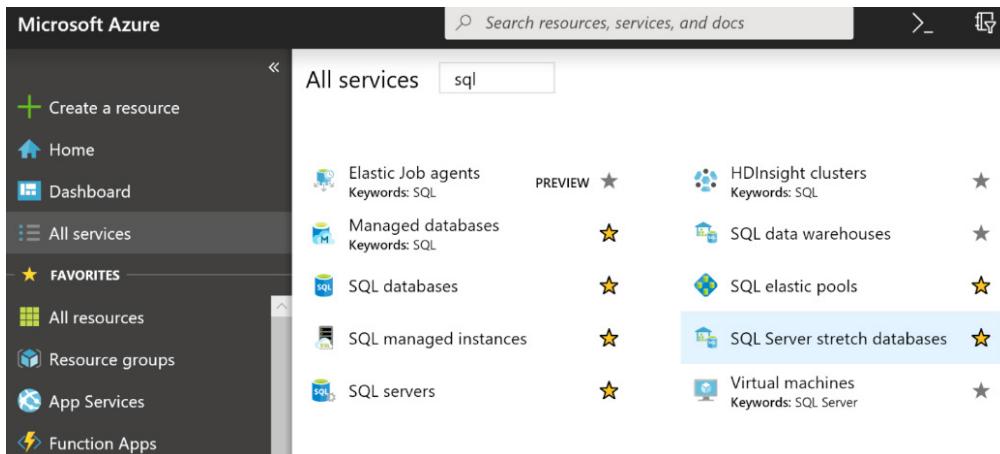


Figura 7.1: lista de serviços do SQL do Azure

A Figura 7.1 mostra os diversos recursos e opções disponíveis para a criação de bancos de dados baseados no SQL Server no Azure.

Novamente, uma pesquisa rápida por **banco de dados** no portal do Azure fornece vários recursos e aqueles marcados na Figura 7.2 podem ser usados para aplicações OLTP:

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with options like 'Create a resource', 'Home', 'Dashboard', 'All services' (which is selected), 'FAVORITES', 'All resources', 'Resource groups', 'App Services', and 'Function Apps'. The main area has a search bar at the top with the placeholder 'Search resources, services, and docs'. Below the search bar, there are two tabs: 'All services' and 'database', with 'database' being the active tab. The results list includes several items with star icons and preview status:

- Azure Cosmos DB (Keywords: Database)
- Azure Database for MySQL servers
- Azure Database Migration Services
- Free services (Keywords: free database)
- SQL databases (Keywords: Stretch Database) - This item is highlighted with a blue selection bar.
- Azure Database for MariaDB servers
- Azure Database for PostgreSQL servers
- Elastic Job agents (Keywords: database) - PREVIEW
- Managed databases (Keywords: database)
- SQL Server stretch databases (Keywords: Stretch Database)

Figura 7.2: lista de serviços do Azure usados para aplicações OLTP

A Figura 7.2 mostra os recursos fornecidos pelo Azure que podem hospedar dados em vários bancos de dados, incluindo os seguintes:

- Bancos de dados MySQL
- Bancos de dados MariaDB
- Bancos de dados PostgreSQL
- Cosmos DB

A seguir, vamos discutir os modelos de implantação.

## Modelos de implantação

Os modelos de implantação no Azure são classificados com base no nível de gerenciamento ou controle. Cabe ao usuário selecionar qual nível de gerenciamento ou controle ele prefere; ele pode optar por controle total usando serviços como Máquinas Virtuais ou pode usar serviços gerenciados em que as coisas serão gerenciadas pelo Azure para ele.

Há dois modelos de implantação para implantar bancos de dados no Azure:

- Bancos de dados em Máquinas Virtuais do Azure (IaaS)
- Bancos de dados hospedados como serviços gerenciados (PaaS)

Agora vamos tentar entender a diferença entre a implantação em Máquinas Virtuais do Azure e em instâncias gerenciadas. Vamos começar com as Máquinas Virtuais.

## Bancos de dados em Máquinas Virtuais do Azure

O Azure fornece várias **unidades de manutenção de estoque (SKUs)** para máquinas virtuais. Há máquinas de alta computação e alta taxa de transferência (IOPS) que também estão disponíveis junto com máquinas virtuais de uso geral. Em vez de hospedar um SQL Server, MySQL ou qualquer outro banco de dados em servidores na infraestrutura local, é possível implantar esses bancos de dados nessas máquinas virtuais. A implantação e a configuração desses bancos de dados não são diferentes das implantações na infraestrutura local. A única diferença é que o banco de dados é hospedado na nuvem, em vez de usar servidores na infraestrutura local.

Os administradores devem executar as mesmas atividades e etapas que normalmente executariam para uma implantação na infraestrutura local. Embora esta opção seja ideal quando os clientes desejam ter controle total sobre sua implantação, há modelos que podem ser mais econômicos, escalonáveis e altamente disponíveis em comparação a essa opção, que serão discutidos posteriormente neste capítulo.

As etapas para implantar qualquer banco de dados em Máquinas Virtuais do Azure são as seguintes:

1. Crie uma máquina virtual com um tamanho que atenda aos requisitos de performance da aplicação.
2. Implante um banco de dados sobre ela.
3. Defina a configuração da máquina virtual e do banco de dados.

Esta opção não fornece nenhuma alta disponibilidade pronta para uso, a menos que vários servidores sejam provisionados. Ela também não fornece recursos para escala automática, a menos que a automação personalizada dê suporte a ela.

A disaster recovery também é responsabilidade do cliente. Os servidores devem ser implantados em várias regiões conectadas por meio de serviços como emparelhamento global, gateways de VPN, ExpressRoute ou WAN Virtual. É possível que essas máquinas virtuais sejam conectadas a um datacenter na infraestrutura local por meio de VPNs site a site ou do ExpressRoute sem ter nenhuma exposição ao mundo externo.

Esses bancos de dados também são conhecidos como **bancos de dados não gerenciados**. Por outro lado, os bancos de dados hospedados com o Azure, diferentes das máquinas virtuais, são gerenciados pelo Azure e são conhecidos como **serviços gerenciados**. Na próxima seção, vamos abordá-los em detalhes.

## Bancos de dados hospedados como serviços gerenciados

Serviços gerenciados significam que o Azure fornece serviços de gerenciamento para os bancos de dados. Esses serviços gerenciados incluem a hospedagem do banco de dados, garantindo que o host esteja altamente disponível, garantindo que os dados sejam replicados internamente para disponibilidade durante a disaster recovery, garantindo a escalabilidade dentro da restrição de um SKU escolhido, monitorando os hosts e bancos de dados e gerando alertas para notificações ou executando ações, fornecendo serviços de log e auditoria para solução de problemas e cuidando do gerenciamento de performance e dos alertas de segurança.

Em suma, há muitos serviços que os clientes obtêm prontos para uso quando utilizam serviços gerenciados do Azure e eles não precisam executar o gerenciamento ativo nesses bancos de dados. Neste capítulo, vamos analisar o Banco de Dados SQL do Azure em detalhes e fornecer informações sobre outros bancos de dados, como MySQL e Postgre. Além disso, vamos abordar os bancos de dados não relacionais, como Cosmos DB, que é um banco de dados NoSQL.

## Banco de Dados SQL do Azure

O SQL do Azure Server fornece um banco de dados relacional hospedado como uma PaaS. Os clientes podem provisionar esse serviço, trazer seus próprios dados e esquemas de banco de dados e conectar suas aplicações a ele. Ele fornece todos os recursos do SQL Server quando implantado em uma máquina virtual. Esses serviços não fornecem uma interface de usuário para criar tabelas e seus esquemas correspondentes, nem fornecem recursos de consulta diretamente. O SQL Server Management Studio e as ferramentas da CLI do SQL devem ser usados para se conectar a esses serviços e trabalhar diretamente com eles.

O Banco de Dados SQL do Azure é fornecido com três modelos de implantação distintos:

- **Instância Única:** neste modelo de implantação, um único banco de dados é implantado em um servidor lógico. Isso envolve a criação de dois recursos no Azure: um servidor lógico do SQL e um banco de dados SQL.
- **Pool elástico:** neste modo de implantação, vários bancos de dados são implantados em um servidor lógico. Novamente, isso envolve a criação de dois recursos no Azure: um servidor lógico do SQL e um pool de banco de dados elástico SQL, que contém todos os bancos de dados.
- **Instância Gerenciada:** este é um modelo de implantação relativamente novo da equipe de SQL do Azure. Esta implantação reflete uma coleção de bancos de dados em um servidor lógico, fornecendo controle total sobre os recursos no que diz respeito aos bancos de dados do sistema. Geralmente, os bancos de dados do sistema não são visíveis em outros modelos de implantação, mas estão disponíveis no modelo. Este modelo é muito parecido com a implantação do SQL Server na infraestrutura local:

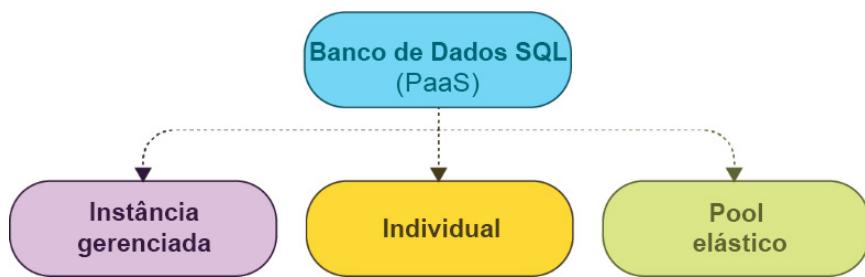


Figura 7.3: modelos de implantação do Banco de Dados SQL do Azure

Se você estiver se perguntando quando usar o quê, analise uma comparação de recursos entre o Banco de Dados SQL e a Instância Gerenciada do SQL. Uma comparação completa de recursos está disponível em <https://docs.microsoft.com/azure/azure-sql/database/features-comparison>.

A seguir, vamos abordar alguns dos recursos do Banco de Dados SQL. Vamos começar com os recursos da aplicação.

## Recursos da aplicação

O Banco de Dados SQL do Azure fornece vários recursos específicos da aplicação que atendem aos diferentes requisitos dos sistemas OLTP:

- **Armazenamento colunar:** este recurso permite o armazenamento de dados em um formato colunar, em vez de um formato de linha.
- **OLTP in-memory:** geralmente, os dados são armazenados em arquivos de backend no SQL e são obtidos desses arquivos quando necessários para a aplicação. Por outro lado, o OLTP in-memory coloca todos os dados na memória e não há latência na leitura do armazenamento de dados. O armazenamento de dados OLTP in-memory no SSD proporciona a melhor performance possível para o SQL do Azure.
- Todos os recursos do SQL Server na infraestrutura local.

O próximo recurso que vamos discutir é a alta disponibilidade.

## Alta disponibilidade

Por padrão, o SQL do Azure é 99,99% altamente disponível. Ele tem duas arquiteturas diferentes para manter a alta disponibilidade com base em SKUs. Para os SKUs Básicos, Padrão e Gerais, toda a arquitetura é dividida nas duas camadas a seguir.

- Camada de computação
- Camada de armazenamento

Há redundância incorporada para essas duas camadas fornecerem alta disponibilidade:

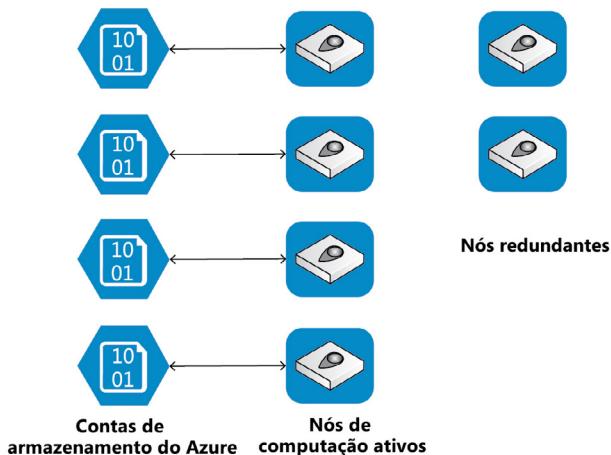


Figura 7.4: camadas de computação e armazenamento em SKUs padrão

Para os SKUs Premium e críticos para os negócios, a computação e o armazenamento estão na mesma camada. A alta disponibilidade é alcançada pela replicação de computação e armazenamento implantada em um cluster de quatro nós, usando tecnologia semelhante aos grupos de disponibilidade AlwaysOn do SQL Server:

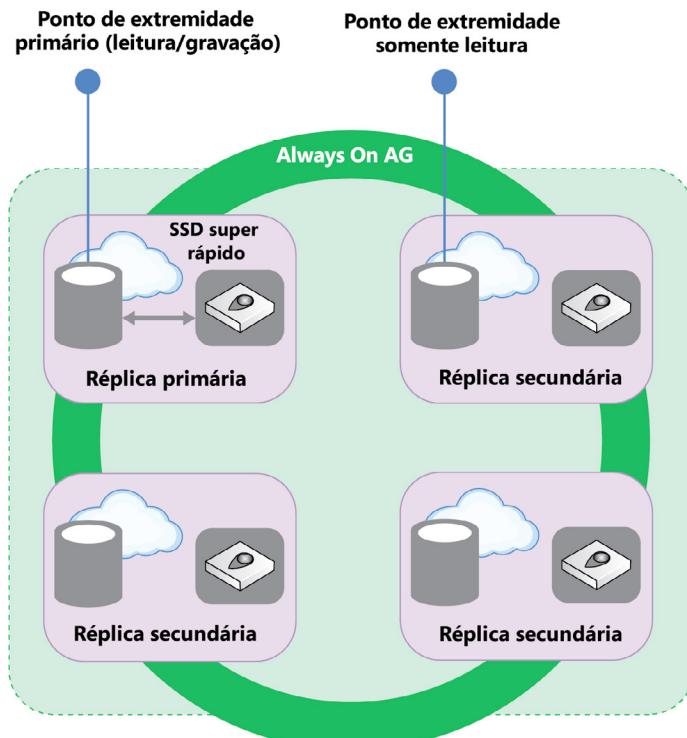


Figura 7.5: implantação em um cluster de quatro nós

Agora que você sabe como a alta disponibilidade é tratada, vamos avançar para o próximo recurso: backups.

## Backups

O Banco de Dados SQL do Azure também fornece recursos para fazer backup automático de bancos de dados e armazená-los em contas de armazenamento. Este recurso é importante principalmente nos casos em que um banco de dados se torna corrompido ou um usuário exclui uma tabela de forma acidental. Ele está disponível no nível do servidor, conforme mostrado na Figura 7.6:

Figura 7.6: fazendo backup de bancos de dados no Azure

Os arquitetos devem preparar uma estratégia de backup para que os backups possam ser usados em momentos de necessidade. Ao configurar backups, garanta que a frequência deles não seja nem muito baixa nem muito alta. Com base nas necessidades da empresa, um backup semanal ou até mesmo um backup diário deve ser configurado, ou ainda mais frequentemente do que isso, se necessário. Esses backups podem ser usados para fins de restauração.

Os backups ajudarão na continuidade dos negócios e na recuperação de dados. Você também pode usar a replicação geográfica para recuperar os dados durante uma falha regional. Na próxima seção, vamos abordar a replicação geográfica.

## Replicação geográfica

O Banco de Dados SQL do Azure também fornece o benefício de ser capaz de replicar um banco de dados para uma região diferente, também conhecida como região secundária; isso é totalmente baseado no plano que você está escolhendo. O banco de dados na região secundária pode ser lido por aplicações. O Banco de Dados SQL do Azure permite bancos de dados secundários legíveis. Esta é uma ótima solução de continuidade dos negócios, pois um banco de dados legível está disponível a qualquer momento. Com a replicação geográfica, é possível ter até quatro regiões secundárias de um banco de dados em diferentes regiões ou na mesma região. Com a replicação geográfica, também é possível fazer o failover para um banco de dados secundário em caso de desastres. A replicação geográfica é configurada no nível do banco de dados, conforme mostrado na Figura 7.7:

The screenshot shows the Azure portal interface for managing a SQL database named 'myecommerce'. The left sidebar navigation bar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Quick start, Query editor (preview), Settings, Configure, Geo-Replication (which is selected and highlighted in blue), Connection strings, Sync to other databases, Add Azure Search, Properties, Locks, Automation script, Security, Advanced Threat Protection, and Auditing.

The main content area displays a world map with various green and blue circular icons representing different regions or servers. A prominent message at the top right states: "Select a region on the map or from the Target Regions list to create a secondary database." Below this, another message indicates: "You can now automatically manage replication, connectivity and failover of this database by adding it to failover group." At the bottom of the screen, there is a table with three columns: SERVER/DATABASE, FAILOVER POLICY, and STATUS. The table shows one entry under the PRIMARY column: "West Europe" and "shardserver1/myecommerce". The FAILOVER POLICY column shows "None" and the STATUS column shows "Online".

Figura 7.7: replicação geográfica no Azure

Se você rolar para baixo nesta tela, as regiões que podem agir como secundárias são listadas, conforme mostrado na Figura 7.8:

Home > SQL databases > myecommerce - Geo-Replication

## myecommerce - Geo-Replication

SQL database

Search (Ctrl+ /)

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Quick start
- Query editor (preview)

**Settings**

- Configure
- Geo-Replication**
- Connection strings
- Sync to other databases
- Add Azure Search
- Properties
- Locks
- Automation script

**Security**

- Advanced Threat Protection
- Auditing

Select a region on the map or from the Target Regions list to create a secondary database.

SERVER/DATABASE	FAILOVER POLICY	STATUS
<b>PRIMARY</b> West Europe shardserver1/myecommerce	None	Online
<b>SECONDARIES</b> <i>Geo-Replication is not configured</i>		
<b>TARGET REGIONS</b>		
West US		
West US 2		
Central US		
West Central US		
South Central US		
North Central US		
Canada Central		
East US		
Canada East		

Figura 7.8: lista de regiões secundárias disponíveis para replicação geográfica

Antes de arquitetar soluções que envolvam replicação geográfica, precisamos validar os regulamentos de residência e conformidade de dados. Se os dados dos clientes não tiverem permissão para serem armazenados fora de uma região por motivos de conformidade, não devemos replicá-los para outras regiões.

Na próxima seção, vamos explorar opções de escalabilidade.

## Escalabilidade

O Banco de Dados SQL do Azure fornece escalabilidade vertical adicionando mais recursos (como computação, memória e IOPS). Isso pode ser feito aumentando o número de **Unidades de Taxa de Transferência de Banco de Dados (DTUs)** ou de recursos de computação e armazenamento no caso do modelo **vCore**:

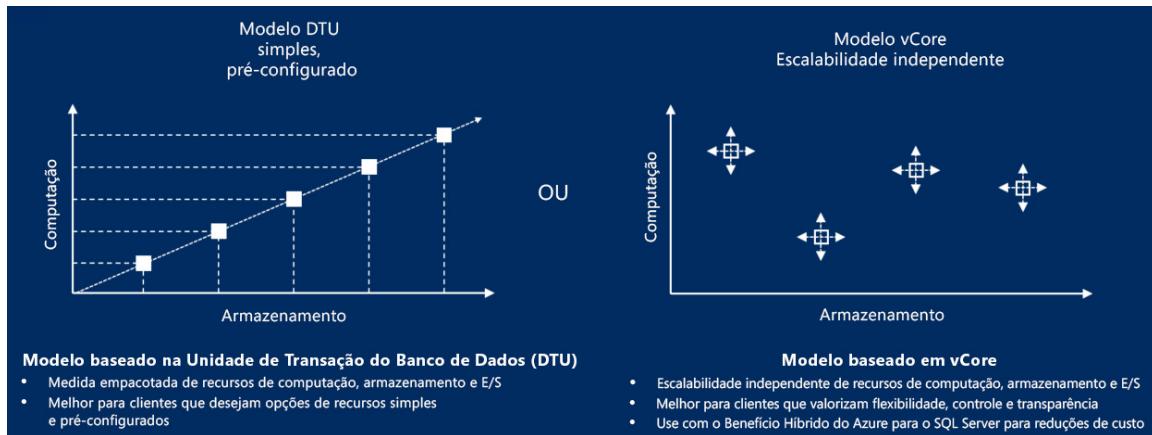


Figura 7.9: escalabilidade no Banco de Dados SQL do Azure

Abordamos as diferenças entre o modelo baseado em DTU e o modelo baseado em vCore posteriormente neste capítulo.

Na próxima seção, abordaremos a segurança, o que ajudará você a entender como criar soluções de dados seguras no Azure.

## Segurança

A segurança é um fator importante para qualquer serviço e solução de banco de dados. O SQL do Azure fornece segurança de nível empresarial para o SQL do Azure, e esta seção listará alguns dos recursos de segurança importantes no SQL do Azure.

### Firewall

Por padrão, o Banco de Dados SQL do Azure não fornece acesso a nenhuma solicitação. Os endereços IP de origem devem ser explicitamente aceitos para acessar o SQL Server. Há uma opção para permitir que todos os serviços baseados no Azure também accessem um banco de dados SQL. Essa opção inclui máquinas virtuais hospedadas no Azure.

O firewall pode ser configurado no nível do servidor, em vez do nível do banco de dados. A opção Permitir o acesso aos serviços do Azure permite que todos os serviços, incluindo máquinas virtuais, accessem um banco de dados hospedado em um servidor lógico.

Por padrão, essa opção será desativada por motivos de segurança; habilitá-la permitiria o acesso de todos os serviços do Azure:

Home > sharedserver1 (sharedserver1/sharedserver1) >

## Firewall settings

sharedserver1 (SQL server)

Save Discard + Add client IP

Deny public network access ⓘ Yes No

**i** Setting to **Yes** allows connections via approved private endpoint only and disables any existing firewall rules. [Learn more](#).

Minimal TLS Version ⓘ > 1.0 >1.1 >1.2

**i** You are setting the Minimal TLS Version property for all SQL Database and SQL Data Warehouse databases associated with the server. Any login attempts from clients using TLS version less than the Minimal TLS Version shall be rejected.

Connection Policy ⓘ Default Proxy Redirect

Allow Azure services and resources to access this server Yes No

**i** Connections from the IPs specified below provides access to all the databases in sharedserver1.

Client IP address	117.210.191.108		
Rule name	Start IP	End IP	...
			...
ClientIPAddress_2020-6-19...	117.210.192.205	117.210.192.205	...
ClientIPAddress_2020-6...	117.210.191.109	117.210.191.109	...

Figura 7.10: configurando um firewall no nível do servidor no Azure

## SQL do Azure Server em redes dedicadas

Embora o acesso ao SQL Server esteja disponível para o público em geral pela Internet, é possível que ele seja limitado a solicitações provenientes de redes virtuais. Este é um recurso relativamente novo no Azure. Ele ajuda no acesso a dados no SQL Server de uma aplicação em outro servidor da rede virtual sem a solicitação passar pela Internet.

Para isso, um ponto de extremidade de serviço do tipo **Microsoft.Sql** deve ser adicionado dentro da rede virtual, e a rede virtual deve estar na mesma região do Banco de Dados SQL do Azure:

The screenshot shows the Azure portal interface for managing service endpoints in a virtual network named 'azureclitest-vnet'. The left sidebar lists various options like Firewall, Security, DNS servers, Peerings, Service endpoints (which is selected), Private endpoints, Properties, Locks, and Export template. Below this is a 'Monitoring' section with Diagnostic settings. The main content area is titled 'Add service endpoints' and shows a search bar and a 'Filter services' dropdown containing a list of available services. The 'Service' dropdown is set to 'Microsoft.Sql', and the 'Subnet' dropdown is empty. A large blue 'Add' button is at the bottom right of the form.

Figura 7.11: adicionando um ponto de extremidade de serviço Microsoft.Sql

Uma sub-rede apropriada dentro da rede virtual deve ser escolhida:

The screenshot shows the Azure portal interface for managing service endpoints in a virtual network. The left sidebar lists various settings like Address space, Connected devices, Subnets, DDoS protection, Firewall, DNS servers, Peerings, and Service endpoints. The 'Service endpoints' option is selected, highlighted with a blue background. The main pane displays a table with columns 'SERVICE' and 'SUBNET'. A search bar at the top right says 'Filter service endpoints' and has 'Microsoft.Sql' selected in a dropdown. Below the table, it says 'No service endpoints.' A large callout box provides information about selecting a subnet for the Microsoft.Sql service. It states: 'rules using Azure public IP addresses will stop working with this switch. Please ensure IP firewall rules allow for this switch before setting up service endpoints. You may also experience temporary interruption to service traffic from this subnet while configuring service endpoints.' There are checkboxes for 'Select all' and 'default', both of which are checked.

Figura 7.12: escolhendo uma sub-rede para o serviço Microsoft.Sql

Por fim, na folha de configuração do SQL do Azure Server, deve ser adicionada uma rede virtual existente que tenha um ponto de extremidade de serviço **Microsoft.Sql** habilitado:

The screenshot shows the Azure portal interface for managing a SQL server's firewalls and virtual networks. On the left, a sidebar lists various management options like SQL databases, elastic pools, and security features. The 'Firewalls and virtual networks' option is selected, indicated by a blue background. The main content area displays a 'Create/Update' dialog for a new virtual network rule. The rule is named 'newVnetRule1' and is configured to allow connections from the subnet 'default / 10.1.0.0/24' of the virtual network 'azureclitest-vnet'. The 'OK' button is visible at the bottom right of the dialog.

Figura 7.13: adicionando uma rede virtual com o ponto de extremidade de serviço Microsoft.Sql

## Bancos de dados criptografados em repouso

Os bancos de dados devem estar em um formato criptografado quando estiverem em repouso. **Em repouso** aqui significa que os dados estão no local de armazenamento do banco de dados. Embora você possa não ter acesso ao SQL Server e seu banco de dados, é preferível criptografar o armazenamento do banco de dados.

Os bancos de dados em um sistema de arquivos podem ser criptografados usando chaves. Essas chaves devem ser armazenadas no Azure Key Vault e o cofre deve estar disponível na mesma região do SQL do Azure Server. O sistema de arquivos pode ser criptografado usando o item de menu **Transparent data encryption** da folha de configuração do SQL Server e selecionando **Sim** para **Usar sua própria chave**.

A chave é uma chave RSA 2048 e deve existir no cofre. O SQL Server vai descriptografar os dados no nível da página quando quiser lê-los e enviá-los para o chamador; em seguida, ele vai criptografá-los depois de gravá-los no banco de dados. Nenhuma alteração nas aplicações é necessária e isso é completamente transparente para elas:

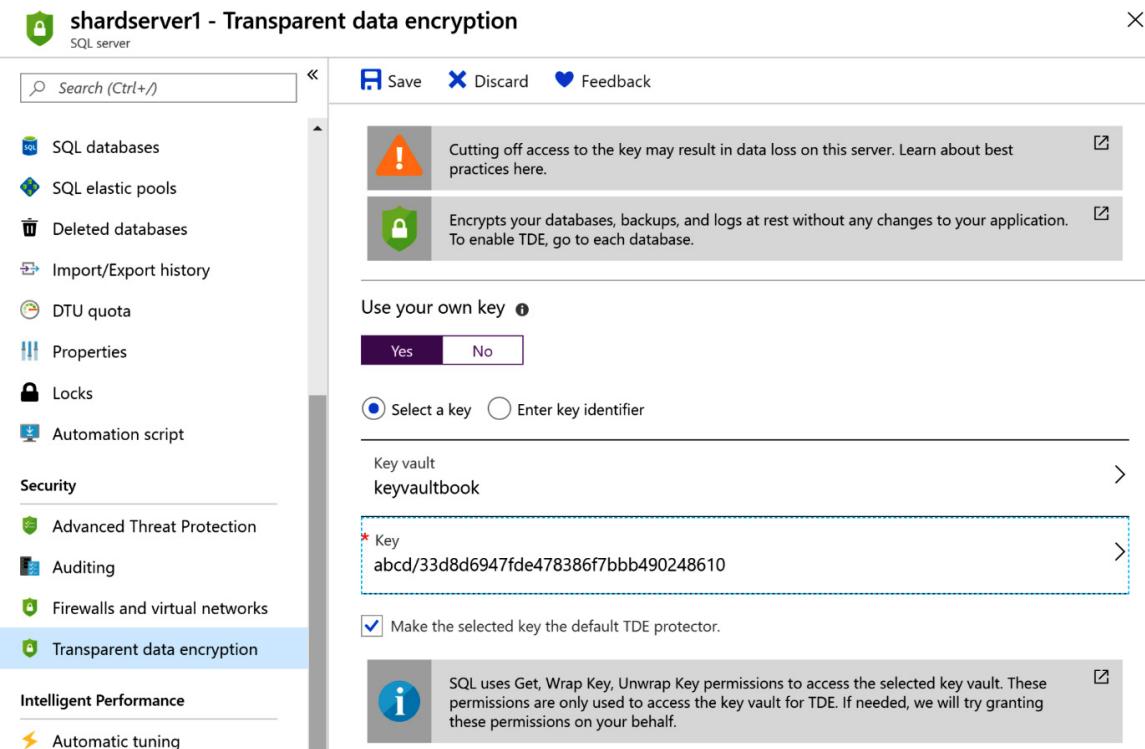


Figura 7.14: transparent data encryption no SQL Server

## Máscara dinâmica de dados

O SQL Server também fornece um recurso que mascara colunas individuais que contêm dados confidenciais para que ninguém, além dos usuários privilegiados, possa exibir os dados reais consultando-os no SQL Server Management Studio. Os dados permanecerão mascarados e só serão desmascarados quando uma aplicação ou usuário autorizado consultar a tabela. Os arquitetos devem garantir que dados confidenciais, como dados de cartão de crédito, números de CPF, números de telefone, endereços de email e outros dados financeiros, sejam mascarados.

As regras de mascaramento podem ser definidas em uma coluna em uma tabela. Há quatro tipos principais de máscaras, você pode conferi-los aqui: <https://docs.microsoft.com/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver15#define-a-dynamic-data-mask>.

A Figura 7.15 mostra como a máscara de dados é adicionada:

MASK NAME	MASK FUNCTION
dbo_Customers_FirstName	Default value (0, xxxx, 01-01-1900)

SCHEMA	TABLE	COLUMN	
dbo	Customers	LastName	Add mask
dbo	Salary	SalaryID	Add mask

Figura 7.15: máscara dinâmica de dados no Banco de Dados SQL

## Integração do Azure Active Directory

Outro importante recurso de segurança do SQL do Azure é que ele pode ser integrado ao Azure **Active Directory (AD)** para fins de autenticação. Sem integração ao Azure AD, o único mecanismo de autenticação disponível para o SQL Server é via autenticação de nome de usuário e senha, ou seja, a autenticação SQL. Não é possível usar a autenticação integrada do Windows. A cadeia de conexão para autenticação SQL consiste em nome de usuário e senha em texto não criptografado, o que não é seguro. A integração ao Azure AD permite a autenticação de aplicações com a autenticação do Windows, um nome da entidade de serviço ou autenticação baseada em token. É recomendável usar o Banco de Dados SQL do Azure integrado ao Azure AD.

Há outros recursos de segurança, como proteção avançada contra ameaças, auditoria do ambiente e monitoramento, que devem ser habilitados em todas as implantações do Banco de Dados SQL do Azure de nível empresarial.

Com isso, concluímos nossa análise dos recursos do Banco de Dados SQL do Azure e agora podemos avançar para os tipos de bancos de dados SQL.

## Instância Única

Os bancos de dados de Instância Única são hospedados como um banco de dados único em um servidor lógico único. Esses bancos de dados não têm acesso aos recursos completos fornecidos pelo SQL Server. Cada banco de dados é isolado e portátil. As instâncias únicas oferecem suporte aos modelos de compra baseados em vCPU e DTU que discutimos anteriormente.

Outra vantagem adicional de um banco de dados único é a eficiência de custos. Se você estiver em um modelo baseado em vCore, poderá optar por recursos de computação e armazenamento mais baixos para otimizar os custos. Se você precisar de mais capacidade de computação ou armazenamento, sempre poderá escalar. A escalabilidade dinâmica é um recurso proeminente de instâncias únicas que ajuda a escalar recursos dinamicamente com base em requisitos de negócios. As instâncias únicas permitem que os clientes existentes do SQL Server façam o "lift and shift" de suas aplicações na infraestrutura local para a nuvem.

Outros recursos incluem disponibilidade, monitoramento e segurança.

Quando iniciamos nossa seção sobre o Banco de Dados SQL do Azure, também mencionamos os pools elásticos. Você também pode fazer a transição de um banco de dados individual para um pool elástico para compartilhamento de recursos. Se você estiver se perguntando o que são compartilhamento de recursos e pools elásticos, na próxima seção, abordaremos isso.

## Pools elásticos

Um pool elástico é um contêiner lógico que pode hospedar vários bancos de dados em um único servidor lógico. Os pools elásticos estão disponíveis nos modelos de compra baseados em vCore e DTU. O modelo de compra baseado em vCPU é o método padrão e recomendado de implantação, em que você terá a liberdade de escolher seus recursos de computação e armazenamento com base nos seus workloads de negócios. Conforme mostrado na Figura 7.16, você pode selecionar a quantidade de núcleos e armazenamento necessário para o seu banco de dados:

[Home](#) > [SQL elastic pools](#) > [Create SQL Elastic pool](#) >

**Configure**

[Feedback](#)

[Looking for basic, standard, premium?](#)

General Purpose	Business Critical
Scalable compute and storage options 500 - 20,000 IOPS 2-10 ms latency Starting at 8919.43 INR / month	High transaction rate and high resiliency 5,000 - 204,800 IOPS 1-2 ms latency Starting at 35656.26 INR / month

**Pool settings**   [Databases](#)   [Per database settings](#)

**Compute Hardware**  
Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

**Hardware Configuration**  
[Gen5](#) up to 80 vCores, up to 408 GB memory  
[Change configuration](#)

**vCores** [How do vCores compare with DTUs?](#)  
8 vCores

**Data max size** 1.5 TB   100 GB

**Cost summary**

Gen5 - General Purpose	8907.57
Cost per vCore (in INR)	x 8
Cost per GB (in INR)	9.12
Max storage selected (in GB)	x 130
ESTIMATED COST / MONTH	72446.34 INR

**Apply**

Figura 7.16: configurando pools elásticos no modelo baseado em vCore

Além disso, na parte superior da figura anterior, há uma opção que diz **Procurando básico, padrão, premium? (Looking for basic, standard, premium?)** Se você selecioná-la, o modelo será alternado para o modelo de DTU.

Os SKUs disponíveis para pools elásticos no modelo baseado em DTU são os seguintes:

- Básico
- Padrão
- Premium

A Figura 7.17 mostra a quantidade máxima de DTUs que podem ser provisionadas para cada SKU:

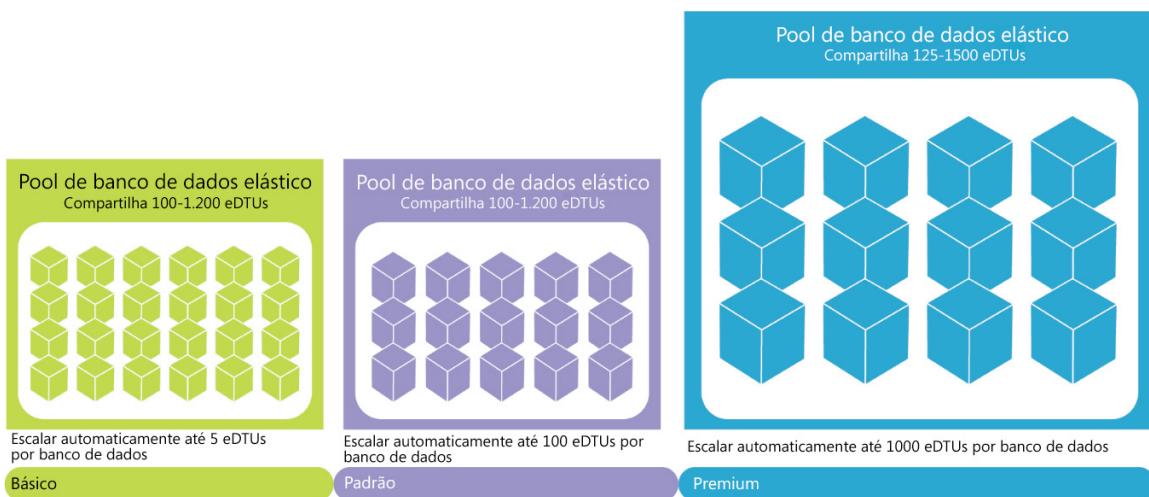


Figura 7.17: quantidade de DTUs por SKU em um pool elástico

Todos os recursos discutidos para as instâncias únicas do SQL do Azure também estão disponíveis para pools elásticos. No entanto, a escalabilidade horizontal é um recurso adicional que permite a fragmentação. A fragmentação se refere ao particionamento vertical ou horizontal de dados e ao armazenamento desses dados em bancos de dados separados. Também é possível ter escala automática de bancos de dados individuais em um pool elástico consumindo mais DTUs do que as alocadas para esse banco de dados.

Os pools elásticos também proporcionam outra vantagem em termos de custo. Você verá em uma seção posterior que o Banco de Dados SQL do Azure é precificado usando DTUs, e as DTUs são provisionadas assim que o serviço do SQL Server é provisionado. As DTUs são cobradas independentemente de serem consumidas ou não. Se houver vários bancos de dados, será possível colocá-los em pools elásticos e eles compartilharem as DTUs entre si.

Todas as informações para implementar a fragmentação com pools elásticos do SQL do Azure foram fornecidas em <https://docs.microsoft.com/azure/sql-database/sql-database-elastic-scale-introduction>.

A seguir, vamos discutir a opção de implantação da Instância Gerenciada, que é um banco de dados escalonável, inteligente, baseado na nuvem e totalmente gerenciado.

## Instância Gerenciada

A Instância Gerenciada é um serviço exclusivo que fornece um servidor SQL gerenciado semelhante ao que está disponível nos servidores na infraestrutura local. Os usuários têm acesso ao mestre, modelo e outros bancos de dados do sistema. A Instância Gerenciada é ideal quando há vários bancos de dados e clientes migrando suas instâncias para o Azure. A Instância Gerenciada consiste em vários bancos de dados.

O Banco de Dados SQL do Azure fornece um novo modelo de implantação conhecido como Instância Gerenciada do Banco de Dados SQL do Azure que oferece quase 100% de compatibilidade com o Mecanismo de Banco de Dados da Edição Enterprise do SQL Server. Esse modelo fornece uma implementação de rede virtual nativa que aborda os problemas de segurança habituais e é um modelo de negócios altamente recomendado para clientes do SQL Server na infraestrutura local. A Instância Gerenciada permite que os clientes existentes do SQL Server façam o "lift and shift" de suas aplicações na infraestrutura local para a nuvem com alterações mínimas em aplicações e bancos de dados, preservando todos os recursos de PaaS ao mesmo tempo. Esses recursos de PaaS reduzem drasticamente a sobrecarga de gerenciamento e o custo total de propriedade, conforme mostrado na Figura 7.18:

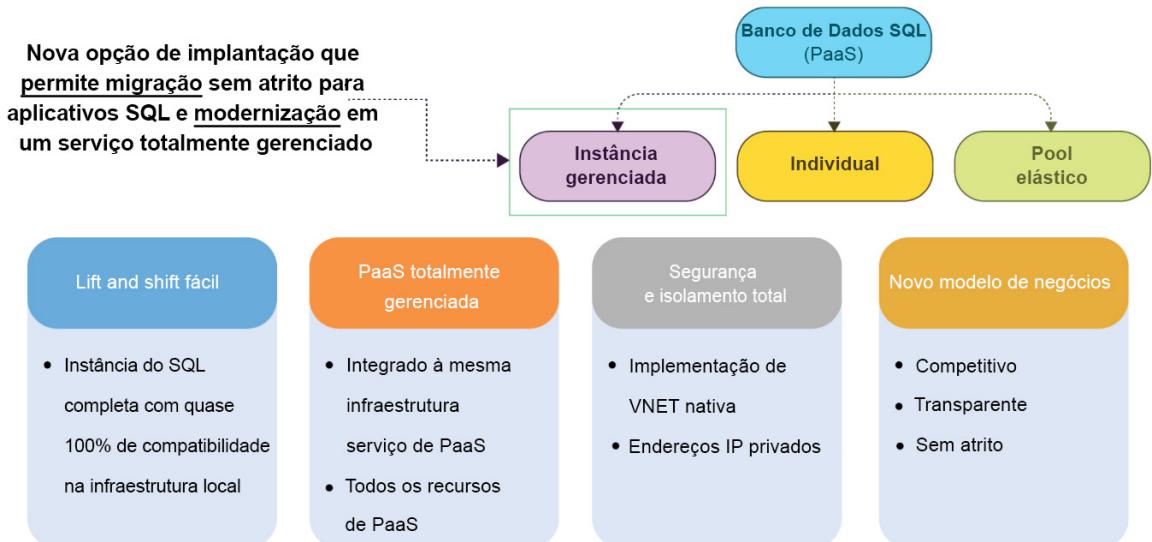


Figura 7.18: instância Gerenciada do Banco de Dados SQL do Azure

A comparação completa entre o Banco de Dados SQL do Azure, a Instância Gerenciada do SQL do Azure e o SQL Server em uma máquina virtual do Azure está disponível aqui: <https://docs.microsoft.com/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview#comparison-table>.

Os principais recursos da Instância Gerenciada são mostrados na *Figura 7.19*:

Feature	Description
SQL Server version / build	SQL Server Database Engine (latest stable)
Managed automated backups	Yes
Built-in instance and database monitoring and metrics	Yes
Automatic software patching	Yes
The latest Database Engine features	Yes
Number of data files (ROWS) per the database	Multiple
Number of log files (LOG) per database	1
VNet - Azure Resource Manager deployment	Yes
VNet - Classic deployment model	No
Portal support	Yes
Built-in Integration Service (SSIS)	No - SSIS is a part of <a href="#">Azure Data Factory PaaS</a>
Built-in Analysis Service (SSAS)	No - SSAS is separate <a href="#">PaaS</a>
Built-in Reporting Service (SSRS)	No - use Power BI or SSRS IaaS

**Figura 7.19: recursos da Instância Gerenciada do Banco de Dados SQL**

Mencionamos os termos modelo de preços baseado em vCPU e modelo de preços baseado em DTU em vários momentos ao longo do capítulo. É hora de analisarmos melhor esses modelos de preços.

## Preços do banco de dados SQL

O SQL do Azure anteriormente tinha apenas um modelo de preços, um modelo baseado em DTUs, mas um modelo de preços alternativo baseado em vCPUs também foi lançado. O modelo de preços é selecionado com base nos requisitos do cliente. O modelo baseado em DTU é selecionado quando o cliente deseja opções de recursos simples e pré-configuradas. Por outro lado, o modelo baseado em vCore oferece a flexibilidade para escolher recursos de computação e armazenamento. Ele também fornece controle e transparência.

Vamos analisar em detalhes cada um desses modelos.

### Preços baseados em DTU

A DTU é a menor unidade de medida de performance para o Banco de Dados SQL do Azure. Cada DTU corresponde a uma determinada quantidade de recursos. Esses recursos incluem armazenamento, ciclos de CPU, IOPS e largura de banda da rede. Por exemplo, uma única DTU pode fornecer três IOPS, alguns ciclos de CPU e latências de E/S de 5 ms para operações de leitura e 10 ms para operações de gravação.

O Banco de Dados SQL do Azure fornece vários SKUs para a criação de bancos de dados, e cada um desses SKUs definiu restrições para a quantidade máxima de DTUs. Por exemplo, o SKU Básico fornece apenas 5 DTUs com, no máximo, 2 GB de dados, conforme mostrado na Figura 7.20:

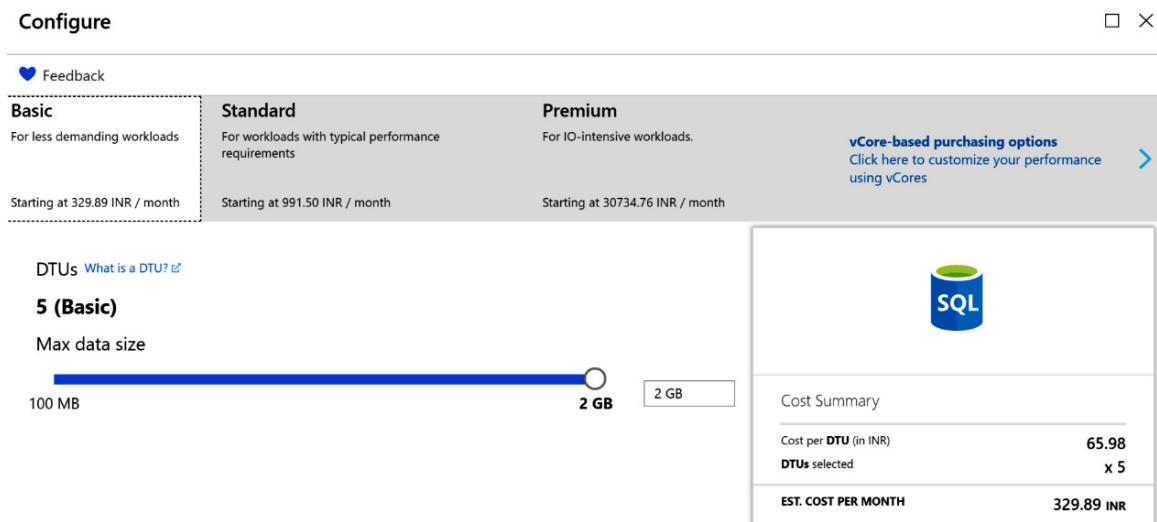


Figura 7.20: DTUs para SKUs diferentes

Por outro lado, o SKU padrão fornece de **10** a **300** DTUs com, no máximo, **250** GB de dados. Como você pode ver aqui, cada DTU custa cerca de 991 rupias ou em torno de US\$ 1,40:

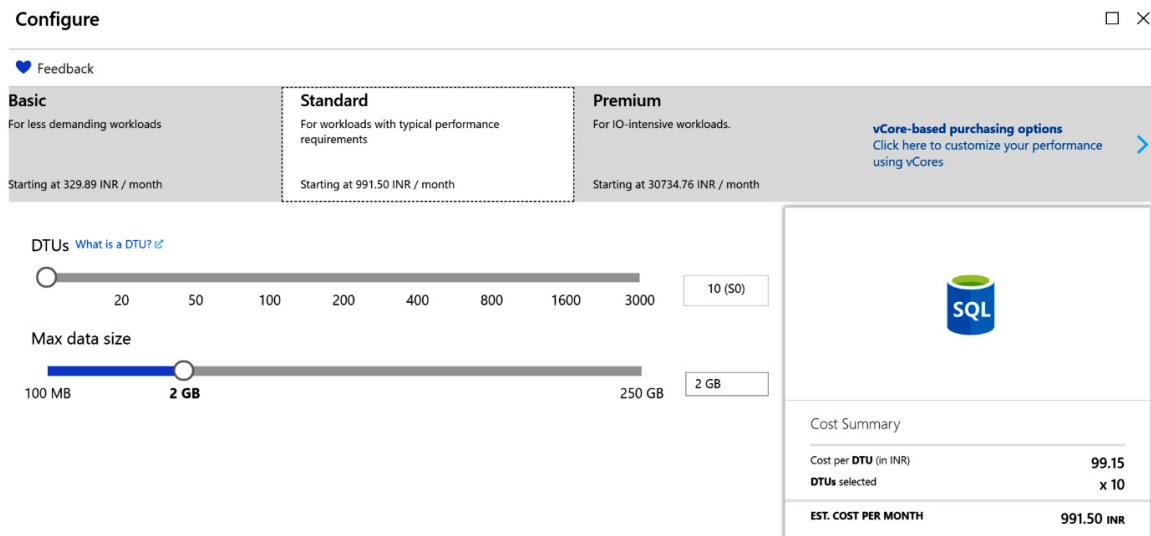


Figura 7.21: resumo de custos para o número selecionado de DTUs no SKU Padrão

Uma comparação desses SKUs em termos de performance e recursos é fornecida pela Microsoft e é mostrada na Figura 7.22:

	Basic	Standard	Premium
Target workload	Development and production	Development and production	Development and production
Uptime SLA	99.99%	99.99%	99.99%
Backup retention	7 days	35 days	35 days
CPU	Low	Low, Medium, High	Medium, High
IO throughput (approximate)	2.5 IOPS per DTU	2.5 IOPS per DTU	48 IOPS per DTU
IO latency (approximate)	5 ms (read), 10 ms (write)	5 ms (read), 10 ms (write)	2 ms (read/write)
Columnstore indexing	N/A	S3 and above	Supported
In-memory OLTP	N/A	N/A	Supported

Figura 7.22: comparação de SKUs no Azure

Depois de provisionar um determinado número de DTUs, os recursos de back-end (CPU, IOPS e memória) são alocados e cobrados, quer sejam consumidos ou não. Se forem adquiridas mais DTUs do que o necessário, isso resultará em desperdício, ao mesmo tempo, haverá gargalos de performance se DTUs insuficientes forem provisionadas.

O Azure fornece pools elásticos por esse motivo também. Como você sabe, há vários bancos de dados em um pool elástico e as DTUs são atribuídas a pools elásticos, em vez de bancos de dados individuais. É possível que todos os bancos de dados em um pool compartilhem as DTUs. Isso significa que, se um banco de dados tiver baixa utilização e estiver consumindo apenas cinco DTUs, haverá outro banco de dados consumindo 25 DTUs para compensar.

É importante observar que, coletivamente, o consumo de DTUs não pode exceder a quantidade de DTUs provisionadas para o pool elástico. Além disso, há uma quantidade mínima de DTUs que devem ser atribuídas a cada banco de dados no pool elástico, e essa contagem mínima de DTUs é pré-alocada para o banco de dados.

Um pool elástico tem seus próprios SKUs:

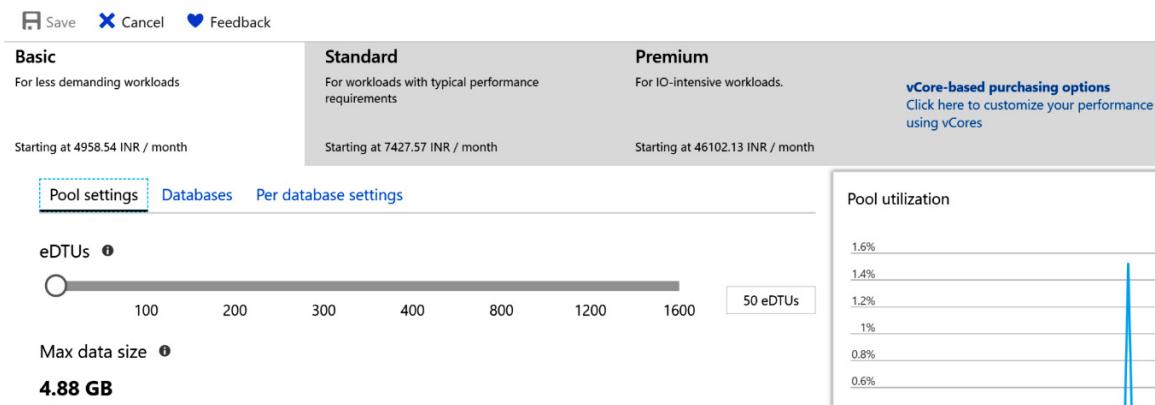


Figura 7.23: SKUs em um pool elástico

Além disso, há um limite no número máximo de bancos de dados que podem ser criados em um único pool elástico. Os limites completos podem ser revisados aqui: <https://docs.microsoft.com/azure/azure-sql/database/resource-limits-dtu-elastic-pools>.

## Preços baseados em vCPU

Este é o novo modelo de preços para o SQL do Azure. Ele fornece opções para adquirir o número de **CPUs virtuais (vCPUs)** alocadas para o servidor em vez de configurar a quantidade de DTUs necessárias para uma aplicação. Uma vCPU é uma CPU lógica com hardware conectado, como armazenamento, memória e núcleos de CPU.

Neste modelo, há três SKUs: **Uso Geral**, **Hiperescala** e **Comercialmente Crítico**, com um número variado de vCPUs e recursos disponíveis. Esses preços estão disponíveis para todos os modelos de implantação de SQL:

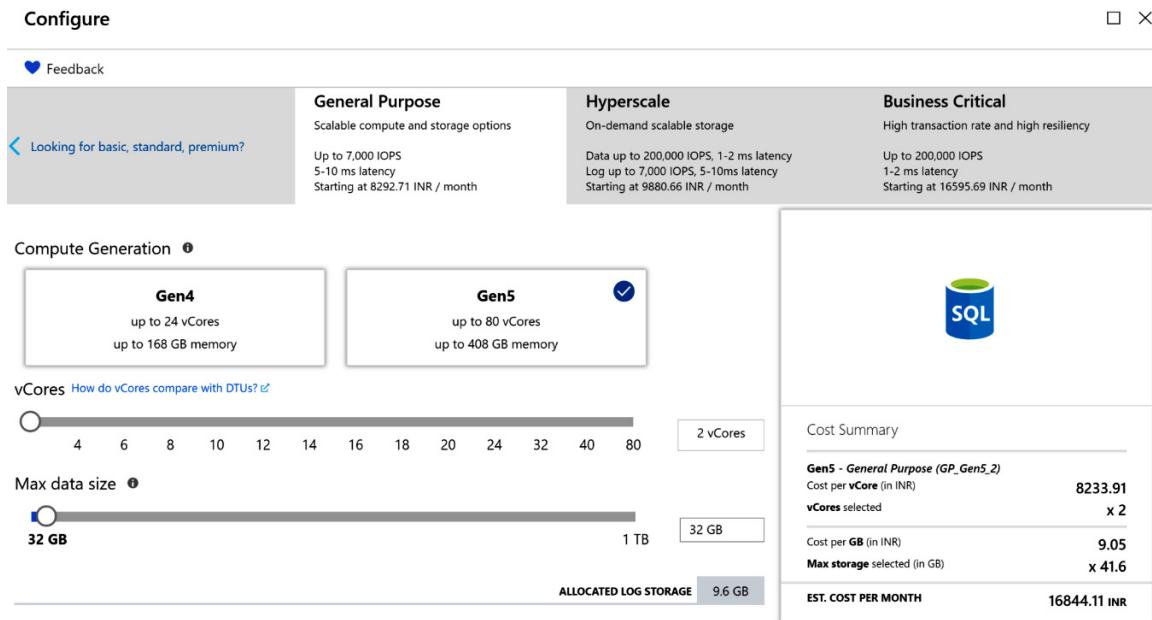


Figura 7.24: preços de vCPU para o SKU de Uso Geral

## Como escolher o modelo de preços apropriado

Os arquitetos devem ser capazes de escolher um modelo de preços apropriado para o Banco de Dados SQL do Azure. As DTUs são um ótimo mecanismo para fixação de preços em que há um padrão de uso aplicável e disponível para o banco de dados. Como a disponibilidade de recursos no esquema da DTU é linear, como mostrado no diagrama a seguir, é bem possível que o uso seja mais intensivo de memória do que de CPU. Nesses casos, é possível escolher diferentes níveis de CPU, memória e armazenamento para um banco de dados.

Nas DTUs, os recursos vêm empacotados, e não é possível configurar esses recursos em um nível granular. Com um modelo vCPU, é possível escolher diferentes níveis de memória e CPU para diferentes bancos de dados. Se o padrão de uso para uma aplicação for conhecido, usar o modelo de preços da vCPU poderá ser uma opção melhor em comparação com o modelo de DTU. Na verdade, o modelo vCPU também fornece o benefício de licenças híbridas se uma organização já tem licenças do SQL Server na infraestrutura local. Um desconto de até 30% é fornecido para essas instâncias do SQL Server.

Na Figura 7.25, você pode ver no gráfico à esquerda que, à medida que a quantidade de DTUs aumenta, a disponibilidade de recursos também aumenta linearmente; no entanto, com os preços de vCPU (no gráfico à direita), é possível escolher configurações independentes para cada banco de dados:

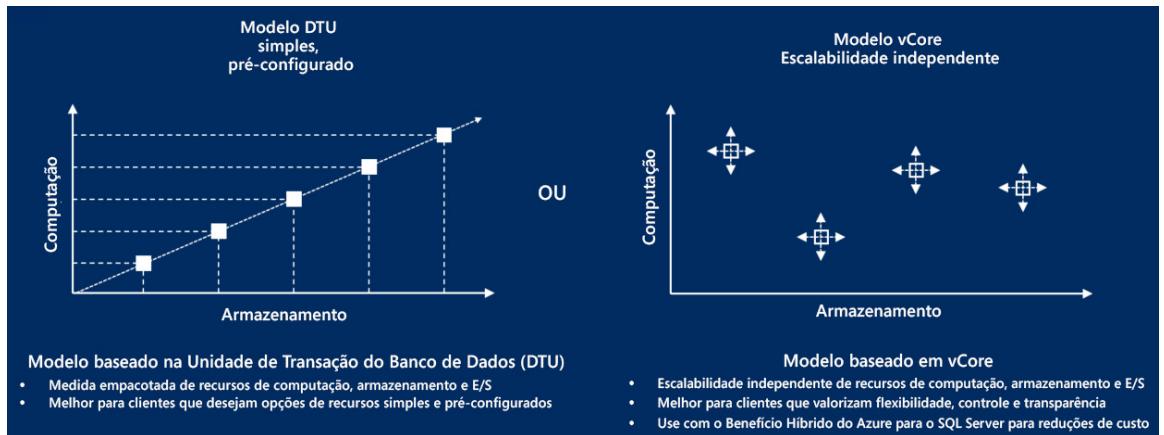


Figura 7.25: gráfico de computação/armazenamento para os modelos de DTU e vCore

Com isso, podemos concluir nossa cobertura do Banco de Dados SQL do Azure. Discutimos diferentes métodos de implantação, recursos, preços e planos relacionados ao Banco de Dados SQL do Azure. Na próxima seção, vamos abordar o Cosmos DB, que é um serviço de banco de dados NoSQL.

## Azure Cosmos DB

O Cosmos DB é o serviço de banco de dados verdadeiramente entre regiões, altamente disponível, distribuído e multimodelo do Azure. Ele é ideal para você, se quiser que sua solução seja altamente responsiva e sempre disponível. Como ele é um banco de dados multimodelo entre regiões, podemos implantar as aplicações mais próximas da localização do usuário e obter baixa latência e alta disponibilidade.

Com o clique de um botão, a taxa de transferência e o armazenamento podem ser escalados em qualquer número de regiões do Azure. Há alguns modelos de bancos de dados diferentes para cobrir quase todos os requisitos de bancos de dados não relacionais, incluindo:

1. SQL (documentos)
2. MongoDB
3. Cassandra
4. Tabela
5. Gráfico Gremlin

A hierarquia de objetos no Cosmos DB começa com a conta do Cosmos DB. Uma conta pode ter vários bancos de dados e cada banco de dados pode ter vários contêineres. Dependendo do tipo de banco de dados, o contêiner pode consistir em documentos, como no caso do SQL; dados de chave-valor semiestruturados no Armazenamento de tabelas; ou entidades e relacionamentos entre essas entidades, ao usar Gremlin e Cassandra para armazenar dados NoSQL.

O Cosmos DB pode ser usado para armazenar dados OLTP. Ele responde por ACID no que diz respeito a dados de transações, com algumas ressalvas.

O Cosmos DB sustenta os requisitos de ACID no nível do documento único. Isso significa que os dados em um documento, quando atualizados, excluídos ou inseridos, terão atomicidade, consistência, isolamento e durabilidade mantidos. No entanto, além dos documentos, a consistência e a atomicidade precisam ser gerenciadas pelo próprio desenvolvedor.

Os preços do Cosmos DB podem ser encontrados aqui: <https://azure.microsoft.com/pricing/details/cosmos-db/>.

A Figura 7.26 mostra alguns recursos do Azure Cosmos DB:

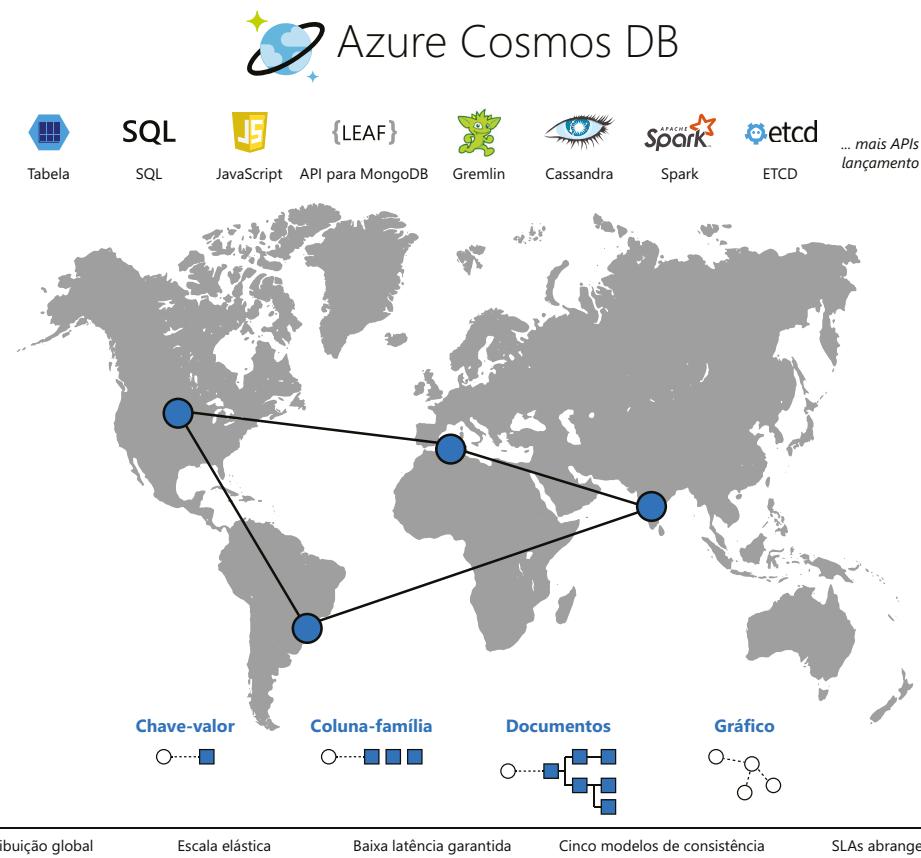


Figura 7.26: uma visão geral do Azure Cosmos DB

Na próxima seção, vamos abordar alguns dos principais recursos do Azure Cosmos DB.

## Recursos

Alguns dos principais benefícios do Azure Cosmos DB são:

- **Distribuição global:** aplicações altamente responsivas e altamente disponíveis podem ser criadas no mundo todo usando o Azure Cosmos DB. Com a ajuda da replicação, as réplicas de dados podem ser armazenadas em regiões do Azure que estejam próximas dos usuários, proporcionando assim menos latência e distribuição global.
- **Replicação:** você pode aceitar ou recusar a replicação para uma região a qualquer momento que desejar. Digamos que você tenha uma réplica dos seus dados disponível na região Leste dos EUA, e sua organização esteja planejando encerrar seus processos no Leste dos EUA e migrar para o Sul do Reino Unido. Com apenas alguns cliques, a região Leste dos EUA pode ser removida e a região Sul do Reino Unido pode ser adicionada à conta para replicação.
- **AlwaysOn:** o Cosmos DB fornece 99,999% de alta disponibilidade para leitura e gravação. O failover regional de uma conta do Cosmos DB para outra região pode ser invocado por meio do portal do Azure ou programaticamente. Isso garante a continuidade dos negócios e o planejamento de disaster recovery para a sua aplicação durante uma falha regional.
- **Escalabilidade:** o Cosmos DB oferece escalabilidade elástica incomparável para gravações e leituras em todo o mundo. A resposta de escalabilidade é enorme, o que significa que você pode escalar de milhares para centenas de milhões de solicitações/segundo com uma única chamada à API. O interessante é que isso é feito em todo o mundo, mas você precisa pagar apenas pela taxa de transferência e pelo armazenamento. Esse nível de escalabilidade é ideal para lidar com picos inesperados.
- **Baixa latência:** como mencionado anteriormente, replicar cópias de dados para locais mais próximos dos usuários reduz drasticamente a latência; isso significa que os usuários podem acessar seus dados em milissegundos. O Cosmos DB garante menos de 10 ms de latência para leituras e gravações em todo o mundo.
- **Economias de TCO:** como o Cosmos DB é um serviço totalmente gerenciado, o nível de gerenciamento exigido do cliente é baixo. Além disso, o cliente não precisa configurar datacenters em todo o mundo para acomodar usuários de outras regiões.
- **SLA:** ele oferece um SLA de 99,999% de alta disponibilidade.
- **Suporte para APIs de Software Open Source (OSS):** o suporte para APIs de OSS é outra vantagem adicional do Cosmos DB. Ele implementa APIs para Cassandra, Mongo DB, Gremlin e Armazenamento de tabelas do Azure.

## Cenários de uso

Se a sua aplicação envolve altos níveis de leituras e gravações de dados em uma escala global, o Cosmos DB é a escolha ideal. Os tipos comuns de aplicações que têm esses requisitos incluem aplicações Web, móveis, de jogos e de Internet das Coisas. Essas aplicações se beneficiariam da alta disponibilidade, baixa latência e presença global do Cosmos DB.

Além disso, o tempo de resposta oferecido pelo Cosmos DB é quase em tempo real. Os SDKs do Cosmos DB podem ser aproveitados para desenvolver aplicações para iOS e Android usando a estrutura Xamarin.

Alguns dos jogos populares que usam o Cosmos DB são **The Walking Dead: No Man's Land** da Next Games e **Halo 5: Guardians**.

Uma lista completa de cenários de uso e exemplos pode ser encontrada aqui: <https://docs.microsoft.com/azure/cosmos-db/use-cases>.

O Cosmos DB é o serviço popular no Azure para armazenar dados semiestruturados como parte de aplicações OLTP. Eu poderia escrever um livro inteiro sobre os recursos e as funcionalidades do Cosmos DB. A intenção desta seção foi fornecer uma introdução ao Cosmos DB e à função que ele desempenha no tratamento de aplicações OLTP.

## Resumo

Neste capítulo, você aprendeu que o Banco de Dados SQL do Azure é um dos principais serviços do Azure. Uma infinidade de clientes estão usando esse serviço hoje e ele oferece todos os recursos empresariais necessários para um sistema de gerenciamento de bancos de dados de missão crítica.

Você descobriu que há vários tipos de implantação para o Banco de Dados SQL do Azure, como Instância Única, Instância Gerenciada e pools elásticos. Os arquitetos devem realizar uma avaliação completa de seus requisitos e escolher o modelo de implantação apropriado. Depois de escolher um modelo de implantação, os arquitetos devem escolher uma estratégia de preços entre DTUs e vCPUs. Eles também devem configurar todos os requisitos de segurança, disponibilidade, disaster recovery, monitoramento, performance e escalabilidade no Banco de Dados SQL do Azure em relação aos dados.

No próximo capítulo, discutiremos como criar aplicações seguras no Azure. Abordaremos as práticas e os recursos de segurança da maioria dos serviços.





# 8

## Arquitetar aplicações seguras no Azure

No capítulo anterior, discutimos os serviços de dados do Azure. Como estamos lidando com dados confidenciais, a segurança é uma grande preocupação. A segurança é, sem dúvida, o requisito não funcional mais importante para que os arquitetos implementem. As empresas enfatizam muito a implementação correta de sua estratégia de segurança. Na verdade, a segurança é uma das principais preocupações de quase todos os stakeholders no desenvolvimento, na implantação e no gerenciamento de uma aplicação. Ela se torna ainda mais importante quando uma aplicação é criada para implantação na nuvem.

Para que você entenda como pode proteger suas aplicações no Azure dependendo da natureza da implantação, os seguintes tópicos serão abordados neste capítulo:

- Entender a segurança no Azure
- Segurança no nível da infraestrutura
- Segurança no nível da aplicação
- Autenticação e autorização em aplicações do Azure
- Trabalhar com OAuth, Azure Active Directory e outros métodos de autenticação usando identidade federada, incluindo provedores de identidade de terceiros, como o Facebook
- Entender identidades gerenciadas e usá-las para acessar recursos

## Segurança

Como mencionado anteriormente, a segurança é um elemento importante para qualquer software ou serviço. A segurança adequada deve ser implementada para que uma aplicação só possa ser usada por pessoas que tenham permissão para acessá-la, e os usuários não devem ser capazes de executar operações para as quais não possuem autorização. Da mesma forma, todo o mecanismo de solicitação-resposta deve ser criado usando métodos que garantam que apenas as partes pretendidas possam entender as mensagens e deve garantir que seja fácil detectar se as mensagens foram adulteradas ou não.

Pelos motivos a seguir, a segurança no Azure é ainda mais importante. Em primeiro lugar, as organizações que implantam suas aplicações não estão no controle total do hardware e das redes subjacentes. Em segundo lugar, a segurança precisa ser incorporada a todas as camadas, incluindo hardware, redes, sistemas operacionais, plataformas e aplicações. Quaisquer omissões ou configurações incorretas podem tornar uma aplicação vulnerável a intrusos. Por exemplo, talvez você tenha ouvido falar da recente vulnerabilidade que afetou as reuniões no Zoom e permitiu que os hackers gravassem reuniões mesmo quando o host da reunião havia desabilitado a gravação para os participantes. Fontes alegam que milhões de contas do Zoom foram vendidas na dark web. A empresa tomou as medidas necessárias para resolver essa vulnerabilidade.

A segurança é uma grande preocupação nos dias de hoje, especialmente ao hospedar aplicações na nuvem, e pode levar a consequências catastróficas se não for tratada de forma adequada. Portanto, é necessário entender as práticas recomendadas envolvidas na proteção dos seus workloads. Estamos progredindo na área de DevOps, onde as equipes de desenvolvimento e operações colaboram de forma eficaz com a ajuda de ferramentas e práticas, e a segurança também tem sido uma grande preocupação.

Para acomodar princípios e práticas de segurança como uma parte vital do DevOps sem afetar a produtividade e a eficiência geral do processo, uma nova cultura conhecida como **DevSecOps** foi introduzida. O DevSecOps nos ajuda a identificar problemas de segurança no início do estágio de desenvolvimento, em vez de mitigá-los após o envio. Em um processo de desenvolvimento que tem a segurança como um princípio fundamental de todos os estágios, o DevSecOps reduz o custo de contratação de profissionais de segurança em um estágio posterior para encontrar falhas de segurança com o software.

Proteger uma aplicação significa que entidades desconhecidas e não autorizadas não podem acessá-la. Também significa que a comunicação com a aplicação é segura e não adulterada. Isso inclui as seguintes medidas de segurança:

- **Autenticação:** a autenticação verifica a identidade de um usuário e garante que a identidade determinada possa acessar a aplicação ou o serviço. Ela é executada no Azure usando o OpenID Connect, que é um protocolo de autenticação criado no OAuth 2.0.
- **Autorização:** a autorização permite e estabelece permissões que uma identidade pode realizar na aplicação ou no serviço. A autorização é realizada no Azure usando OAuth.
- **Confidencialidade:** a confidencialidade garante que a comunicação entre o usuário e a aplicação permaneça segura. A troca de carga entre entidades é criptografada de modo que faça sentido apenas para o remetente e o destinatário, mas não para outras pessoas. A confidencialidade de mensagens é garantida usando criptografia simétrica e assimétrica. Os certificados são usados para implementar a criptografia, isto é, a criptografia e descriptografia de mensagens.

A criptografia simétrica usa uma chave única, que é compartilhada com o remetente e o destinatário, enquanto a criptografia assimétrica usa um par de chaves privada e pública para criptografia, o que é mais seguro. Os pares de chaves SSH no Linux, que são usados para autenticação, são um ótimo exemplo de criptografia assimétrica.

- **Integridade:** a integridade garante que a troca de cargas e mensagens entre o remetente e o destinatário não seja adulterada. O destinatário recebe a mesma mensagem enviada pelo remetente. Os hashes e as assinaturas digitais são os mecanismos de implementação para verificar a integridade de mensagens recebidas.

A segurança é uma parceria entre o provedor e o consumidor do serviço. Ambas as partes têm diferentes níveis de controle em pilhas de implantação e cada uma deve implementar as práticas recomendadas de segurança para garantir que todas as ameaças sejam identificadas e mitigadas. Já vimos no Capítulo 1, *Introdução ao Azure*, que a nuvem fornece amplamente três paradigmas, IaaS, PaaS e SaaS, e cada um deles tem diferentes níveis de controle colaborativo sobre a pilha de implantação. Cada parte deve implementar práticas de segurança para os componentes sob seu controle e dentro do seu escopo. A falha na implementação da segurança em qualquer camada da pilha ou por qualquer parte tornaria toda a implantação e a aplicação vulneráveis a ataques. Toda organização precisa ter um modelo de ciclo de vida para a segurança, assim como para qualquer outro processo. Isso garante que as práticas de segurança sejam continuamente aprimoradas para evitar falhas de segurança. Na próxima seção, discutiremos o ciclo de vida da segurança e como ele pode ser usado.

## Ciclo de vida da segurança

A segurança é frequentemente considerada um requisito não funcional para uma solução. No entanto, com o número crescente de ataques cibernéticos no momento, hoje, ela é considerada um requisito funcional de todas as soluções.

Toda organização segue algum tipo de gerenciamento de ciclo de vida da aplicação para suas aplicações. Quando a segurança é tratada como um requisito funcional, ela deve seguir o mesmo processo de desenvolvimento de aplicações. A segurança não deve ser deixada para um segundo momento. Ela deve fazer parte da aplicação desde o início. No âmbito da fase de planejamento geral de uma aplicação, a segurança também deve ser planejada. Dependendo da natureza da aplicação, diferentes tipos e categorias de ameaças devem ser identificados e, com base nessas identificações, eles devem ser documentados em termos de escopo e abordagem para mitigá-los. Um exercício de modelagem de ameaças deve ser realizado para ilustrar a ameaça a que cada componente pode estar sujeito. Isso levará ao design de padrões e políticas de segurança para a aplicação. Isso normalmente é a fase de design de segurança. A próxima fase é chamada de fase de criação ou mitigação de ameaças. Nela, a implementação de segurança em termos de código e configuração é executada para mitigar as ameaças e os riscos à segurança.

Um sistema não pode estar seguro até que seja testado. Os testes de penetração adequados e outros testes de segurança devem ser realizados para identificar possíveis mitigações de ameaças que não foram implementadas ou foram ignoradas. Os bugs de teste são corrigidos, e o ciclo continua até o final da vida útil da aplicação. Esse processo de gerenciamento de ciclo de vida da aplicação, mostrado na Figura 8.1, deve ser seguido para a segurança:

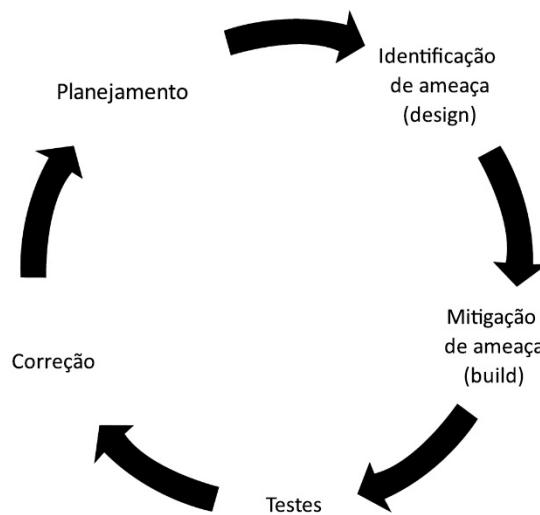


Figura 8.1: ciclo de vida da segurança

Planejamento, modelagem de ameaças, identificação, mitigação, testes e correção são processos iterativos que continuam mesmo quando uma aplicação ou um serviço está em funcionamento. Deve haver monitoramento ativo de ambientes e aplicações inteiros para atenuá-los e identificar ameaças proativamente. O monitoramento também deve habilitar alertas e logs de auditoria para ajudar no diagnóstico reativo, na solução de problemas e na eliminação de ameaças e vulnerabilidades.

O ciclo de vida de segurança para qualquer aplicação começa com a fase de planejamento, que leva à fase de design. Na fase de design, a arquitetura da aplicação é decomposta em componentes granulares com comunicação discreta e limites de hospedagem. As ameaças são identificadas com base na sua interação com outros componentes dentro e entre os limites de hospedagem. Na arquitetura geral, as ameaças são mitigadas implementando os recursos de segurança adequados e, quando a mitigação estiver em vigor, mais testes serão realizados para verificar se a ameaça ainda existe. Depois que a aplicação é implantada na produção e se torna operacional, ela é monitorada para verificar se possui quaisquer vulnerabilidades e violações de segurança, e uma correção proativa ou reativa é realizada.

Como mencionado anteriormente, diferentes organizações têm diferentes processos e métodos para implementar o ciclo de vida da segurança; da mesma forma, a Microsoft fornece orientações completas e informações sobre o ciclo de vida da segurança, que estão disponíveis em <https://www.microsoft.com/securityengineering/sdl/practices>. Usando as práticas que a Microsoft compartilhou, todas as organizações podem se concentrar na criação de soluções mais seguras. À medida que estamos progredindo na era da computação na nuvem e migrando nossos dados corporativos e de clientes para a nuvem, aprender a proteger esses dados é vital. Na próxima seção, exploraremos a segurança do Azure e os diferentes níveis de segurança, o que nos ajudará a criar soluções seguras no Azure.

## Segurança do Azure

O Azure fornece todos os seus serviços por meio de datacenters em várias regiões. Esses datacenters estão interconectados nas regiões, bem como entre elas. O Azure entende que ele hospeda dados, serviços e aplicações importantes e de missão crítica para seus clientes. Ele deve garantir que a segurança seja de extrema importância para seus datacenters e regiões.

Os clientes implantam aplicações na nuvem com base na crença de que o Azure protegerá suas aplicações e dados contra vulnerabilidades e violações. Eles não migrarão para a nuvem se essa confiança for quebrada, portanto, o Azure implementa a segurança em todas as camadas, conforme visto na Figura 8.2, desde o perímetro físico dos datacenters até os componentes de software lógicos. Cada camada é protegida e até mesmo a equipe do datacenter do Azure não tem acesso a elas:

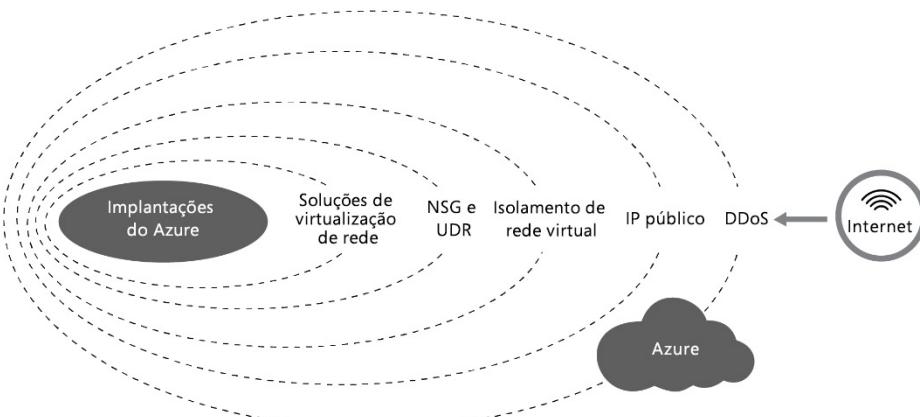


Figura 8.2: recursos de segurança em diferentes camadas nos datacenters do Azure

A segurança é de suma importância para a Microsoft e para o Azure. A Microsoft garante que a confiança seja construída com seus clientes e faz isso garantindo que as implantações, as soluções e os dados desses clientes estejam completamente seguros, física e virtualmente. As pessoas não usarão uma plataforma de nuvem se ela não for segura de forma física e digital.

Para garantir que os clientes tenham confiança no Azure, cada atividade em desenvolvimento do Azure é planejada, documentada, auditada e monitorada de uma perspectiva de segurança. Os datacenters físicos do Azure são protegidos contra intrusão e acesso não autorizado. Na verdade, nem a equipe de operações e os funcionários da Microsoft têm acesso às soluções e aos dados dos clientes. Alguns dos recursos de segurança prontos para uso fornecidos pelo Azure estão listados aqui:

- **Acesso seguro do usuário:** a implantação, a solução e os dados de um cliente só podem ser acessados pelo cliente. Até mesmo os funcionários do datacenter do Azure não têm acesso aos artefatos do cliente. Os clientes podem permitir o acesso de outras pessoas; no entanto, isso fica a critério deles.

- **Criptografia em repouso:** o Azure criptografa todos os seus dados de gerenciamento, o que inclui uma variedade de soluções de armazenamento de nível empresarial para acomodar diferentes necessidades. A Microsoft também fornece criptografia a serviços gerenciados, como Banco de Dados SQL do Azure, Azure Cosmos DB e Azure Data Lake Storage. Como os dados são criptografados em repouso, eles não podem ser lidos por ninguém. Ele também fornece essa funcionalidade aos seus clientes, bem como aqueles que podem criptografar os dados em repouso.
- **Criptografia em trânsito:** o Azure criptografa todos os dados que fluem de sua rede. Ele também garante que seu backbone de rede esteja protegido contra acessos não autorizados.
- **Monitoramento e auditoria ativos:** o Azure monitora todos os seus datacenters de forma ativa e contínua. Ele identifica ativamente qualquer violação, ameaça ou risco e os mitiga.

O Azure atende aos padrões de conformidade locais, internacionais e específicos da indústria e do país. Você pode explorar a lista completa de ofertas de conformidade da Microsoft em <https://www.microsoft.com/trustcenter/compliance/complianceofferings>. Mantenha isso como uma referência ao implantar soluções em conformidade no Azure. Agora que conhecemos os principais recursos de segurança no Azure, vamos nos aprofundar na segurança de IaaS. Na próxima seção, exploraremos como os clientes podem aproveitar os recursos de segurança disponíveis para IaaS no Azure.

## Segurança de IaaS

O Azure é uma plataforma madura para a implantação de soluções de IaaS. Há muitos usuários do Azure que desejam o controle total sobre as implantações, e normalmente eles usam a IaaS para suas soluções. É importante que essas implantações e soluções sejam seguras por padrão e design. O Azure fornece recursos de segurança avançados para proteger soluções de IaaS. Nesta seção, alguns dos principais recursos serão abordados.

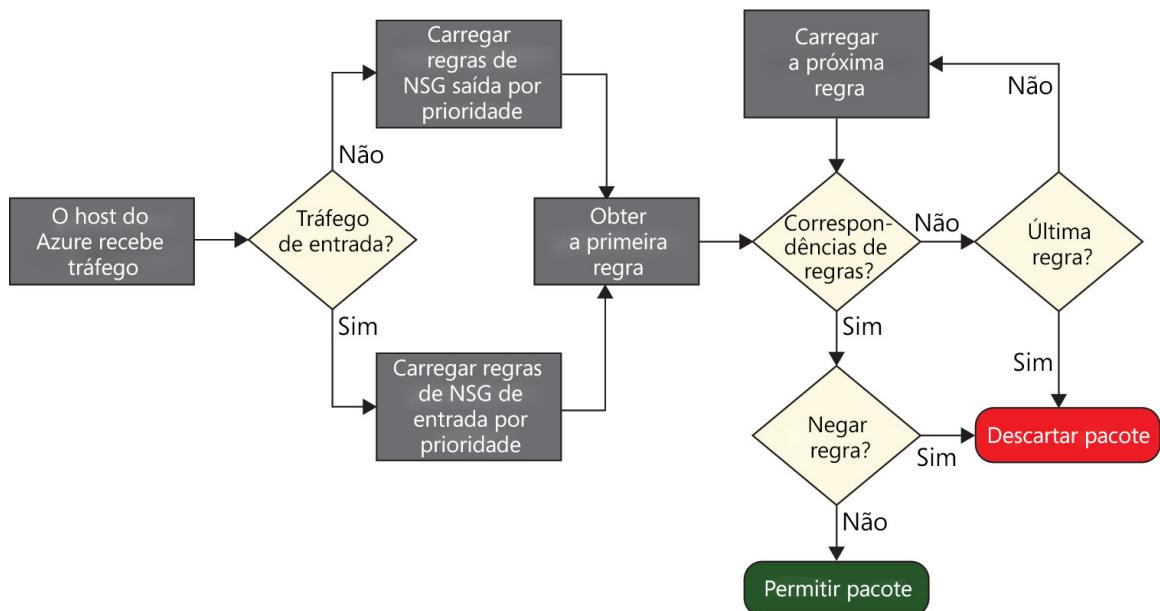
### Grupos de segurança de rede

O mínimo necessário para a implantação de IaaS consiste em máquinas e redes virtuais. Uma máquina virtual pode ser exposta à internet aplicando um IP público à sua interface de rede ou pode estar disponível somente para os recursos internos. Alguns desses recursos internos, por sua vez, podem ser expostos à Internet. De qualquer maneira, as máquinas virtuais devem ser protegidas de modo que as solicitações não autorizadas nem mesmo as alcancem. As máquinas virtuais devem ser protegidas usando instalações que possam filtrar solicitações na própria rede, em vez de as solicitações chegarem a uma máquina virtual e essa máquina precisar tomar medidas em relação a elas.

Delimitação é um mecanismo que as máquinas virtuais usam como um de seus mecanismos de segurança. Essa delimitação pode permitir ou negar solicitações com base em seu protocolo, IP de origem, IP de destino, porta de origem e porta de destino. Esse recurso é implantado usando os **grupos de segurança de rede (NSGs)** do Azure. Os NSGs são compostos por regras que são avaliadas para solicitações de entrada e de saída. Dependendo da execução e da avaliação dessas regras, será determinado se as solicitações deverão ter o acesso permitido ou negado.

Os NSGs são flexíveis e podem ser aplicados a uma sub-rede da rede virtual ou interfaces de rede individuais. Quando aplicados a uma sub-rede, as regras de segurança são aplicadas a todas as máquinas virtuais hospedadas nessa sub-rede. Por outro lado, a aplicação a uma interface de rede afeta as solicitações apenas para uma determinada máquina virtual associada a essa interface. Também é possível aplicar NSGs a interfaces de rede e de sub-rede da rede simultaneamente. Normalmente, esse design deve ser usado para a aplicação de regras de segurança comuns no nível da sub-rede e regras de segurança exclusivas no nível da interface de rede. Ele ajuda no design de regras de segurança modulares.

O fluxo para avaliar NSGs é mostrado na *Figura 8.3*:



**Figura 8.3:** um diagrama de fluxo que representa a avaliação de NSGs

Quando uma solicitação atinge um host do Azure, se é uma solicitação de entrada ou saída, as regras apropriadas são carregadas e executadas em relação à solicitação/resposta. Se a regra corresponder à solicitação/resposta, a solicitação/resposta será permitida ou negada. A correspondência de regras consiste em informações importantes de solicitação/resposta, como o endereço IP de origem, o endereço IP de destino, a porta de origem, a porta de destino e o protocolo usado. Além disso, os NSGs oferecem suporte às marcas de serviço. Uma marca de serviço denota um grupo de prefixos de endereços IP de um determinado serviço do Azure. A Microsoft gerencia os prefixos de endereços e atualiza-os automaticamente. Isso elimina o transtorno de atualizar as regras de segurança toda vez que houver uma alteração em prefixos de endereços.

O conjunto de marcas de serviço disponíveis para uso está disponível em <https://docs.microsoft.com/azure/virtual-network/service-tags-overview#available-service-tags>. As marcas de serviço podem ser usadas com NSGs e com o Firewall do Azure. Agora que você aprendeu sobre como os NSGs funcionam, vamos analisar o design de NSGs, o que ajudará você a determinar os principais pontos a serem considerados na criação de regras de NSG para melhorar a segurança.

## Design de NSGs

A primeira etapa no design de um NSG é determinar os requisitos de segurança do recurso. O seguinte deve ser determinado ou considerado:

- O recurso só pode ser acessado por meio da Internet?
- O recurso pode ser acessado por meio de recursos internos e da Internet?
- O recurso só pode ser acessado por meio de recursos internos?
- Com base na arquitetura da solução que está sendo implantada, determine os recursos dependentes,平衡adores de carga, gateways e máquinas virtuais usados.
- Configure uma rede virtual e sua sub-rede.

Com os resultados dessas investigações, um design NSG adequado deve ser criado. Idealmente, deve haver várias sub-redes da rede para cada tipo de recurso e workload. Não é recomendável implantar平衡adores de carga e máquinas virtuais na mesma sub-rede.

Considerando os requisitos do projeto, devem ser determinadas as regras que são comuns para sub-redes e workloads de máquina virtual diferentes. Por exemplo, para uma implantação do SharePoint, a aplicação de front-end e os SQL Servers são implantados em sub-redes separadas. Portanto, devem ser determinadas regras para cada sub-rede.

Após a identificação das regras comuns no nível da sub-rede, as regras para recursos individuais devem ser identificadas e aplicadas no nível da interface de rede. É importante entender que, se uma regra permitir uma solicitação de entrada em uma porta, essa porta também poderá ser usada para solicitações de saída sem nenhuma configuração.

Se os recursos forem acessíveis pela Internet, as regras deverão ser criadas com portas e intervalos de IP específicos, sempre que possível, em vez de permitir o tráfego de todos os intervalos de IP (geralmente representados como 0.0.0.0/0). Testes funcionais e de segurança cuidadosos devem ser executados para garantir que regras de NSG adequadas e ideais sejam abertas e fechadas.

## Firewalls

Os NSGs fornecem perímetros de segurança externa para solicitações. No entanto, isso não significa que as máquinas virtuais não devem implementar medidas de segurança adicionais. É sempre melhor implementar a segurança interna e externamente. As máquinas virtuais no Linux ou no Windows fornecem um mecanismo para filtrar solicitações no nível do sistema operacional. Isso é conhecido como firewall no Windows e no Linux.

É aconselhável implementar firewalls para sistemas operacionais. Eles ajudam na criação de um muro de segurança virtual que permite somente as solicitações consideradas confiáveis. Quaisquer solicitações não confiáveis terão o acesso negado. Há até mesmo dispositivos de firewall físicos, mas, na nuvem, os firewalls do sistema operacional são usados. A Figura 8.4 mostra a configuração do firewall para um sistema operacional Windows:

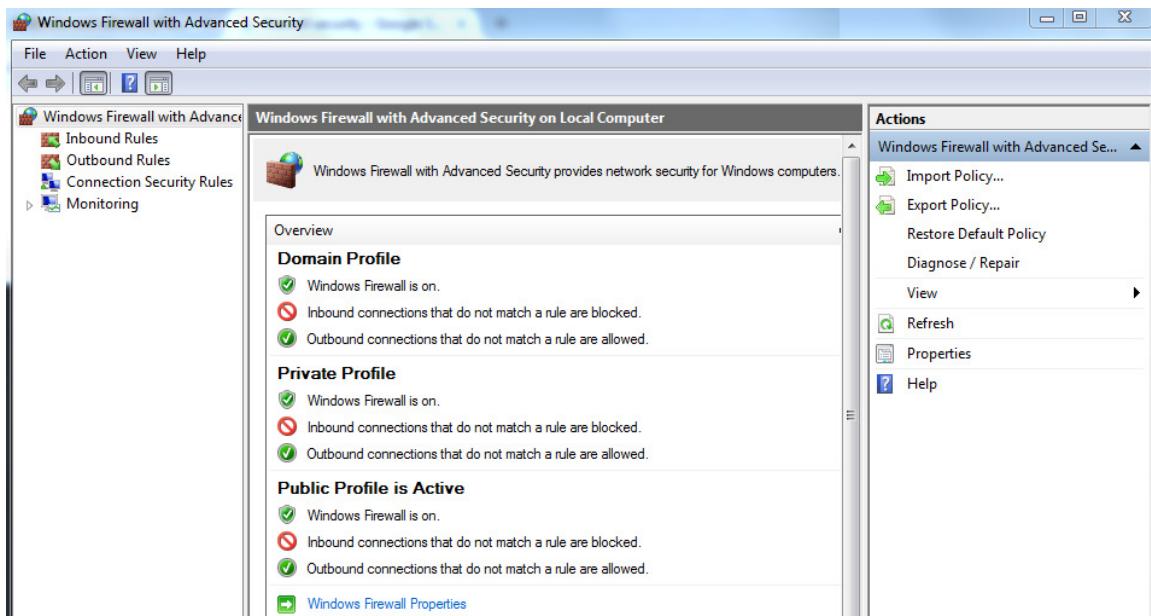


Figura 8.4: configuração do firewall

Os firewalls filtram pacotes de rede e identificam portas de entrada e endereços IP. Com as informações desses pacotes, o firewall avalia as regras e decide se deve permitir ou negar o acesso.

Quando se trata do Linux, há diferentes soluções de firewall disponíveis. Algumas das ofertas de firewall são muito específicas para a distribuição que está sendo usada; por exemplo, o SUSE usa SuSefirewall2 e o Ubuntu usa ufw. As implementações mais usadas são firewalld e iptables, que estão disponíveis em todas as distribuições.

## Design de firewall

Como uma prática recomendada, os firewalls devem ser avaliados para sistemas operacionais individuais. Cada máquina virtual tem uma responsabilidade distinta na implantação e na solução gerais. As regras para essas responsabilidades individuais devem ser identificadas e os firewalls devem ser abertos e fechados adequadamente.

Ao avaliar as regras de firewall, é importante considerar as regras de NSG no nível da interface de rede individual e da sub-rede. Se isso não for feito corretamente, é possível que as regras sejam negadas no nível do NSG, mas deixadas em aberto no nível do firewall e vice-versa. Se uma solicitação for permitida no nível do NSG e negada no nível do firewall, a aplicação não funcionará conforme o esperado, enquanto os riscos à segurança aumentarão se uma solicitação for negada no nível do NSG e permitida no nível do firewall.

Um firewall ajuda você a criar várias redes isoladas por suas regras de segurança. Testes funcionais e de segurança cuidadosos devem ser executados para garantir que regras de firewall adequadas e ideais sejam abertas e fechadas.

Faz mais sentido usar o Firewall do Azure, que é um serviço de rede baseado na nuvem sobre os NSGs. Ele é muito fácil de configurar, fornece gerenciamento central para administração e não requer manutenção. O Firewall do Azure e os NSGs combinados podem fornecer segurança entre máquinas virtuais, redes virtuais e até mesmo diferentes assinaturas do Azure. Dito isso, se uma solução exigir esse nível extra de segurança, poderemos considerar a implementação de um firewall no nível do sistema operacional. Vamos discutir o Firewall do Azure em mais detalhes em uma das próximas seções, *Firewall do Azure*.

## Grupos de segurança de aplicações

Os NSGs são aplicados no nível da sub-rede da rede virtual ou diretamente nas interfaces de rede individuais. Embora seja suficiente aplicar NSGs no nível da sub-rede, há momentos em que isso não basta. Há diferentes tipos de workloads disponíveis em uma única sub-rede e cada um deles requer um grupo de segurança diferente. É possível atribuir grupos de segurança a **placas de interface de rede (NICs)** individuais das máquinas virtuais, mas isso pode facilmente se tornar um pesadelo de manutenção se houver um grande número de máquinas virtuais.

O Azure tem um recurso relativamente novo conhecido como grupos de segurança de aplicações. Podemos criar grupos de segurança de aplicações e atribuí-los diretamente a várias NICs, mesmo quando essas NICs pertencerem a máquinas virtuais em diferentes sub-redes e grupos de recursos. A funcionalidade dos grupos de segurança de aplicações é semelhante aos NSGs, exceto que eles fornecem uma maneira alternativa de atribuir grupos a recursos de rede, oferecendo flexibilidade adicional na atribuição deles entre grupos de recursos e sub-redes. Os grupos de segurança de aplicações podem simplificar os NSGs; no entanto, há uma limitação principal. Podemos ter um grupo de segurança de aplicações na origem e no destino de uma regra de segurança, mas, no momento, não há suporte para vários grupos de segurança de aplicações em uma origem ou destino.

Uma das práticas recomendadas para a criação de regras é sempre minimizar o número de regras de segurança necessárias, para evitar a manutenção de regras explícitas. Na seção anterior, discutimos o uso de marcas de serviço com NSGs para eliminar o transtorno de manter os prefixos de endereços IP individuais de cada serviço. Da mesma forma, ao usar grupos de segurança de aplicações, podemos reduzir a complexidade de endereços IP explícitos e várias regras. Essa prática é recomendada sempre que possível. Se a sua solução exigir uma regra explícita com um endereço IP individual ou um intervalo de endereços IP, somente então você deverá optar por ela.

## Firewall do Azure

Na seção anterior, discutimos o uso do Firewall do Azure em um sistema operacional Windows/Linux para permitir ou negar solicitações e respostas por meio de portas e serviços específicos. Embora os firewalls do sistema operacional desempenhem um papel importante do ponto de vista da segurança e devam ser implementados para qualquer implantação empresarial, o Azure fornece um recurso de segurança conhecido como Firewall do Azure que tem uma funcionalidade semelhante de filtrar solicitações com base em regras e determinar se uma solicitação deve ser permitida ou rejeitada.

A vantagem de usar o Firewall do Azure é que ele avalia uma solicitação antes que ela alcance um sistema operacional. O Firewall do Azure é um recurso de rede e é um serviço autônomo que protege recursos no nível da rede virtual. Quaisquer recursos, incluindo máquinas virtuais e平衡adores de carga, que estejam diretamente associados a uma rede virtual podem ser protegidos usando o Firewall do Azure.

O Firewall do Azure é um serviço altamente disponível e escalonável capaz de proteger não apenas solicitações baseadas em HTTP, mas qualquer tipo de solicitação que entre ou saia de uma rede virtual, incluindo FTP, SSH e RDP. Ele também pode abranger várias Zonas de Disponibilidade durante a implantação para fornecer maior disponibilidade.

É altamente recomendável que o Firewall do Azure seja implantado para workloads de missão crítica no Azure, juntamente com outras medidas de segurança. Também é importante observar que o Firewall do Azure deve ser usado mesmo se outros serviços, como Gateway de Aplicativo do Azure e Azure Front Door, forem usados, pois todas essas ferramentas têm escopos e recursos diferentes. Além disso, o Firewall do Azure fornece suporte para marcas de serviço e inteligência contra ameaças. Na seção anterior, discutimos as vantagens de usar marcas de serviço. A inteligência contra ameaças pode ser usada para gerar alertas quando o tráfego vier ou for para domínios e endereços IP mal-intencionados conhecidos, que são registrados no feed da Inteligência contra Ameaças da Microsoft.

## Reducir a área de superfície de ataque

Os NSGs e os firewalls ajudam no gerenciamento de solicitações autorizadas para o ambiente. No entanto, o ambiente não deve ser excessivamente exposto a ataques. A área de superfície do sistema deve ser idealmente habilitada para atingir sua funcionalidade, mas desabilitada o suficiente para que os invasores não possam encontrar brechas e acessar áreas que estejam abertas sem qualquer uso pretendido ou abertas, mas não protegidas adequadamente. A segurança deve ser adequadamente fortalecida, tornando difícil para qualquer invasor entrar no sistema.

Algumas das configurações que devem ser feitas incluem:

- Remover todos os grupos e usuários desnecessários do sistema operacional.
- Identificar a associação de grupo para todos os usuários.
- Implementar políticas de grupo usando os serviços de diretório.
- Bloquear a execução de scripts, a menos que ele seja assinado por autoridades confiáveis.
- Registrar em log e auditar todas as atividades.
- Instalar software antivírus e malware, agendar varreduras e atualizar as definições frequentemente.
- Desabilitar ou desligar serviços que não são necessários.
- Bloquear o sistema de arquivos para permitir somente acessos autorizados.
- Bloquear as alterações no registro.
- Um firewall deve ser configurado de acordo com os requisitos.
- A execução de script do PowerShell deve ser definida como **Restricted** ou **RemoteSigned**. Isso pode ser feito usando os comandos do PowerShell `Set-ExecutionPolicy -ExecutionPolicy Restricted` ou `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`.
- Habilitar a proteção avançada por meio do Internet Explorer.

- Restringir a capacidade de criar novos usuários e grupos.
- Remover o acesso à Internet e implementar servidores de salto para RDP.
- Proibir o registro em log de servidores que usam o RDP por meio da Internet. Em vez de usar a VPN de site a site, a VPN de ponto a site ou o express route para RDP em máquinas remotas de dentro da rede.
- Implantar regularmente todas as atualizações de segurança.
- Executar a ferramenta de gerenciador de conformidade de segurança no ambiente e implementar todas as suas recomendações.
- Monitorar o ambiente ativamente usando a Central de Segurança e o Operations Management Suite.
- Implantar appliances de rede virtual para rotear o tráfego para proxies internos e reversos.
- Todos os dados confidenciais, como configuração, cadeias de conexão e credenciais, devem ser criptografados.

Os itens mencionados acima são alguns dos principais pontos a serem considerados do ponto de vista da segurança. A lista continuará crescendo, e precisamos melhorar constantemente a segurança para evitar qualquer tipo de violação de segurança.

## Implementar servidores de salto

É recomendável remover o acesso à Internet de máquinas virtuais. Também é uma boa prática limitar a acessibilidade de serviços de área de trabalho remota da Internet, mas então como acessar as máquinas virtuais? Uma boa maneira é permitir apenas recursos internos para RDP em máquinas virtuais usando opções de VPN do Azure. No entanto, também há uma outra maneira, usando **servidores de salto**.

Os servidores de salto são servidores implantados na **zona desmilitarizada (DMZ)**. Isso significa que eles não estão na rede que hospeda as soluções e as aplicações principais. Em vez disso, estão numa rede ou sub-rede separada. A finalidade principal do servidor de salto é aceitar solicitações RDP de usuários e ajudá-los a iniciar sessão nele. Desse servidor de salto, os usuários ainda podem navegar para outras máquinas virtuais usando o RDP. Eles têm acesso a duas ou mais redes: uma que tem conectividade com o mundo externo e outra que é interna para a solução. O servidor de salto implementa todas as restrições de segurança e fornece um cliente seguro para conexão com outros servidores. Normalmente, o acesso a emails e à Internet está desabilitado em servidores de salto.

Um exemplo de implantação de um servidor de salto com **conjuntos de escalas de máquinas virtuais (VMSSes)**, usando modelos do Azure Resource Manager está disponível em <https://azure.microsoft.com/resources/templates/201-vmss-windows-jumpbox>.

## Azure Bastion

Na seção anterior, discutimos a implementação de servidores de salto. O Azure Bastion é um serviço totalmente gerenciado que pode ser provisionado em uma rede virtual para fornecer acesso RDP/SSH às suas máquinas virtuais diretamente no portal do Azure sobre o TLS. O host do Bastion atuará como um servidor de salto e eliminará a necessidade de endereços IP públicos para suas máquinas virtuais. O conceito de usar o Bastion é o mesmo que implementar um servidor de salto; no entanto, como esse é um serviço gerenciado, ele é totalmente gerenciado pelo Azure.

Como o Bastion é um serviço totalmente gerenciado do Azure e é fortalecido internamente, não precisamos aplicar NSGs adicionais na sub-rede dele. Além disso, como não estamos anexando IPs públicos às nossas máquinas virtuais, elas estão protegidas contra a verificação de portas.

## Segurança de aplicações

As aplicações Web podem ser hospedadas em soluções baseadas em IaaS sobre máquinas virtuais, e podem ser hospedadas em serviços gerenciados fornecidos pelo Azure, como o Serviço de Aplicativo. O Serviço de Aplicativo faz parte do paradigma de implantação de PaaS, e vamos analisá-lo na próxima seção. Nesta seção, analisaremos a segurança no nível da aplicação.

### SSL/TLS

O **Secure Socket Layer (SSL)** foi preterido e substituído pelo **Transport Layer Security (TLS)**. O TLS fornece segurança de ponta a ponta por meio de criptografia. Ele fornece dois tipos de criptografia:

- Simétrica: a mesma chave é disponibilizada para o remetente e o destinatário da mensagem, e é usada para a criptografia e a descriptografia dessa mensagem.
- Assimétrica: cada stakeholder tem duas chaves, uma chave privada e uma chave pública. A chave privada permanece no servidor ou com o usuário e é mantida em segredo, enquanto a chave pública é distribuída livremente para todos. Os titulares da chave pública a usam para criptografar a mensagem, que só pode ser descriptografada pela chave privada correspondente. Como a chave privada permanece com o proprietário, só ele pode descriptografar a mensagem. O **Rivest-Shamir-Adleman (RSA)** é um dos algoritmos usados para gerar esses pares de chaves pública e privada.
- As chaves também estão disponíveis em certificados popularmente conhecidos como certificados X.509, embora os certificados tenham mais detalhes além das chaves e sejam geralmente emitidos por autoridades de certificação confiáveis.

O TLS deve ser usado por aplicações Web para garantir que a troca de mensagens entre os usuários e o servidor seja segura e confidencial e que as identidades sejam protegidas. Esses certificados devem ser comprados de uma autoridade de certificação confiável, em vez de serem certificados autoassinados.

## Identidades gerenciadas

Antes de analisarmos as identidades gerenciadas, é importante saber como as aplicações foram criadas sem elas.

O método tradicional de desenvolvimento de aplicações é usar segredos, como nome de usuário, senha ou cadeias de conexão SQL, em arquivos de configuração. Colocar esses segredos em arquivos de configuração torna as alterações de aplicação nesses segredos fáceis e flexíveis sem modificar o código. Isso nos ajuda a manter o princípio "aberto para extensão, fechado para modificação". No entanto, essa abordagem tem uma desvantagem do ponto de vista da segurança. Os segredos podem ser visualizados por qualquer pessoa que tenha acesso aos arquivos de configuração, pois geralmente esses segredos são listados lá em texto não criptografado. Há alguns hacks para criptografá-los, mas eles não são infalíveis.

Uma maneira melhor de usar segredos e credenciais dentro de uma aplicação é armazená-los em um repositório de segredos, como o Azure Key Vault. O Azure Key Vault fornece segurança completa usando o **módulo de segurança de hardware (HSM)**, e os segredos são armazenados de forma criptografada com descriptografia sob demanda usando chaves armazenadas em hardware separado. Os segredos podem ser armazenados no Key Vault, com cada segredo tendo um nome de exibição e uma chave. A chave tem a forma de um URI que pode ser usado para fazer referência ao segredo de aplicações, conforme mostrado na Figura 8.5:

The screenshot shows the Azure Key Vault interface for managing secrets. The left sidebar has links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events (preview), Settings (with Keys and Secrets selected), and Help. The main area is titled 'keyvaultbook | Secrets' and contains a search bar and action buttons for Generate/Import, Refresh, and Restore Backup. A table lists three secrets:

Name	Type	Status
storageKey		✓ Enabled
storageKey1		✓ Enabled
storageKey2		✓ Enabled

Figura 8.5: armazenando segredos dentro de um cofre de chaves

Nos arquivos de configuração da aplicação, podemos nos referir ao segredo usando o nome ou a chave. No entanto, há outro desafio agora. Como a aplicação se conecta e autentica com o cofre de chaves?

Os cofres de chaves têm políticas de acesso que definem permissões para um usuário ou grupo no que diz respeito ao acesso a segredos e credenciais dentro deles. Os usuários, grupos ou aplicações de serviço que podem obter acesso são provisionados e hospedados no **Azure Active Directory (Azure AD)**. Embora contas de usuário individuais possam receber acesso usando as políticas de acesso do Key Vault, é uma prática melhor usar uma entidade de serviço para acessar o cofre de chaves. Uma entidade de serviço tem um identificador, também conhecido como ID de aplicação ou ID de cliente, juntamente com uma senha. A ID de cliente, juntamente com sua senha, pode ser usada para autenticar com o Azure Key Vault. Essa entidade de serviço pode ter permissão para acessar os segredos. As políticas de acesso do Azure Key Vault são concedidas no painel **Políticas de acesso** do cofre de chaves. Na Figura 8.6, você pode ver que a entidade de serviço, <https://keyvault.book.com>, concedeu acesso ao cofre de chaves chamado **keyvaultbook**:

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions	Action
APPLICATION					
<a href="https://keyvault.book.com">https://keyvault.book.com</a>		3 selected	3 selected	4 selected	Delete
<a href="http://automationcertcred2">http://automationcertcred2</a>		3 selected	3 selected	4 selected	Delete
USER					
Ritesh Modi	callritz_hotmail.com#...	9 selected	7 selected	15 selected	Delete

Figura 8.6: acesso concedido para uma entidade de serviço acessar um cofre de chaves

Isso nos leva a outro desafio: para acessar o cofre de chaves, precisamos usar a ID de cliente e o segredo em nossos arquivos de configuração para que possamos nos conectar ao cofre de chaves, obter o segredo e recuperar seu valor. Isso é quase equivalente ao uso de nome de usuário, senha e cadeia de conexão SQL em arquivos de configuração.

É nesse aspecto que as identidades gerenciadas podem ajudar. O Azure lançou as identidades de serviço gerenciadas e posteriormente as renomeou como identidades gerenciadas. As identidades gerenciadas são identidades gerenciadas pelo Azure. Em segundo plano, elas também criam uma entidade de serviço juntamente com uma senha. Com identidades gerenciadas, não há necessidade de colocar credenciais em arquivos de configuração.

As identidades gerenciadas só podem ser usadas para autenticar com serviços que ofereçam suporte ao Azure AD como um provedor de identidade. Elas se destinam apenas à autenticação. Se o serviço de destino não fornecer permissão de **controle de acesso baseado em função (RBAC)** à identidade, ela pode não ser capaz de executar a atividade pretendida nesse serviço.

Há dois tipos de identidades gerenciadas:

- Identidades gerenciadas atribuídas ao sistema
- Identidades gerenciadas atribuídas ao usuário

As identidades atribuídas ao sistema são geradas pelo próprio serviço. Por exemplo, se um serviço de aplicação quiser se conectar ao Banco de Dados SQL do Azure, ele poderá gerar a identidade gerenciada atribuída ao sistema como parte de suas opções de configuração. Essas identidades gerenciadas também são excluídas quando o recurso ou serviço pai é excluído. Como mostrado na Figura 8.7, uma identidade atribuída ao sistema pode ser usada pelo Serviço de Aplicativo para se conectar ao Banco de Dados SQL do Azure:

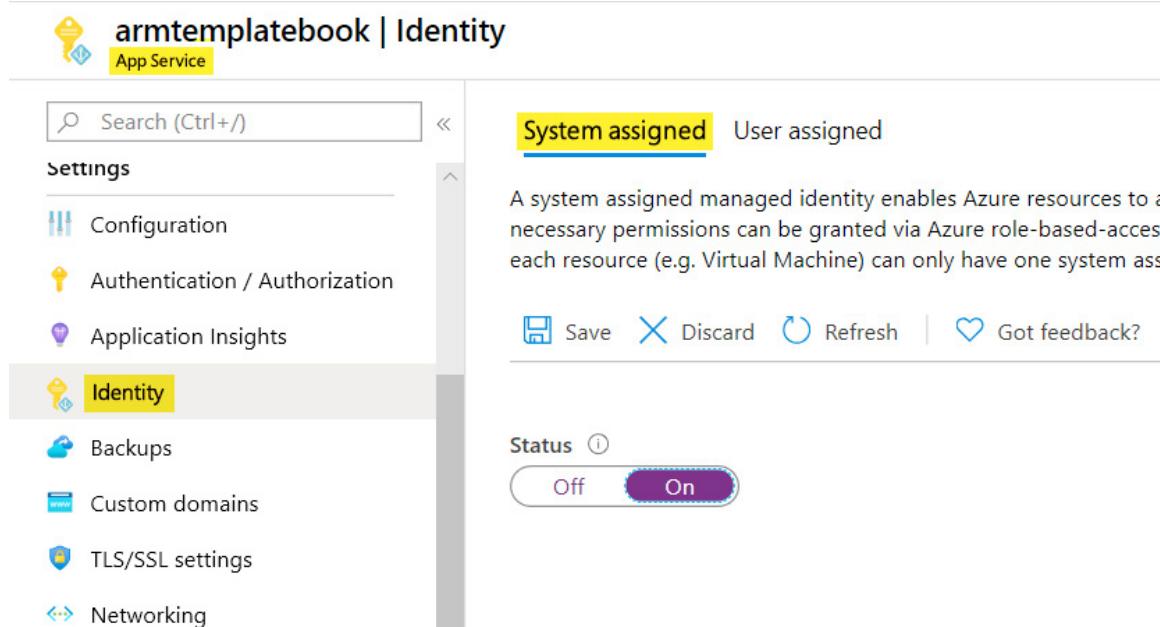


Figura 8.7: habilitando uma identidade gerenciada atribuída ao sistema para o Serviço de Aplicativo

As identidades gerenciadas atribuídas ao usuário são criadas como identidades separadas autônomas e posteriormente atribuídas aos serviços do Azure. Elas podem ser aplicadas e reutilizadas com vários serviços do Azure, pois seus ciclos de vida não dependem do recurso a que são atribuídas.

Depois que uma identidade gerenciada é criada e permissões de RBAC ou de acesso são concedidas a ela no recurso de destino, ela pode ser usada dentro de aplicações para acessar os recursos e serviços de destino.

O Azure fornece um SDK, bem como uma API REST, para falar com o Azure AD, obter um token de acesso para identidades gerenciadas e, em seguida, usar o token para acessar e consumir os recursos de destino.

O SDK faz parte do pacote NuGet **Microsoft.Azure.Services.AppAuthentication** para C#. Quando o token de acesso estiver disponível, ele poderá ser usado para consumir o recurso de destino.

O código necessário para obter o token de acesso é o seguinte:

```
var tokenProvider = new AzureServiceTokenProvider();  
string token = await tokenProvider.GetAccessTokenAsync("https://vault.azure.net");
```

Como alternativa, use isto:

```
string token = await tokenProvider.GetAccessTokenAsync("https://database.windows.net");
```

Deve-se observar que o código da aplicação precisa ser executado no contexto do Serviço de Aplicativo ou de uma aplicação de função porque a identidade é anexada a eles e só está disponível no código quando ele é executado dentro deles.

O código anterior tem dois casos de uso diferentes. Os códigos para acessar o cofre de chaves e o Banco de Dados SQL do Azure são exibidos juntos.

É importante observar que as aplicações não fornecem nenhuma informação relacionada a identidades gerenciadas no código e são totalmente gerenciadas usando a configuração. Os desenvolvedores, administradores de aplicações individuais e operadores não se depararão com credenciais relacionadas a identidades gerenciadas e, além disso, não há nenhuma menção a elas no código. A rotação de credenciais é completamente regulamentada pelo provedor de recursos que hospeda o serviço do Azure. A rotação padrão ocorre a cada 46 dias. Cabe ao provedor de recursos solicitar novas credenciais, se necessário, para que ele possa esperar mais de 46 dias.

Na próxima seção, discutiremos um **gerenciador de eventos e informações de segurança (SIEM)** nativo de nuvem: Azure Sentinel.

## Azure Sentinel

O Azure fornece SIEM e **resposta automatizada de orquestração de segurança (SOAR)** como um serviço autônomo que pode ser integrado a qualquer implantação personalizada no Azure. A Figura 8.8 mostra alguns dos principais recursos do Azure Sentinel:

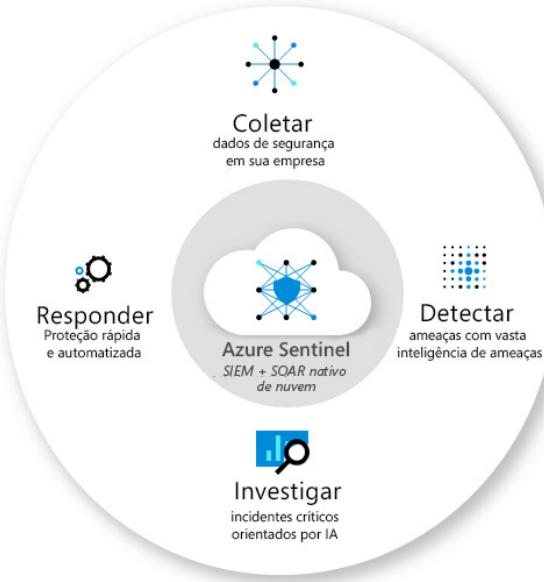


Figura 8.8: principais recursos do Azure Sentinel

O Azure Sentinel coleta logs de informações de implantações e recursos e executa análises para encontrar padrões e tendências relacionados a vários problemas de segurança que são extraídos de fontes de dados.

Deve haver monitoramento ativo do ambiente, os logs devem ser coletados e as informações devem ser selecionadas desses logs como uma atividade separada da implementação de código. É aqui que o serviço SIEM entra em cena. Há inúmeros conectores que podem ser usados com o Azure Sentinel; cada um desses conectores será usado para adicionar fontes de dados ao Azure Sentinel. O Azure Sentinel fornece conectores para serviços da Microsoft, como Office 365, Azure AD e Proteção contra Ameaças do Azure. Os dados coletados alimentarão um espaço de trabalho do Log Analytics e você poderá escrever consultas para pesquisar esses logs.

As ferramentas SIEM, como o Azure Sentinel, podem ser habilitadas no Azure para obter todos os logs do Log Analytics e da Central de Segurança do Azure que, por sua vez, podem obtê-los de várias fontes, implantações e serviços. O SIEM pode então executar sua inteligência sobre esses dados coletados e gerar insights. Ele pode gerar relatórios e painéis com base em inteligência descoberta para consumo, mas também pode investigar atividades suspeitas e ameaças, além de tomar medidas em relação a elas.

Embora o Azure Sentinel possa parecer muito semelhante em funcionalidade à Central de Segurança do Azure, ele é capaz de fazer muito mais do que ela. Sua capacidade de coletar logs de outros caminhos usando conectores o torna diferente da Central de Segurança do Azure.

## Segurança de PaaS

O Azure fornece inúmeros serviços de PaaS, cada um com seus próprios recursos de segurança. Em geral, os serviços de PaaS podem ser acessados usando credenciais, certificados e tokens. Eles permitem a geração de tokens de acesso de segurança de curta duração. As aplicações cliente podem enviar esses tokens de acesso de segurança para representar usuários confiáveis. Nesta seção, vamos abordar alguns dos mais importantes serviços de PaaS que são usados em quase todas as soluções.

### Link Privado do Azure

O Link Privado do Azure fornece acesso a serviços de PaaS do Azure, bem como serviços compartilhados com parceiros ou de propriedade de clientes hospedados no Azure por meio de um ponto de extremidade privado na sua rede virtual. Ao usar o Link Privado do Azure, não precisamos expor nossos serviços à Internet pública, e todo o tráfego entre o serviço e a rede virtual passa pela rede de backbone da Microsoft.

O Ponto de Extremidade Privado do Azure é a interface de rede que ajuda a conectar-se de forma privada e segura a um serviço habilitado pelo Link Privado do Azure. Como o ponto de extremidade privado é mapeado para a instância do serviço de PaaS, não para todo o serviço, os usuários só podem se conectar ao recurso. As conexões com qualquer outro serviço são negadas, e isso protege contra o vazamento de dados. O Ponto de Extremidade Privado também permite o acesso seguro na infraestrutura local via ExpressRoute ou Túneis VPN. Isso elimina a necessidade de configurar o emparelhamento público ou passar pela Internet pública para alcançar o serviço.

### Gateway de Aplicativo do Azure

O Azure fornece um balanceador de carga de Nível 7 conhecido como Gateway de Aplicativo do Azure que não serve apenas para balancear a carga, mas também ajudar no roteamento usando valores na URL. Ele também tem um recurso conhecido como Firewall de Aplicativo Web. O Gateway de Aplicativo do Azure oferece suporte à terminação TLS no gateway, para que os servidores de back-end recebam o tráfego não criptografado. Isso tem várias vantagens, como melhor performance, melhor utilização dos servidores de back-end e roteamento inteligente de pacotes. Na seção anterior, discutimos o Firewall do Azure e como ele protege os recursos no nível da rede. O Firewall de Aplicativo Web, por outro lado, protege a implantação no nível da aplicação.

Qualquer aplicação implantada exposta à Internet enfrenta inúmeros desafios de segurança. Algumas das principais ameaças à segurança são:

- Scripts entre sites
- Execução remota de código
- Injeção de SQL
- Ataques de **Negação de Serviço (DoS)**
- Ataques de **Negação de Serviço Distribuído (DDoS)**

Mas há muitas outras.

Um grande número desses ataques pode ser abordado por desenvolvedores, escrevendo código defensivo e seguindo as práticas recomendadas; no entanto, não é apenas o código que deve ser responsável por identificar esses problemas em um site online. O Firewall de Aplicativo Web configura regras que podem identificar esses problemas, como mencionado anteriormente, e negar solicitações.

É aconselhável usar recursos de Gateway de Aplicativo e Firewall de Aplicativo Web para proteger aplicações contra ameaças dinâmicas à segurança. O Firewall de Aplicativo Web permitirá a passagem da solicitação ou a interromperá, dependendo de como foi configurado.

## Azure Front Door

O Azure lançou um serviço relativamente novo conhecido como Azure Front Door. A função do Azure Front Door é bastante semelhante à função do Gateway de Aplicativo do Azure; no entanto, há uma diferença no escopo. Enquanto o Gateway de Aplicativo funciona em uma única região, o Azure Front Door funciona no nível global entre regiões e datacenters. Ele também tem um firewall de aplicação Web que pode ser configurado para proteger aplicações implantadas em várias regiões contra várias ameaças à segurança, como injeção de SQL, execução remota de código e scripts entre sites.

O Gateway de Aplicativo pode ser implantado atrás do Front Door para resolver a descarga de conexão. Além disso, a implantação do Gateway de Aplicativo atrás do Front Door ajudará com o requisito de balanceamento de carga, pois o Front Door só pode executar o balanceamento de carga baseado em caminho no nível global. A adição do Gateway de Aplicativo à arquitetura fornecerá mais balanceamento de carga aos servidores de back-end na rede virtual.

## Ambiente do Serviço de Aplicativo do Azure

O Serviço de Aplicativo do Azure é implantado em redes compartilhadas nos bastidores. Todos os SKUs do Serviço de Aplicativo usam uma rede virtual, que pode potencialmente ser usada por outros locatários também. Para ter mais controle e uma implantação segura do Serviço de Aplicativo no Azure, os serviços podem ser hospedados em redes virtuais dedicadas. Isso pode ser feito usando o **Ambiente do Serviço de Aplicativo do Azure (ASE)**, que fornece isolamento completo para executar seu Serviço de Aplicativo em grande escala. Isso também fornece segurança adicional, permitindo que você implante Firewall do Azure, Grupos de Segurança de Aplicações, NSGs, Gateway de Aplicativo, Firewall de Aplicativo Web e Azure Front Door. Todos os planos do Serviço de Aplicativo criados no Ambiente do Serviço de Aplicativo estarão em uma camada de preços isolada, e não podemos escolher outra camada.

Todos os logs dessa rede virtual e da computação podem ser agrupados no Azure Log Analytics, na Central de Segurança e, finalmente, no Azure Sentinel.

O Azure Sentinel pode então fornecer insights e executar pastas de trabalho e runbooks para responder a ameaças à segurança de forma automatizada. Os guias estratégicos de segurança podem ser executados no Azure Sentinel em resposta a alertas. Todo guia estratégico de segurança é composto por medidas que precisam ser tomadas em caso de alertas. Os guias estratégicos são baseados em Aplicativos Lógicos do Azure, e isso fornecerá a liberdade de usar e personalizar os modelos internos disponíveis com Aplicativos Lógicos.

## Log Analytics

O Log Analytics é uma nova solução analítica para o gerenciamento de implantações de nuvem, datacenters na infraestrutura local e soluções híbridas.

Ele fornece várias soluções modulares, funcionalidade específica que ajuda a implementar um recurso. Por exemplo, as soluções de segurança e auditoria ajudam a determinar uma visão completa de segurança para a implantação de uma organização. Da mesma forma, existem muitas outras soluções, como automação e controle de alterações, que devem ser implementadas do ponto de vista da segurança. Os serviços de segurança e auditoria do Log Analytics fornecem informações nas cinco categorias a seguir:

- **Domínios de segurança:** eles permitem exibir registros de segurança, avaliações de malware, avaliações de atualização, segurança de rede, informações de identidade e de acesso e computadores com eventos de segurança. O acesso também é fornecido para o painel da Central de Segurança do Azure.
- **Avaliação de antimalware:** ajuda na identificação de servidores que não estão protegidos contra malware e têm problemas de segurança. Ela fornece informações sobre exposição a possíveis problemas de segurança e avalia o nível de importância de qualquer risco. Os usuários podem tomar medidas proativas com base nessas recomendações. As subcategorias da Central de Segurança do Azure fornecem informações coletadas pela Central de Segurança do Azure.

- Principais problemas:** esta categoria identifica rapidamente problemas ativos e classifica a gravidade deles.
- Detecções:** esta categoria está no modo de visualização. Ela permite a identificação de padrões de ataque por meio da visualização de alertas de segurança.
- Inteligência contra ameaças:** ajuda na identificação de padrões de ataque visualizando o número total de servidores com tráfego IP de saída mal-intencionado, o tipo de ameaça mal-intencionada e um mapa que mostra de onde esses IPs vieram.

Os detalhes anteriores, quando visualizados no portal, são mostrados na Figura 8.9:

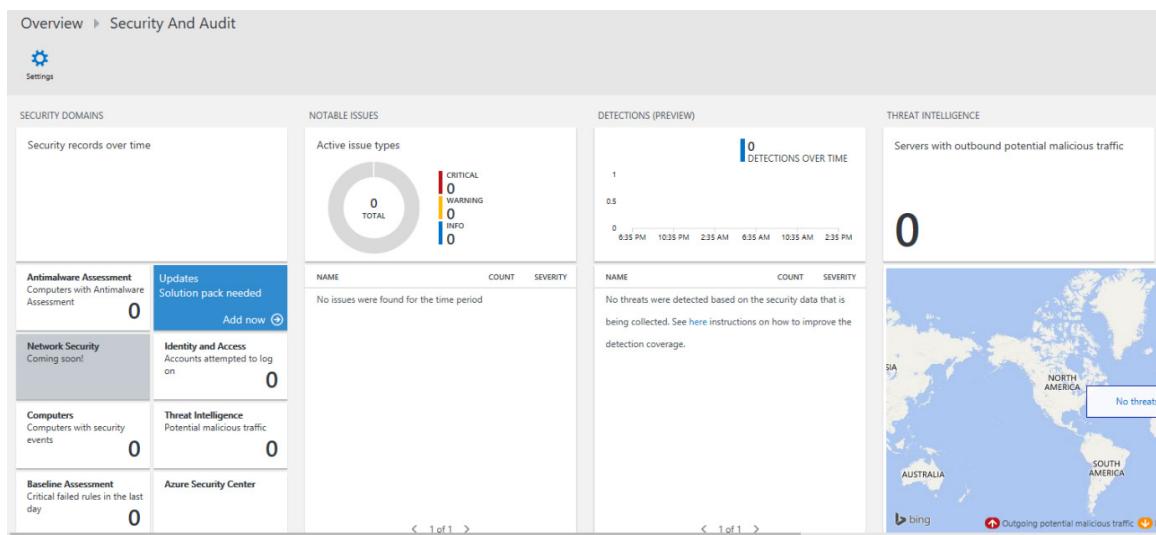


Figura 8.9: informações em exibição no painel Segurança e Auditoria do Log Analytics

Agora que você aprendeu sobre a segurança para serviços de PaaS, vamos explorar como proteger os dados armazenados no Armazenamento do Azure.

## Armazenamento do Azure

As contas de armazenamento desempenham um papel importante na arquitetura geral da solução. Elas podem armazenar informações importantes, como **informações de identificação pessoal (PII)** de usuários, transações comerciais e outros dados confidenciais. É de extrema importância que as contas de armazenamento sejam seguras e permitam apenas o acesso de usuários autorizados. Os dados armazenados são criptografados e transmitidos por meio de canais seguros. O armazenamento, assim como os usuários e as aplicações cliente que utilizam a conta de armazenamento e seus dados, desempenha um papel crucial na segurança geral dos dados. Os dados devem permanecer criptografados o tempo todo. Isso também inclui credenciais e cadeias de conexão que se conectam a repositórios de dados.

O Azure fornece RBAC para controlar quem pode gerenciar contas de armazenamento dele. Essas permissões de RBAC são concedidas a usuários e grupos no Azure AD. No entanto, quando uma aplicação a ser implantada no Azure for criada, ela terá usuários e clientes que não estão disponíveis no Azure AD. Para permitir que os usuários acessem a conta de armazenamento, o Armazenamento do Azure fornece armazenamento de chaves de acesso. Há dois tipos de chaves de acesso no nível da conta de armazenamento: primário e secundário. Os usuários com essas chaves podem se conectar à conta de armazenamento. Essas chaves de acesso de armazenamento são usadas na etapa de autenticação ao acessar a conta de armazenamento. As aplicações podem acessar contas de armazenamento usando chaves primárias ou secundárias. Duas chaves são fornecidas de modo que, se a chave primária for comprometida, as aplicações poderão ser atualizadas para usar a chave secundária, enquanto a primária é gerada novamente. Isso ajuda a minimizar o tempo de inatividade da aplicação. Ademais, isso aumenta a segurança ao remover a chave comprometida sem afetar as aplicações. Os detalhes da chave de armazenamento, como visto no portal do Azure, são mostrados na Figura 8.10:

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines. [Learn more](#)

The screenshot shows the 'Default keys' section of the Azure Storage account settings. It includes a table with columns for NAME, KEY, and CONNECTION STRING. Two rows are present: 'key1' and 'key2'. Each row has a copy icon (blue square with white text) and a refresh/circular arrow icon.

NAME	KEY	CONNECTION STRING
key1	[REDACTED]	DefaultEndpointsProtocol=https;AccountName= [REDACTED]
key2	[REDACTED]	DefaultEndpointsProtocol=https;AccountName= [REDACTED]

Figura 8.10: chaves de acesso para uma conta de armazenamento

O Armazenamento do Azure fornece quatro serviços, blob, arquivos, filas e tabelas, em uma conta. Cada um desses serviços também fornece a infraestrutura para sua própria segurança usando tokens de acesso seguro.

Uma **assinatura de acesso compartilhado (SAS)** é um URI que concede direitos de acesso restrito a serviços de Armazenamento do Azure: blobs, arquivos, filas e tabelas. Esses tokens SAS podem ser compartilhados com clientes que não são confiáveis a ponto de receber a chave da conta de armazenamento inteira para restringir o acesso a determinados recursos da conta. Ao distribuir um URI de SAS a esses clientes, o acesso a recursos é concedido por um período especificado.

Os tokens SAS existem na conta de armazenamento e nos níveis individuais de blob, arquivo, tabela e fila. Uma assinatura no nível da conta de armazenamento é mais poderosa e tem os direitos para permitir e negar permissões no nível do serviço individual. Também pode ser usada em vez de níveis do serviço de recurso individual.

Os tokens SAS fornecem acesso granular a recursos e também podem ser combinados. Esses tokens incluem leitura, gravação, exclusão, listagem, adição, criação, atualização e processamento. Além disso, até mesmo o acesso a recursos pode ser determinado durante a geração de tokens SAS. Poderia ser para blobs, tabelas, filas e arquivos individualmente ou uma combinação deles. As chaves da conta de armazenamento são para a conta inteira, elas não podem ser restritas a serviços individuais, nem podem ser restritas do ponto de vista das permissões. É muito mais fácil criar e revogar tokens SAS do que fazer isso com as chaves de acesso da conta de armazenamento. Os tokens SAS podem ser criados para uso por um determinado período, após o qual eles se tornam inválidos automaticamente.

Lembre-se de que, se as chaves de conta de armazenamento forem regeneradas, o token SAS baseado nelas será invalidado, e um novo token SAS deverá ser criado e compartilhado com os clientes. Na Figura 8.11, você pode ver uma opção para selecionar o escopo, as permissões, a data de início, a data de término, o endereço IP permitido, os protocolos permitidos e a chave de assinatura para criar um token SAS:

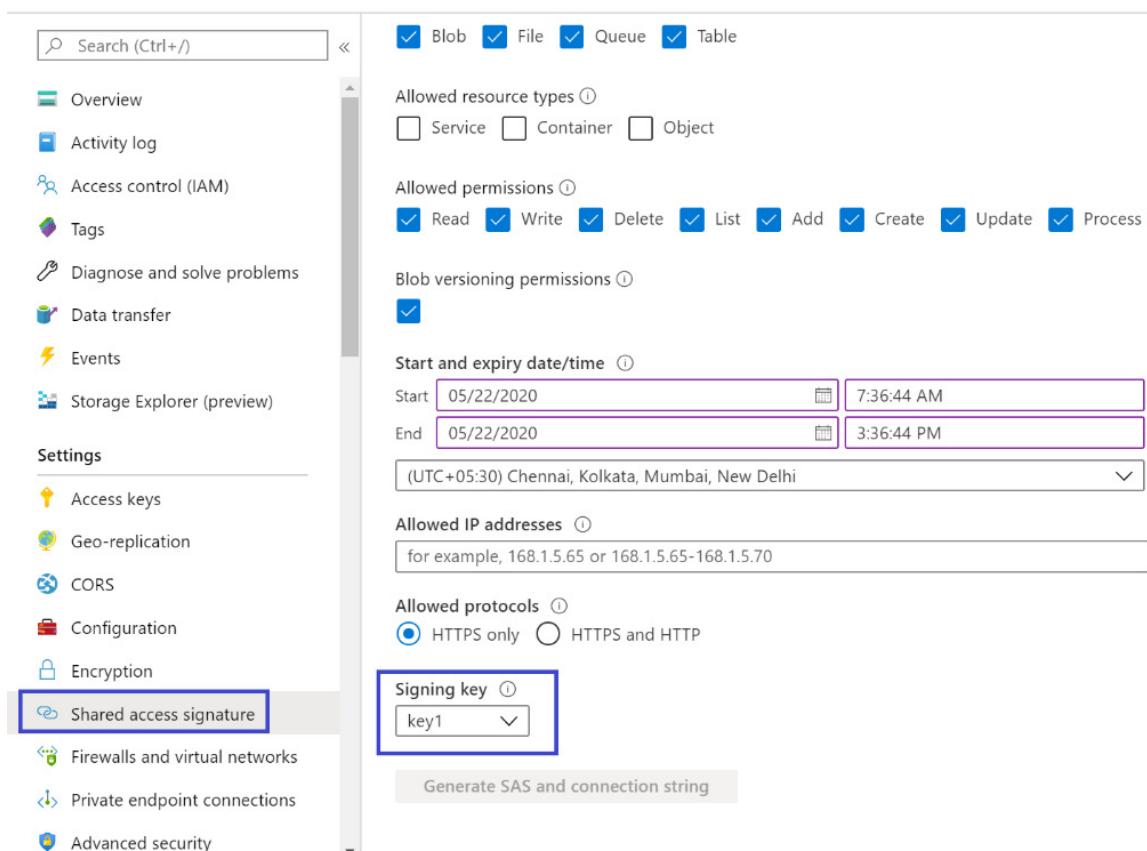


Figura 8.11: criando um token SAS

Se estivermos regenerando a **key1**, que usamos para assinar o token SAS no exemplo anterior, precisaremos criar um novo token SAS com a **key2** ou a nova **key1**.

Roubo de cookies, injeção de scripts e ataques de DoS são meios comuns usados por invasores para violar um ambiente e roubar dados. Os navegadores e o protocolo HTTP implementam um mecanismo embutido que garante que essas atividades mal-intencionadas não possam ser executadas. Geralmente, qualquer coisa entre domínios não é permitida por HTTP e ou navegadores. Um script em execução em um domínio não pode solicitar recursos de outro domínio. No entanto, há casos de uso válidos em que tais solicitações devem ser autorizadas. O protocolo HTTP implementa o **compartilhamento de recursos entre origens (CORS)**. Com a ajuda do CORS, é possível acessar recursos entre domínios e fazê-los funcionar. O Armazenamento do Azure configura regras de CORS para recursos de blob, arquivo, fila e tabela. Ele permite a criação de regras que são avaliadas para cada solicitação autenticada. Se as regras forem atendidas, a solicitação terá permissão para acessar o recurso. Na Figura 8.12, você pode ver como adicionar regras de CORS a cada um dos serviços de armazenamento:

Figura 8.12: criando regras de CORS para uma conta de armazenamento

Os dados não devem apenas ser protegidos em trânsito, eles também devem ser protegidos em repouso. Se os dados em repouso não forem criptografados, qualquer um que tiver acesso à unidade física no datacenter poderá lê-los. Embora a possibilidade de uma violação de dados seja insignificante, os clientes ainda devem criptografar seus dados. A criptografia do serviço de armazenamento também ajuda a proteger dados em repouso. Esse serviço funciona de forma transparente e injeta a si mesmo sem que os usuários tomem conhecimento. Ele criptografa dados quando eles são salvos em uma conta de armazenamento e os descriptografa automaticamente quando eles são lidos. Todo esse processo acontece sem que os usuários executem qualquer atividade adicional.

As chaves de conta do Azure devem ser giradas periodicamente. Isso garantirá que um invasor não seja capaz de obter acesso a contas de armazenamento.

Também é uma boa ideia regenerar as chaves; no entanto, isso deve ser avaliado em relação ao seu uso em aplicações existentes. Se isso interromper as aplicações existentes, essas aplicações deverão ser priorizadas para o gerenciamento de alterações, que deverão ser aplicadas gradualmente.

É sempre recomendável ter tokens SAS no nível do serviço individual com prazos limitados. Esse token só deve ser fornecido aos usuários que devem acessar os recursos. Sempre siga o princípio do privilégio mínimo e forneça apenas as permissões necessárias.

As chaves SAS e as chaves da conta de armazenamento devem ser armazenadas no Azure Key Vault. Isso fornece armazenamento e acesso seguro a elas. Essas chaves podem ser lidas no tempo de execução por aplicações do cofre de chaves, em vez de armazená-las em arquivos de configuração.

Além disso, você também pode usar o Azure AD para autorizar as solicitações para o armazenamento de blobs e filas. Vamos usar o RBAC para conceder as permissões necessárias a uma entidade de serviço e, depois que autenticarmos a entidade de serviço usando o Azure AD, um token OAuth 2.0 será gerado. Esse token poderá ser adicionado ao cabeçalho de autorização das suas chamadas à API para autorizar uma solicitação em relação ao armazenamento de blobs ou filas. A Microsoft recomenda o uso da autorização do Azure AD ao trabalhar com aplicações de blobs e filas devido à segurança superior proporcionada pelo Azure AD e sua simplicidade em comparação com tokens SAS.

Na próxima seção, vamos avaliar as opções de segurança disponíveis para o Banco de Dados SQL do Azure.

## SQL do Azure

O SQL Server armazena dados relacionais no Azure, que é um serviço de banco de dados relacional gerenciado. Ele também é conhecido como um **Banco de Dados como Serviço (DBaaS)** que fornece uma plataforma altamente disponível, escalável, centrada na performance e segura para armazenar dados. É acessível de qualquer lugar, com qualquer plataforma e linguagem de programação. Os clientes precisam de uma cadeia de conexão composta por servidor, banco de dados e informações de segurança para se conectarem a ele.

O SQL Server fornece configurações de firewall que impedem o acesso de qualquer pessoa por padrão. Os endereços IP e os intervalos devem ser incluídos na lista de permissões para acessar o SQL Server. Os arquitetos só devem permitir os endereços IP nos quais possam confiar e que pertençam a clientes/parceiros. Existem implantações no Azure para as quais há vários endereços IP ou os endereços IP não são conhecidos, como aplicações implantadas no Azure Functions ou em Aplicativos Lógicos. Para que tais aplicações acessem o SQL do Azure, ele permite uma lista de permissões de todos os endereços IP para serviços Azure em assinaturas.

Lembre-se de que a configuração do firewall está no nível do servidor, e não no nível do banco de dados. Isso significa que quaisquer alterações aqui afetam todos os bancos de dados dentro de um servidor. Na Figura 8.13, você pode ver como adicionar IPs de clientes ao firewall para conceder acesso ao servidor:

**Firewall settings**  
dbemployerecords-prod (SQL server)

Save Discard + Add client IP

Deny public network access  Yes  No

Setting to **Yes** allows connections via approved private endpoint only and disables any existing firewall rules. [Learn more.](#)

Minimal TLS Version  > 1.0  >1.1  >1.2

You are setting the Minimal TLS Version property for all SQL Database and SQL Data Warehouse databases associated with the server. Any login attempts from clients using TLS version less than the Minimal TLS Version shall be rejected.

Connection Policy  Default  Proxy  Redirect

Allow Azure services and resources to access this server  Yes  No

Connections from the IPs specified below provides access to all the databases in dbemployerecords-prod.

Rule name	Start IP	End IP	...
allow-vendors	56.17.113.55	56.17.113.55	...
allow-internal	172.17.1.0	172.17.1.250	...

Connections from the VNET/Subnet specified below provides access to all databases in dbemployerecords-prod.

Figura 8.13: configurando regras de firewall

O SQL do Azure também fornece segurança avançada ao criptografar os dados em repouso. Isso garante que ninguém, incluindo os administradores de datacenter do Azure, possa exibir os dados armazenados no SQL Server. A tecnologia usada pelo SQL Server para criptografar dados em repouso é conhecida como **Transparent Data Encryption (TDE)**. Não há nenhuma alteração necessária no nível da aplicação para implementar a TDE. O SQL Server criptografa e descriptografa dados de forma transparente quando o usuário salva e lê os dados. Esse recurso está disponível no nível do banco de dados. Também podemos integrar a TDE ao Azure Key Vault para ter **Bring Your Own Key (BYOK)**. Usando BYOK, podemos habilitar a TDE usando uma chave gerenciada pelo cliente no Azure Key Vault.

O SQL Server também fornece **máscara dinâmica de dados (DDM)**, que é especialmente útil para mascarar determinados tipos de dados, como dados de cartão de crédito ou PII de usuários. O mascaramento não é igual à criptografia. Ele não criptografa os dados, apenas os mascara, o que garante que eles não estejam em um formato legível por humanos. Os usuários devem mascarar e criptografar dados confidenciais no SQL do Azure Server.

O SQL Server também fornece um serviço de auditoria e detecção de ameaças para todos os servidores. Existem serviços avançados de coleta de dados e inteligência em execução nesses bancos de dados para descobrir ameaças e vulnerabilidades e alertar os usuários com base neles. Os logs de auditoria são mantidos pelo Azure em contas de armazenamento e podem ser exibidos por administradores para serem acionados. As ameaças, como injeção de SQL e logons de clientes anônimos, podem gerar alertas que podem ser enviados para os administradores por email. Na Figura 8.14, você pode ver como habilitar a Detecção de Ameaças:

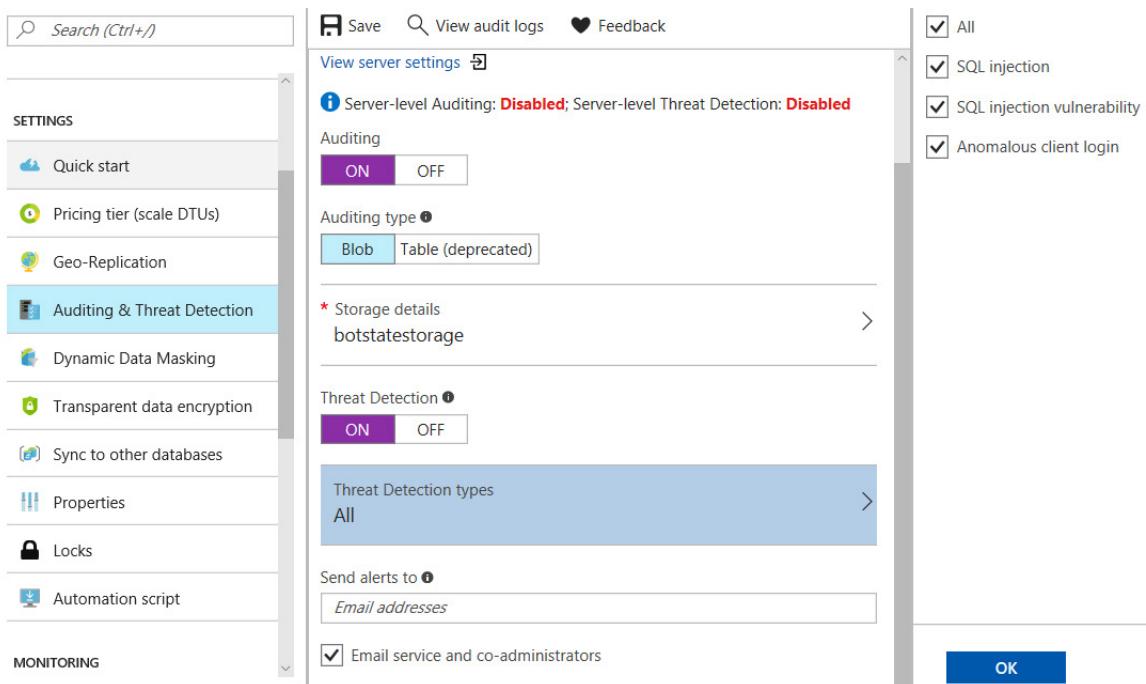


Figura 8.14: habilitando a Proteção contra Ameaças e selecionar os tipos de ameaças a serem detectados

Os dados podem ser mascarados no SQL do Azure. Isso nos ajuda a armazenar dados em um formato que não pode ser lido por humanos:

The screenshot shows the 'Masking rules' section with a 'MASK NAME' column and a 'MASK FUNCTION' column. A message states 'You haven't created any masking rules.' Below this, a note says 'SQL users excluded from masking (administrators are always excluded)' with a green checkmark. The 'Recommended fields to mask' section shows 'SCHEMA', 'TABLE', and 'COLUMN' columns, with a message 'There are no recommended fields to mask'.

On the right, there's a configuration panel for 'Mask name' (text input), 'Select what to mask' (Schema, Table, Column dropdowns), and 'Select how to mask' (Masking field format dropdown with 'Default value (0, xxxx, 01-01-1900)').

Figura 8.15: definindo as configurações para mascarar dados

O SQL do Azure também fornece a TDE para criptografar dados em repouso, conforme mostrado na Figura 8.16:

The left sidebar shows a 'SETTINGS' menu with options: Quick start, Pricing tier (scale DTUs), Geo-Replication, Auditing & Threat Detection, Dynamic Data Masking, and Transparent data encryption (which is highlighted with a blue background).

The main area displays a shield icon with the text 'Encrypts your databases, backups enable TDE, go to each database.' and a 'Learn more' link. It includes a 'Data encryption' toggle switch set to 'ON', an 'Encryption status' section showing 'Encrypted' with a green checkmark, and a 'Transparent data encryption' section.

Figura 8.16: habilitando a TDE

Para realizar uma avaliação de vulnerabilidade no SQL Server, você pode aproveitar a Avaliação de Vulnerabilidade do SQL, que faz parte do pacote unificado para recursos avançados de segurança do SQL conhecido como Segurança de Dados Avançada. A Avaliação de Vulnerabilidade do SQL pode ser usada pelos clientes de forma proativa para melhorar a segurança do banco de dados, descobrindo, rastreando e ajudando você a corrigir possíveis vulnerabilidades do banco de dados.

Mencionamos o Azure Key Vault algumas vezes nas seções anteriores, quando discutimos identidades gerenciadas, Banco de Dados SQL e assim por diante. Você já sabe qual é o objetivo do Azure Key Vault e, na próxima seção, exploraremos alguns métodos que podem ajudar a proteger o conteúdo do seu cofre de chaves.

## Azure Key Vault

A proteção de recursos usando senhas, chaves, credenciais, certificados e identificadores exclusivos é um elemento importante de qualquer ambiente e aplicação do ponto de vista da segurança. Eles precisam ser protegidos e garantir que esses recursos permaneçam seguros e não sejam comprometidos é um importante pilar da arquitetura de segurança. O gerenciamento e as operações que mantêm os segredos e as chaves protegidos enquanto os disponibiliza quando necessário são aspectos importantes que não podem ser ignorados. Normalmente, esses segredos são usados por todo o lado, no código-fonte, dentro de arquivos de configuração, em pedaços de papel e em outros formatos digitais. Para superar esses desafios e armazenar todos os segredos uniformemente em um repositório centralizado e seguro, o Azure Key Vault deve ser usado.

O Azure Key Vault é bem integrado a outros serviços do Azure. Por exemplo, seria fácil usar um certificado armazenado no Azure Key Vault e implantá-lo no armazenamento de certificados de máquinas virtuais do Azure. Todos os tipos de chaves, incluindo chaves de armazenamento, chaves de IoT e eventos e cadeias de conexão, podem ser armazenados como segredos no Azure Key Vault. Eles podem ser recuperados e usados de forma transparente sem que ninguém os exiba ou armazene temporariamente em qualquer lugar. As credenciais do SQL Server e outros serviços também podem ser armazenadas no Azure Key Vault.

O Azure Key Vault funciona por região. Isso significa que um recurso do Azure Key Vault deve ser provisionado na mesma região em que a aplicação e o serviço são implantados. Se uma implantação consistir em mais de uma região e precisar dos serviços do Azure Key Vault, várias instâncias dele deverão ser provisionadas.

Um recurso importante do Azure Key Vault é que os segredos, chaves e certificados não são mantidos no armazenamento geral. O backup desses dados confidenciais é feito pelo HSM. Isso significa que esses dados são armazenados em hardware separado no Azure que só podem ser desbloqueados por chaves pertencentes a usuários. Para fornecer segurança adicional, você também pode implementar pontos de extremidade de serviço de rede virtual para o Azure Key Vault. Isso restringirá o acesso ao cofre de chaves para redes virtuais específicas. Você também pode restringir o acesso a um intervalo de endereços IPv4.

Na seção *Armazenamento do Azure*, discutimos o uso do Azure AD para autorizar solicitações a blobs e filas. Foi mencionado que usamos um token OAuth, que é obtido do Azure AD, para autenticar chamadas à API. Na próxima seção, você verá como a autenticação e a autorização são feitas usando OAuth. Depois de concluir a próxima seção, você poderá relacioná-la ao que discutimos na seção *Armazenamento do Azure*.

## Autenticação e autorização usando OAuth

O Azure AD é um provedor de identidade que pode autenticar os usuários com base em usuários e entidades de serviço já disponíveis no locatário. Ele implementa o protocolo OAuth e oferece suporte à autorização na Internet. Ele implementa um servidor de autorização e serviços para habilitar o fluxo de autorização de OAuth, fluxos implícitos e de credenciais de clientes. Esses são diferentes fluxos de interação OAuth bem documentados entre aplicações cliente, pontos de extremidade de autorização, usuários e recursos protegidos.

O Azure AD também oferece suporte ao **logon único (SSO)**, que adiciona segurança e facilidade ao entrar em aplicações registradas no Azure AD. Você pode usar os métodos OpenID Connect, OAuth, SAML, baseado em senha, ou o método SSO vinculado ou desabilitado ao desenvolver novas aplicações. Se você não tiver certeza de qual usar, consulte o fluxograma da Microsoft aqui: <https://docs.microsoft.com/azure/active-directory/manage-apps/what-is-single-sign-on#choosing-a-single-sign-on-method>.

Aplicações Web, aplicações baseadas em JavaScript e aplicações cliente nativas (como aplicações móveis e de área de trabalho) podem usar o Azure AD para autenticação e autorização. Há plataformas de redes sociais, como Facebook, Twitter etc., que oferecem suporte ao protocolo OAuth para autorização.

Uma das maneiras mais fáceis de habilitar a autenticação para aplicações Web usando o Facebook é mostrada a seguir. Há outros métodos que usam binários de segurança, mas isso está fora do escopo deste livro.

Neste passo a passo, um Serviço de Aplicativo do Azure será provisionado juntamente com um Plano do Serviço de Aplicativo para hospedar uma aplicação Web personalizada. Uma conta válida do Facebook será necessária como pré-requisito a fim de redirecionar os usuários para ela para autenticação e autorização.

Um novo grupo de recursos pode ser criado usando o portal do Azure, como mostrado na Figura 8.17:

**Create a resource group**

**Basics** Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

**Project details**

Subscription \* ⓘ RiteshSubscription

Resource group \* ⓘ FaceauthRG

**Resource details**

Region \* ⓘ (US) East US

Figura 8.17: criando um novo grupo de recursos

Depois que o grupo de recursos for criado, um novo serviço de aplicação poderá ser criado usando o portal, como mostrado na Figura 8.18:

**Web App**

**Project Details**

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ RiteshSubscription

Resource Group \* ⓘ FaceauthRG

[Create new](#)

**Instance Details**

Name \* faceauthenticationdemo .azurewebsites.net

Publish \* [Code](#) Docker Container

Runtime stack \* NET Core 2.1 (LTS)

Operating System \* Linux Windows

Region \* Central US

Not finding your App Service Plan? Try a different region.

**App Service Plan**

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Windows Plan (Central US) \* ⓘ (New) ASP-FaceauthRG-87df

[Create new](#)

[Review + create](#) < Previous Next : Monitoring >

Figura 8.18: criando uma nova aplicação

É importante anotar a URL da aplicação Web porque ela será necessária posteriormente ao configurar a aplicação do Facebook.

Depois que a aplicação Web for provisionada no Azure, a próxima etapa será criar uma nova aplicação no Facebook. Isso é necessário para representar sua aplicação Web no Facebook e gerar as credenciais de cliente apropriadas para ela. É assim que o Facebook conhece a aplicação Web.

Navegue até [developers.facebook.com](https://developers.facebook.com) e faça logon usando as credenciais apropriadas. Crie uma nova aplicação selecionando a opção **Criar Aplicativo** em **Meus Aplicativos** no canto superior direito, como mostrado na Figura 8.19:



Figura 8.19: criando uma nova aplicação no portal do desenvolvedor do Facebook

A página da Web solicitará que você forneça um nome para a aplicação Web a fim de criar uma nova aplicação no Facebook:

## Create a New App ID

Get started integrating Facebook into your app or website

Display Name

AzureforArchitectsoauth

Contact Email

callrites@yahoo.com

This email address is used to contact you about potential policy violations, app restrictions or steps to recover the app if it's been deleted or compromised.

By proceeding, you agree to the [Facebook Platform Policies](#)

[Cancel](#)

[Create App ID](#)

Figura 8.20: adicionando uma nova aplicação

Adicione um novo produto de **Logon no Facebook** e clique em **Configurar** para configurar o logon para que a aplicação Web personalizada seja hospedada no Serviço de Aplicativo do Azure:

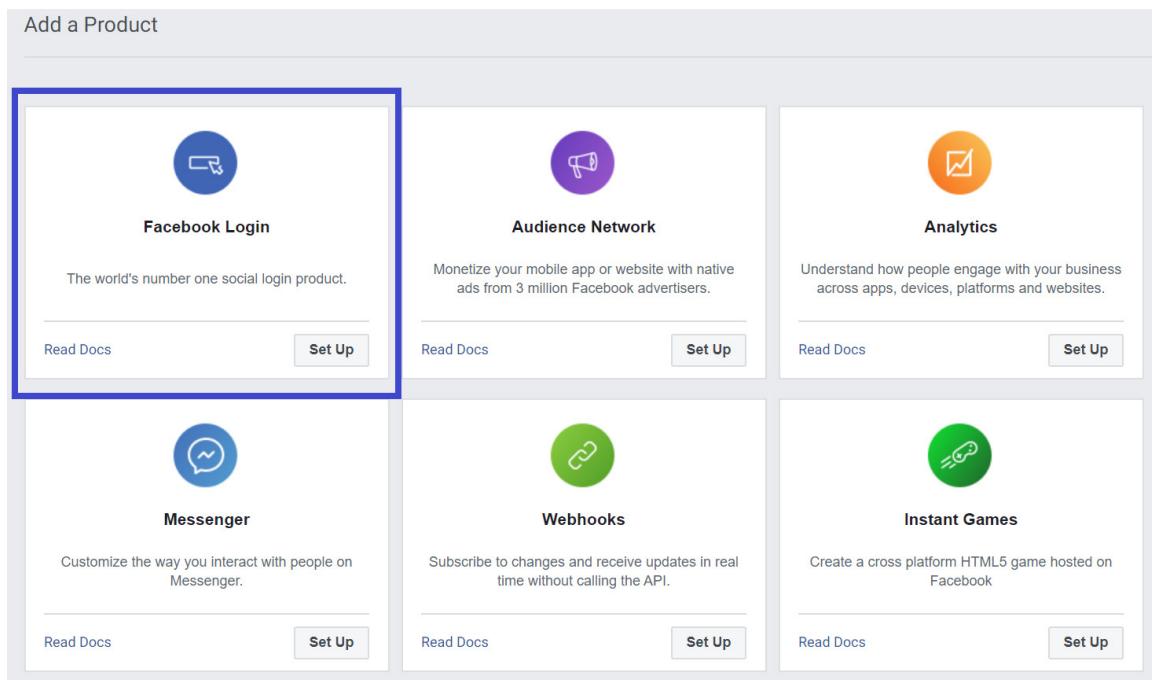


Figura 8.21: adicionando logon no Facebook à aplicação

O botão **Configurar** fornece algumas opções, conforme mostrado na Figura 8.22, e essas opções configuram o fluxo de OAuth, como fluxo de autorização, fluxo implícito ou fluxo de credenciais de cliente. Selecione a opção **Web** porque é ela que precisa da autorização do Facebook:

**Use o Início rápido para adicionar o logon do Facebook ao seu aplicativo. Para começar, selecione a plataforma para este aplicativo.**



Figura 8.22: selecionando a plataforma

Forneça a URL da aplicação Web que anotamos anteriormente após o provisionamento da aplicação Web no Azure:

### 1. Tell Us about Your Website

Tell us what the URL of your site is.

#### Site URL

`https://faceauthenticationdemo.azurewebsites.net/`

**Save**

Figura 8.23: fornecendo a URL do site à aplicação

Clique no item **Configurações** no menu à esquerda e forneça a URL de redirecionamento OAuth para a aplicação. O Azure já tem URLs de retorno de chamada bem definidas para cada uma das plataformas de redes sociais populares, e a usada para o Facebook é **domain name/.auth/login/facebook/callback**:

The screenshot shows the Facebook for Developers interface. On the left sidebar, under 'Facebook Login', the 'Settings' option is selected. In the main content area, the 'Client OAuth Settings' section is displayed. It includes several configuration options with checkboxes:
 

- Client OAuth Login**: Yes (selected). Description: Enables OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]
- Web OAuth Login**: Yes (selected). Description: Enables web-based Client OAuth Login. [?]
- Enforce HTTPS**: Yes (selected). Description: Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]
- Force Web OAuth Reauthentication**: No (selected). Description: When on, prompts people to enter their Facebook password in order to log in on the web. [?]
- Embedded Browser OAuth Login**: No (selected). Description: Enable webview Redirect URIs for Client OAuth Login. [?]
- Use Strict Mode for Redirect URIs**: Yes (selected). Description: Only allow redirects that use the Facebook SDK or that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

 Below these settings, there is a 'Valid OAuth Redirect URIs' field containing the URL `https://faceauthenticationdemo.azurewebsites.net/.auth/login/facebook/callback`. There is also a 'Login from Devices' checkbox set to 'No'.

Figura 8.24: adicionando URIs de redirecionamento OAuth

Vá para as configurações **Básicas** no menu à esquerda e anote os valores de **ID do Aplicativo** e **Segredo do Aplicativo**. Eles são necessários para configurar a autenticação/autorização dos Serviços de Aplicativo do Azure:

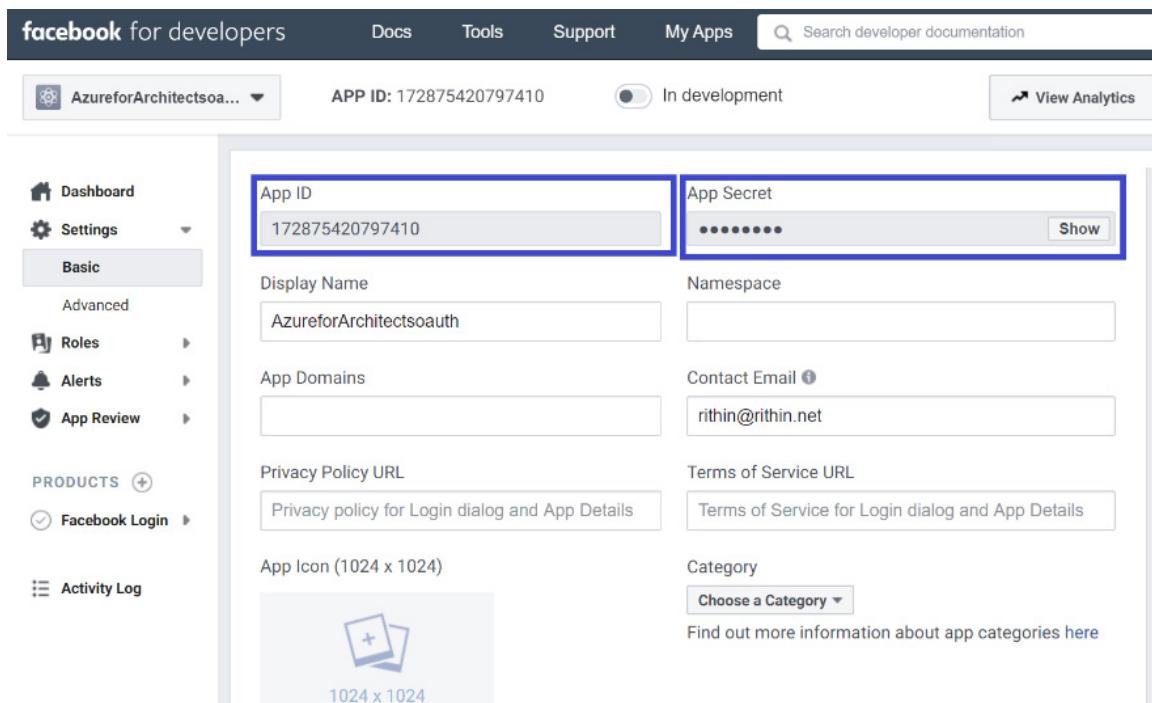


Figura 8.25: encontrando a ID do Aplicativo e o Segredo do Aplicativo

No portal do Azure, volte para o Serviço de Aplicativo do Azure criado nas primeiras etapas desta seção e navegue até a folha de autenticação/autorização. Ative a **Autenticação do Serviço de Aplicativo**, selecione **Fazer logon com o Facebook** para autenticação e clique no item **Facebook** na lista:

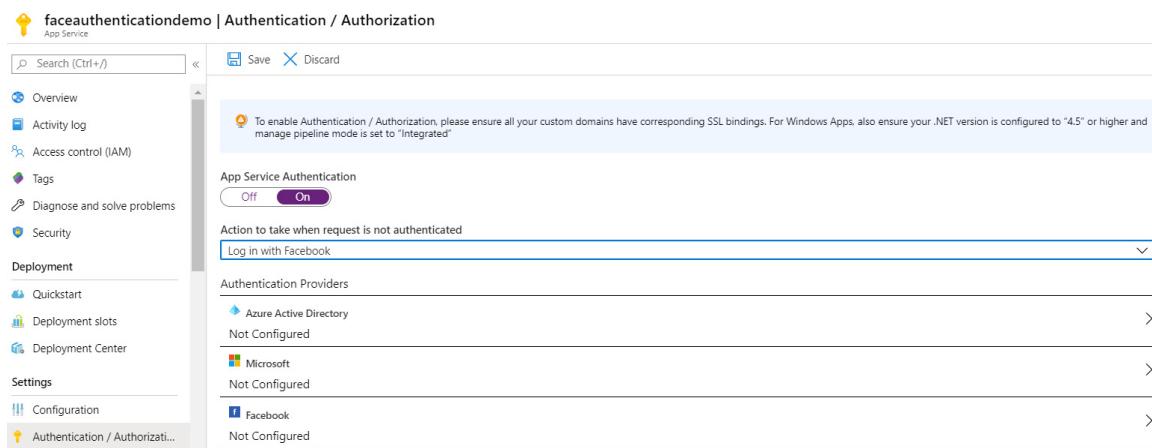
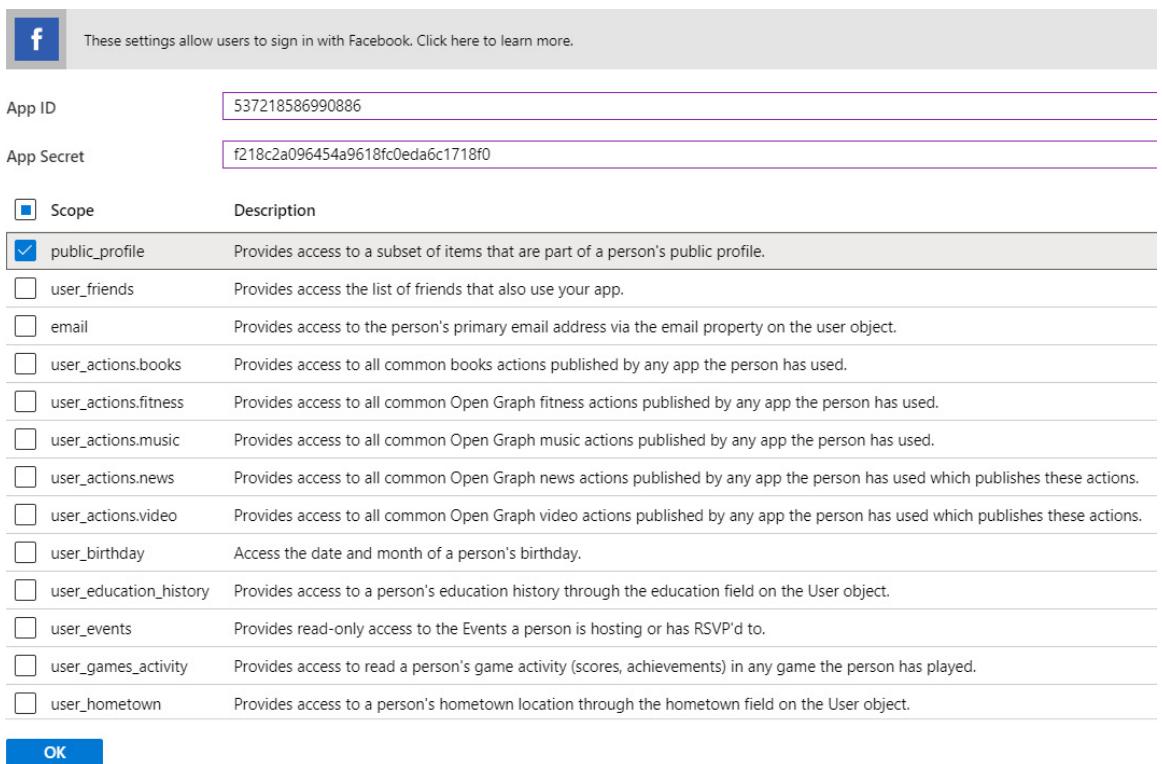


Figura 8.26: habilitando a autenticação do Facebook no Serviço de Aplicativo

Na página resultante, forneça a ID do aplicativo e o segredo do aplicativo anotados, e também selecione o escopo. O escopo decide as informações compartilhadas pelo Facebook com a aplicação Web:



These settings allow users to sign in with Facebook. Click here to learn more.

App ID	537218586990886
App Secret	f218c2a096454a9618fc0eda6c1718f0
<input type="checkbox"/> Scope	Description
<input checked="" type="checkbox"/> public_profile	Provides access to a subset of items that are part of a person's public profile.
<input type="checkbox"/> user_friends	Provides access the list of friends that also use your app.
<input type="checkbox"/> email	Provides access to the person's primary email address via the email property on the user object.
<input type="checkbox"/> user_actions.books	Provides access to all common books actions published by any app the person has used.
<input type="checkbox"/> user_actions.fitness	Provides access to all common Open Graph fitness actions published by any app the person has used.
<input type="checkbox"/> user_actions.music	Provides access to all common Open Graph music actions published by any app the person has used.
<input type="checkbox"/> user_actions.news	Provides access to all common Open Graph news actions published by any app the person has used which publishes these actions.
<input type="checkbox"/> user_actions.video	Provides access to all common Open Graph video actions published by any app the person has used which publishes these actions.
<input type="checkbox"/> user_birthday	Access the date and month of a person's birthday.
<input type="checkbox"/> user_education_history	Provides access to a person's education history through the education field on the User object.
<input type="checkbox"/> user_events	Provides read-only access to the Events a person is hosting or has RSVP'd to.
<input type="checkbox"/> user_games_activity	Provides access to read a person's game activity (scores, achievements) in any game the person has played.
<input type="checkbox"/> user_hometown	Provides access to a person's hometown location through the hometown field on the User object.

**OK**

Figura 8.27: selecionando o escopo

Clique em **OK** e clique no botão **Salvar** para salvar as configurações de autenticação/autorização.

Agora, se uma nova sessão anônima do navegador for iniciada e você acessar a aplicação Web personalizada, a solicitação será redirecionada para o Facebook. Como você pode ter visto em outros sites, ao usar **Fazer logon com o Facebook**, será necessário fornecer suas credenciais:

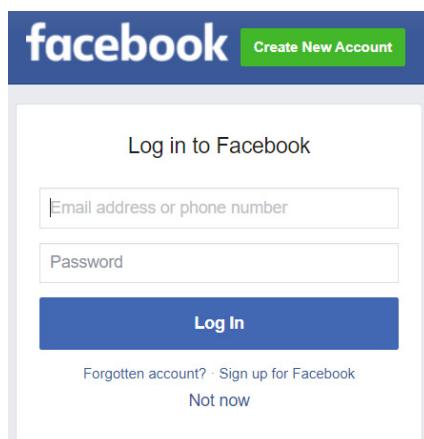


Figura 8.28: fazendo logon no site usando o Facebook

Depois que você inserir suas credenciais, uma caixa de diálogo de consentimento do usuário solicitará permissão para que os dados do Facebook sejam compartilhados com a aplicação Web:

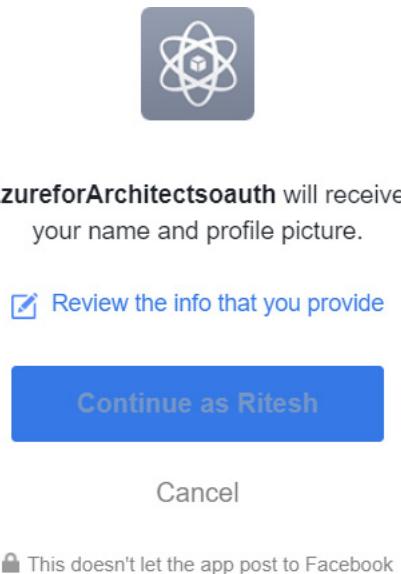


Figura 8.29: consentimento do usuário para compartilhar suas informações com a aplicação

Se o consentimento for concedido, a página da Web da aplicação Web aparecerá:

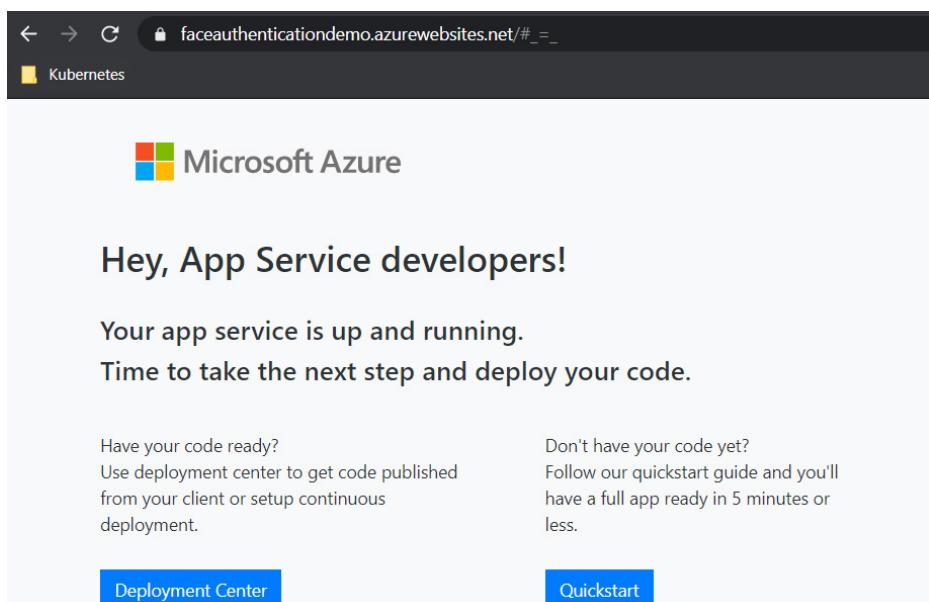


Figura 8.30: acessando a página de destino

Uma abordagem semelhante pode ser usada para proteger sua aplicação Web usando Azure AD, Twitter, Microsoft e Google. Você também pode integrar seu próprio provedor de identidade, se necessário.

A abordagem mostrada aqui ilustra apenas uma das maneiras de proteger um site usando credenciais armazenadas em outro lugar e a autorização de aplicações externas para acessar recursos protegidos. O Azure também fornece bibliotecas de JavaScript e assemblies .NET para usar o método de programação imperativo a fim de consumir os pontos de extremidade OAuth fornecidos pelo Azure AD e outras plataformas de redes sociais. É recomendável usar essa abordagem para obter maior controle e flexibilidade para autenticação e autorização nas suas aplicações.

Até agora, discutimos os recursos de segurança e como eles podem ser implementados. Também é relevante ter monitoramento e auditoria em vigor. A implementação de uma solução de auditoria ajudará sua equipe de segurança a auditar os logs e tomar medidas preventivas. Na próxima seção, discutiremos as soluções de monitoramento e auditoria de segurança no Azure.

## Monitoramento e auditoria de segurança

Todas as atividades no seu ambiente, desde emails até a alteração de um firewall, podem ser categorizadas como um evento de segurança. Do ponto de vista da segurança, é necessário ter um sistema de registro em log central para monitorar e rastrear as alterações feitas. Durante uma auditoria, se você encontrar atividades suspeitas, poderá descobrir qual é a falha na arquitetura e como ela pode ser corrigida. Caso você tenha tido uma violação de dados, os logs ajudarão os profissionais de segurança a entender o padrão de um ataque e como ele foi executado. Além disso, as medidas preventivas necessárias podem ser tomadas para evitar incidentes semelhantes no futuro. O Azure fornece os dois recursos de segurança importantes a seguir para gerenciar todos os aspectos de segurança da assinatura, dos grupos de recursos e dos recursos do Azure:

- Azure Monitor
- Central de Segurança do Azure

Desses dois recursos de segurança, primeiro exploraremos o Azure Monitor.

### Azure Monitor

O Azure Monitor é uma solução completa para o monitoramento de recursos do Azure. Ele fornece informações sobre os recursos do Azure e seu estado. Também oferece uma interface de consulta avançada com informações que podem ser divididas usando dados nos níveis de assinatura, grupo de recursos, recurso individual e tipo de recurso. O Azure Monitor coleta dados de várias fontes de dados, incluindo métricas e logs do Azure, aplicações de clientes e os agentes executados em máquinas virtuais. Outros serviços, como a Central de Segurança do Azure e o Observador de Rede, também ingerem dados no espaço de trabalho do Log Analytics, que podem ser analisados no Azure Monitor. Você pode usar APIs REST para enviar dados personalizados para o Azure Monitor.

O Azure Monitor pode ser usado por meio do portal do Azure, do PowerShell, da CLI e das APIs REST:

The screenshot shows the Azure Activity log interface. At the top, there are navigation links for 'Dashboard' and 'Activity log'. Below that is a search bar and a 'Quick Insights' button. Filter options include 'Management Group: None', 'Subscription: 2 selected', 'Timespan: Last 6 hours', 'Event severity: All', and a 'Add Filter' button. The main area displays 13 items of activity logs with columns for Operation name, Status, Time, Time stamp, and Subscription.

Operation name	Status	Time	Time stamp	Subscription
> Delete resource group	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Delete SQL server	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Update SQL database	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Update SQL database	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Update SQL server	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Validate Deployment	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Registers the Microsoft SQL Database Resource Provider	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Check Server Name Availability	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Check Server Name Availability	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Registers the Microsoft SQL Database Resource Provider	Succeeded	3 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Delete Disk	Failed	3 hours ago	Fri May 22 ...	[PROD] rithin.net
> List Storage Account Keys	Succeeded	4 hours ago	Fri May 22 ...	Project Hawk Eye 360
> Incident	Resolved	6 hours ago	Fri May 22 ...	Project Hawk Eye 360

Figura 8.31: explorando logs de atividades

Os logs a seguir são aqueles fornecidos pelo Azure Monitor:

- **Log de atividades:** mostra todas as operações no nível do gerenciamento realizadas nos recursos. Ele fornece detalhes sobre a hora de criação, o criador, o tipo de recurso e o status dos recursos.
- **Log de operações (clássico):** fornece detalhes de todas as operações realizadas nos recursos dentro de um grupo de recursos e de uma assinatura.
- **Métricas:** obtêm informações sobre performance para recursos individuais e definem alertas neles.
- **Configurações de diagnóstico:** ajudam na configuração dos logs de efeitos por meio da configuração do Armazenamento do Azure para armazenar logs, transmiti-los em tempo real para Hubs de Eventos do Azure e enviá-los ao Log Analytics.
- **Pesquisa de logs:** ajuda na integração do Log Analytics ao Azure Monitor.

O Azure Monitor pode identificar incidentes relacionados à segurança e tomar medidas apropriadas. É importante que somente pessoas autorizadas possam acessar o Azure Monitor, já que ele pode conter informações confidenciais.

## Central de Segurança do Azure

A Central de Segurança do Azure, como o nome sugere, é uma solução completa para todas as necessidades de segurança. Geralmente há duas atividades relacionadas à segurança: implementação de segurança e monitoramento para quaisquer ameaças e violações. A Central de Segurança foi criada principalmente para ajudar com essas duas atividades. A Central de Segurança do Azure permite que os usuários definam suas políticas de segurança e as implementem nos recursos do Azure. Com base no estado atual dos recursos do Azure, a Central de Segurança do Azure fornece recomendações de segurança para fortalecer a solução e os recursos individuais do Azure.

As recomendações incluem quase todas as práticas recomendadas de segurança do Azure, incluindo criptografia de dados e discos, proteção de redes, proteção de pontos de extremidade, listas de controle de acesso, listas de permissões de solicitações de entrada e bloqueio de solicitações não autorizadas. Os recursos abrangem desde componentes de infraestrutura, como平衡adores de carga, grupos de segurança de rede e redes virtuais, até recursos de PaaS, como SQL e Armazenamento do Azure. Veja um trecho do painel **Visão geral** da Central de Segurança do Azure, que mostra a classificação geral de segurança da assinatura, a higiene de segurança dos recursos e muito mais:

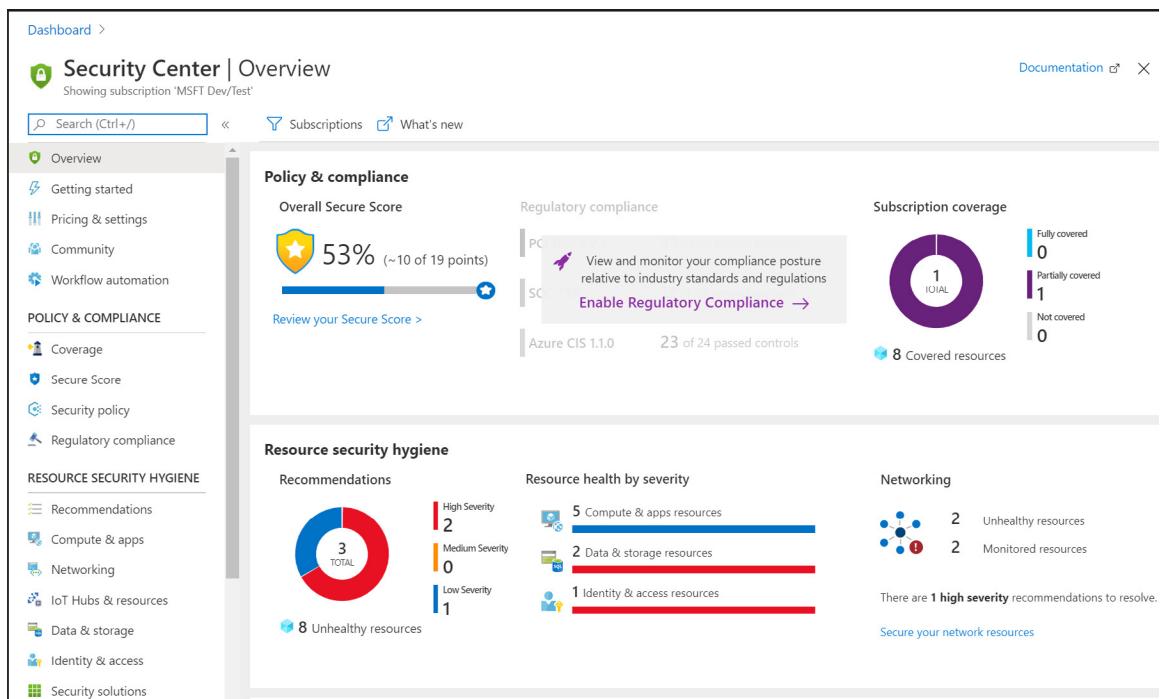


Figura 8.32: visão geral da Central de Segurança do Azure

A Central de Segurança do Azure é uma plataforma avançada que fornece recomendações para vários serviços, conforme mostrado na Figura 8.33. Além disso, essas recomendações podem ser exportadas para arquivos CSV para referência:

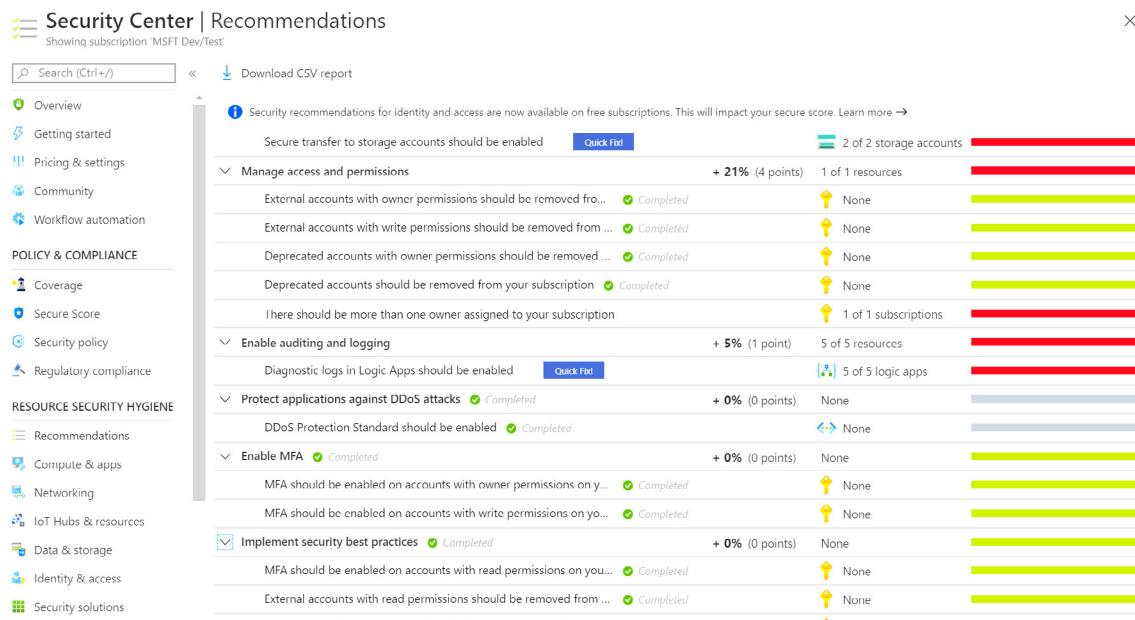


Figura 8.33: recomendações da Central de Segurança do Azure

Como mencionado no início desta seção, o monitoramento e a auditoria são cruciais em um ambiente corporativo. O Azure Monitor pode ter várias fontes de dados e pode ser usado para auditar logs dessas fontes. A Central de Segurança do Azure fornece avaliações contínuas e recomendações de segurança priorizadas, juntamente com a classificação geral de segurança.

## Resumo

A segurança é sempre um aspecto importante de qualquer implantação ou solução. Ela se tornou muito mais importante e relevante por causa da implantação na nuvem. Além disso, há uma ameaça crescente de ataques cibernéticos. Em tais circunstâncias, a segurança tornou-se um ponto focal para as organizações. Não importa o tipo de implantação ou solução, seja IaaS, PaaS ou SaaS, a segurança é necessária em todos eles. Os datacenters do Azure são completamente seguros e têm diversas certificações internacionais de segurança. Elas são seguras por padrão. Eles fornecem recursos de segurança de IaaS, como NSGs, conversão de endereços de rede, pontos de extremidade seguros, certificados, cofres de chaves, armazenamento, criptografia de máquinas virtuais, e recursos de segurança de PaaS para recursos individuais de PaaS. A segurança tem seu próprio ciclo de vida completo, e ele deve ser adequadamente planejado, projetado, implementado e testado, assim como qualquer outra funcionalidade de aplicação.

---

Discutimos os firewalls do sistema operacional e o Firewall do Azure e como eles podem ser aproveitados para aumentar o panorama geral de segurança da sua solução. Também exploramos novos serviços do Azure, como Azure Bastion, Azure Front Door e Link Privado do Azure.

A segurança das aplicações foi outra área fundamental, e discutimos a execução de autenticação e autorização usando OAuth. Fizemos uma demonstração rápida de como criar um serviço de aplicação e integrar o logon do Facebook. O Facebook foi apenas um exemplo; você pode usar Google, Twitter, Microsoft, Azure AD ou qualquer provedor de identidade personalizado.

Também exploramos as opções de segurança oferecidas pelo SQL do Azure, que é um serviço de banco de dados gerenciado fornecido pelo Azure. Discutimos a implementação de recursos de segurança e, na seção final, concluímos com monitoramento e auditoria usando o Azure Monitor e a Central de Segurança do Azure. A segurança desempenha um papel essencial no seu ambiente. Um arquiteto deve sempre projetar e arquitetar soluções com a segurança como um dos principais pilares da arquitetura; o Azure fornece muitas opções para fazer isso.

Agora que você sabe como proteger seus dados no Azure, no próximo capítulo, vamos nos concentrar em soluções de big data do Hadoop, seguidas por Data Lake Storage, Data Lake Analytics e Data Factory.



# 9

## Soluções de Big Data do Azure

No capítulo anterior, você aprendeu sobre as várias estratégias de segurança que podem ser implementadas no Azure. Com uma aplicação segura, gerenciamos grandes quantidades de dados. O big data vem ganhando uma força significativa ao longo dos últimos anos. Ferramentas especializadas, software e armazenamento são necessários para lidar com ele. Curiosamente, essas ferramentas, plataformas e opções de armazenamento não estavam disponíveis como serviços há alguns anos.

No entanto, com uma nova tecnologia de nuvem, o Azure fornece inúmeras ferramentas, plataformas e recursos para criar soluções de big data com facilidade. Este capítulo detalhará a arquitetura completa para ingerir, limpar, filtrar e visualizar dados de forma significativa.

Os tópicos a seguir serão abordados neste capítulo:

- Visão geral de big data
- Integração de dados
- **Extrair-Transformar-Carregar (ETL)**
- Data Factory
- Data Lake Storage
- Ecossistemas de ferramentas como Spark, Databricks e Hadoop
- Databricks

## Big data

Com o influxo de dispositivos baratos, como dispositivos de Internet das Coisas e dispositivos portáteis, a quantidade de dados que estão sendo gerados e capturados aumentou exponencialmente. Quase todas as organizações têm uma grande quantidade de dados e estão prontas para comprar mais, se necessário. Quando grandes quantidades de dados chegam em vários formatos diferentes e com uma frequência cada vez maior, podemos dizer que estamos lidando com big data. Em suma, há três características principais de big data:

- **Volume:** a quantidade de dados em termos de tamanho (em GB, TB e PB, por exemplo) e em termos do número de registros (como em um milhão de linhas em um repositório de dados hierárquico, 100.000 imagens, meio bilhão de documentos JSON e assim por diante).
- **Velocidade:** refere-se à velocidade com que os dados chegam ou são ingeridos. Se os dados não forem alterados com frequência ou se dados novos não chegarem com frequência, a velocidade dos dados será baixa, por outro lado, se houver atualizações frequentes e muitos dados novos chegarem de forma contínua e frequente, eles terão velocidade alta.
- **Variedade:** refere-se a diferentes tipos e formatos de dados. Os dados podem vir de diferentes fontes em diferentes formatos. Os dados podem chegar como dados estruturados (por exemplo, em arquivos separados por vírgula, arquivos JSON ou dados hierárquicos), bancos de dados semiestruturados (por exemplo, em documentos NoSQL sem esquema) ou dados não estruturados (como objetos binários grandes (blobs), imagens, PDFs etc.). Com tantas variantes, é importante ter um processo definido para processar dados ingeridos.

Na próxima seção, vamos conferir o processo geral de big data.

## Processo de big data

Quando os dados vêm de várias fontes em diferentes formatos e em diferentes velocidades, é importante definir um processo de armazenamento, assimilação, filtragem e limpeza de dados de uma forma que nos ajude a trabalhar com esses dados com mais facilidade e torná-los úteis para outros processos. É necessário haver um processo bem definido para o gerenciamento de dados. O processo geral de big data que deve ser seguido é mostrado na Figura 9.1:



Figura 9.1: processo de big data

Há quatro estágios principais de processamento de big data. Vamos explorá-los em detalhes:

- **Ingestão:** este é o processo de trazer e ingerir dados no ambiente de big data. Os dados podem vir de várias fontes, e os conectores devem ser usados para ingerir esses dados na plataforma de big data.
- **Armazenamento:** após a ingestão, os dados devem ser armazenados no pool de dados para armazenamento de longo prazo. O repositório deve estar disponível para dados históricos e dados online, além disso, deve ser capaz de armazenar dados estruturados, semiestruturados e não estruturados. Deve haver conectores para ler os dados de fontes de dados, ou as fontes de dados devem ser capazes de enviar dados para o armazenamento.
- **Análise:** depois que os dados forem lidos no repositório, eles deverão ser analisados, um processo que requer filtragem, agrupamento, junção e transformação de dados para obter insights.
- **Visualização:** as análises podem ser enviadas como relatórios usando várias plataformas de notificação ou usadas para gerar painéis com gráficos.

Anteriormente, as ferramentas necessárias para capturar, ingerir, armazenar e analisar big data não estavam prontamente disponíveis para as organizações devido ao envolvimento de hardware caro e grandes investimentos. Além disso, nenhuma plataforma estava disponível para processá-las. Com o advento da nuvem, tornou-se mais fácil para as organizações capturar, ingerir, armazenar e realizar análises de big data usando as ferramentas e estruturas de sua preferência. Elas podem pagar ao provedor de nuvem para usar sua infraestrutura e evitar despesas de capital. Além disso, o custo da nuvem é muito baixo em comparação com qualquer solução na infraestrutura local.

O big data exige uma enorme quantidade de recursos de computação, armazenamento e rede. Geralmente, não é prático manter a quantidade de recursos necessários em uma única máquina ou servidor. Mesmo que, de alguma forma, recursos suficientes sejam disponibilizados em um único servidor, o tempo necessário para processar um pool de big data inteiro é consideravelmente grande, pois cada trabalho é feito em sequência e cada etapa tem uma dependência da etapa anterior. Há uma necessidade de estruturas especializadas e ferramentas que possam distribuir o trabalho em vários servidores e, eventualmente, trazer os resultados deles e apresentá-los ao usuário depois de combinar de forma adequada os resultados de todos os servidores. Essas ferramentas são ferramentas especializadas de big data que ajudam na obtenção de disponibilidade, escalabilidade e distribuição imediata para garantir que uma solução de big data possa ser otimizada para ser executada rapidamente com robustez e estabilidade incorporadas.

Os dois proeminentes serviços de big data do Azure são o HD Insights e o Databricks. Vamos explorar as várias ferramentas disponíveis no panorama de big data.

## Ferramentas de big data

Há muitos serviços e ferramentas no espaço de big data, e vamos abordar alguns deles neste capítulo.

### Azure Data Factory

O Azure Data Factory é o principal serviço ETL no Azure. Ele define dados de entrada (em termos de seu formato e esquema), transforma dados de acordo com filtros e regras de negócios, aumenta dados existentes e, por fim, transfere dados para um repositório de destino prontamente consumível por outros serviços downstream. Ele é capaz de executar pipelines (contendo lógica ETL) no Azure, bem como infraestrutura personalizada, e também pode executar pacotes de SQL Server Integration Services.

### Azure Data Lake Storage

O Azure Data Lake Storage é um repositório de big data de nível empresarial que é resiliente, altamente disponível e possui segurança imediata. Ele é compatível com o Hadoop e pode escalar para petabytes de armazenamento de dados. Ele é baseado em contas de armazenamento do Azure e, portanto, obtém todos os benefícios das contas de armazenamento diretamente. A versão atual foi denominada Gen2, depois que os recursos do Armazenamento do Azure e do Data Lake Storage Gen1 foram combinados.

## Hadoop

O Hadoop foi criado pela Apache Software Foundation e é uma estrutura distribuída, escalonável e confiável para o processamento de big data, dividindo big data em blocos menores de dados e distribuindo-os em um cluster. Um cluster do Hadoop é composto por dois tipos de servidores: mestres e subordinados. O servidor mestre contém os componentes administrativos do Hadoop, enquanto os subordinados são aqueles em que o processamento de dados acontece. O Hadoop é responsável pelos dados lógicos de partição entre subordinados; os subordinados executam toda a transformação nos dados, reúnem insights e os repassam aos nós mestres que os agruparão para gerar a saída final. Ele pode escalar para milhares de servidores, com cada servidor fornecendo computação e armazenamento para os trabalhos. O Hadoop está disponível como serviço usando o serviço **HDIInsight** no Azure.

Há três componentes principais que compõem o sistema central do Hadoop:

**HDFS:** o Hadoop Distributed File System é um sistema de arquivos para o armazenamento de big data. Ele é uma estrutura distribuída que ajuda a dividir grandes arquivos de big data em blocos menores e colocá-los em diferentes subordinados em um cluster. O HDFS é um sistema de arquivos tolerante a falhas. Isso significa que, embora diferentes blocos de dados sejam disponibilizados a diferentes subordinados no cluster, há também a replicação de dados entre os subordinados para garantir que, em caso de falhas em qualquer subordinado, esses dados também sejam disponibilizados em outro servidor. Ele também fornece acesso rápido e eficiente aos dados para o solicitante.

**MapReduce:** o MapReduce é outra estrutura importante que permite ao Hadoop processar dados em paralelo. Essa estrutura é responsável pelo processamento de dados armazenados em subordinados do HDFS e pelo mapeamento deles para os subordinados. Depois que os subordinados concluírem o processamento, a parte "reduzir" trará informações de cada subordinado e as agrupará como a saída final. Geralmente, o HDFS e o MapReduce ficam disponíveis no mesmo nó, de modo que os dados não precisem viajar entre subordinados e uma maior eficiência possa ser alcançada ao processá-los.

**YARN: Yet Another Resource Negotiator (YARN)** é um importante componente arquitetônico do Hadoop que ajuda no agendamento de trabalhos relacionados a aplicações e gerenciamento de recursos em um cluster. O YARN foi lançado como parte do Hadoop 2.0, e muitas pessoas o escolheram como o sucessor do MapReduce, pois ele é mais eficiente em termos de processamento em lote e alocação de recursos.

## Apache Spark

O Apache Spark é uma solução analítica distribuída e confiável para processamento de dados em grande escala. Ele fornece um cluster capaz de executar trabalhos de transformação e machine learning em grandes quantidades de dados em paralelo e trazer um resultado consolidado para o cliente. Ele é composto por nós mestres e de trabalho. Os nós mestres são responsáveis por dividir e distribuir as ações em trabalhos e dados entre nós de trabalho, bem como consolidar os resultados de todos os nós de trabalho e devolver esses resultados ao cliente. Uma coisa importante a ser lembrada ao usar o Spark é que a lógica ou os cálculos devem ser facilmente parallelizados, e a quantidade de dados é muito grande para caber em uma máquina. O Spark está disponível no Azure como um serviço do HDInsight e do Databricks.

## Databricks

O Databricks é baseado no Apache Spark. Ele é uma Plataforma como Serviço em que um cluster gerenciado do Spark é disponibilizado para os usuários. Ele fornece muitos recursos adicionais, como um portal completo para gerenciar o cluster do Spark e seus nós, além de ajudar a criar blocos de anotações, agendar e executar trabalhos e fornecer segurança e suporte para vários usuários.

Agora, é hora de aprender a integrar dados de várias fontes e trabalhar com eles juntos usando as ferramentas sobre as quais temos falado.

## Integração de dados

Estamos cientes de como os padrões de integração são usados para aplicações. As aplicações compostas de vários serviços são integradas usando vários padrões. No entanto, há outro paradigma que é um requisito fundamental para muitas organizações, conhecido como integração de dados. O aumento na integração de dados aconteceu principalmente durante a última década, quando a geração e a disponibilidade de dados se tornaram incrivelmente altas. A velocidade, a variedade e o volume de dados que estão sendo gerados aumentaram drasticamente, e há dados em quase todos os lugares.

Cada organização tem muitos tipos diferentes de aplicações, e todos eles geram dados em seus próprios formatos. Muitas vezes, os dados também são adquiridos do mercado. Mesmo durante fusões e integrações de organizações, os dados precisam ser migrados e combinados.

A integração de dados refere-se ao processo de trazer dados de várias fontes e gerar uma nova saída que tenha mais significado e usabilidade.

Há uma necessidade absoluta de integração de dados nos seguintes cenários:

- Migrar dados de uma origem ou de um grupo de origens para um destino. Isso é necessário para disponibilizar dados em diferentes formatos para diferentes stakeholders e consumidores.
- Obter insights de dados. Com o rápido aumento da disponibilidade de dados, as organizações desejam obter insights deles. Elas desejam criar soluções que forneçam insights; dados de várias fontes devem ser mesclados, limpos, aumentados e armazenados em um data warehouse.
- Gerar painéis e relatórios em tempo real.
- Criar soluções de análise.

A integração de aplicações apresenta um comportamento de tempo de execução quando os usuários estão consumindo a aplicação; por exemplo, em caso de validação e integração de cartões de crédito. Por outro lado, a integração de dados acontece como um exercício de back-end e não está diretamente vinculada à atividade do usuário.

Vamos compreender o processo ETL com o Azure Data Factory.

## ETL

Um processo muito popular, conhecido como ETL, ajuda na criação de uma fonte de dados de destino para armazenar dados que são consumíveis por aplicações. Geralmente, os dados estão em um formato bruto e, para que se tornem consumíveis, devem passar pelas três fases distintas a seguir:

- **Extrair:** durante essa fase, os dados são extraídos de vários lugares. Por exemplo, pode haver várias fontes e todas precisam estar conectadas para recuperar os dados. As fases de extração normalmente usam conectores de dados que consistem em informações de conexão relacionadas à fonte de dados de destino. Elas também podem ter armazenamento temporário para trazer os dados da fonte e armazená-los para uma recuperação mais rápida. Essa fase é responsável pela ingestão de dados.
- **Transformar:** os dados que ficam disponíveis após a fase de extração podem não ser diretamente consumíveis por aplicações. Isso pode ocorrer por vários motivos; por exemplo, os dados podem ter irregularidades, pode haver dados ausentes ou pode haver dados errados. Ou pode até mesmo haver dados que não são necessários. Como alternativa, o formato dos dados pode não ser apropriado para consumo pelas aplicações de destino. Em todos esses casos, a transformação deve ser aplicada aos dados de forma que possa ser consumida com eficiência pelas aplicações.

- **Carregar:** após a transformação, os dados devem ser carregados na fonte de dados de destino em um formato e esquema que permitam uma disponibilidade mais rápida, fácil e centrada na performance para aplicações. Mais uma vez, isso geralmente consiste em conectores de dados para fontes de dados de destino e no carregamento de dados nelas.

A seguir, vamos abordar como o Azure Data Factory se relaciona ao processo ETL.

## Uma cartilha sobre o Azure Data Factory

O Azure Data Factory é uma ferramenta totalmente gerenciada, altamente disponível, altamente escalável e fácil de usar para criar soluções de integração e implementar fases de ETL. Ele ajuda você a criar novos pipelines por meio do recurso de arrastar e soltar usando uma interface de usuário, sem escrever código; no entanto, ele ainda fornece recursos para permitir que você escreva código na linguagem de sua preferência.

Há alguns conceitos importantes a serem conhecidos antes de usar o serviço Data Factory, que serão explorados em mais detalhes nas seções a seguir:

- **Atividades:** tarefas individuais que permitem a execução e o processamento da lógica em um pipeline do Data Factory. Há vários tipos de atividades. Há atividades relacionadas a movimentação de dados, transformação de dados e atividades de controle. Cada atividade tem uma política por meio da qual é possível decidir o mecanismo e o intervalo de repetição.
- **Pipelines:** no Data Factory, eles são compostos por grupos de atividades e são responsáveis por reunir as atividades. Os pipelines são os fluxos de trabalho e os orquestradores que permitem a execução das fases de ETL. Eles permitem a união de atividades e a declaração de dependências entre elas. Ao usar dependências, é possível executar algumas tarefas em paralelo e outras em sequência.
- **Conjuntos de dados:** são as origens e os destinos dos dados. Eles podem ser contas de armazenamento do Azure, Data Lake Storage ou uma série de outras origens.
- **Serviços vinculados:** serviços que contêm as informações de conexão e conectividade para os conjuntos de dados e são utilizados por tarefas individuais para se conectar a eles.
- **Tempo de execução de integração:** o mecanismo principal responsável pela execução do Data Factory é chamado de tempo de execução de integração. O tempo de execução de integração está disponível nas três configurações a seguir:
- **Azure:** nesta configuração, o Data Factory é executado nos recursos de computação fornecidos pelo Azure.

- **Auto-hospedada:** nesta configuração, o Data Factory é executado quando você traz seus próprios recursos de computação. Isso pode acontecer por meio de servidores de máquinas virtuais baseados na nuvem ou na infraestrutura local.
- **SQL do Azure Server Integration Services (SSIS):** esta configuração permite a execução de pacotes SSIS tradicionais gravados usando o SQL Server.
- **Versões:** o Data Factory é fornecido em duas versões diferentes. É importante entender que todos os novos desenvolvimentos ocorrerão na V2 e que a V1 permanecerá como está, ou desaparecerá em algum momento. A V2 é a versão preferencial pelos seguintes motivos:

Fornece a capacidade de executar pacotes de integração do SQL Server.

Tem funcionalidades aprimoradas em comparação com a V1.

É fornecida com monitoramento aprimorado, que não existe na V1.

Agora que você tem uma boa compreensão do Data Factory, vamos analisar as várias opções de armazenamento disponíveis no Azure.

## Uma cartilha sobre o Azure Data Lake Storage

O Azure Data Lake Storage fornece armazenamento para soluções de big data. Ele foi especialmente projetado para armazenar as grandes quantidades de dados que costumam ser necessárias em soluções de big data. Ele é um serviço gerenciado fornecido pelo Azure. Os clientes precisam trazer seus dados e armazená-los em um data lake.

Há duas versões do Azure Data Lake Storage: versão 1 (Gen1) e a versão atual, versão 2 (Gen2). A Gen2 tem todas as funcionalidades da Gen1, mas uma diferença particular é que ela é baseada no Armazenamento de Blobs do Azure.

Como o Armazenamento de Blobs do Azure é altamente disponível, pode ser replicado várias vezes, está preparado para desastres e tem baixo custo, esses benefícios são transferidos para o Data Lake Storage Gen2. O Data Lake Storage pode armazenar qualquer tipo de dados, incluindo dados relacionais, não relacionais, baseados em sistemas de arquivos e hierárquicos.

Criar uma instância do Data Lake Storage Gen2 é tão simples quanto criar uma nova conta de armazenamento. A única alteração que precisa ser feita é habilitar o namespace hierárquico na guia **Avançado** da sua conta de armazenamento. É importante observar que não há migração ou conversão direta de uma conta de armazenamento geral para o Azure Data Lake Storage ou vice-versa. Além disso, as contas de armazenamento são para armazenar arquivos, enquanto o Data Lake Storage é otimizado para leitura e ingestão de grandes quantidades de dados.

A seguir, investigaremos o processo e as principais fases ao trabalhar com big data. Essas são fases distintas e cada uma é responsável por diferentes atividades nos dados.

## Migrar dados do Armazenamento do Azure para o Data Lake Storage Gen2

Nesta seção, migraremos dados do Armazenamento de Blobs do Azure para outro contêiner do Azure da mesma instância do Armazenamento de Blobs do Azure e também migraremos dados para uma instância do Azure Data Lake Storage Gen2 usando um pipeline do Azure Data Factory. As seções a seguir detalham as etapas necessárias para criar essa solução completa.

### Preparar a conta de armazenamento de origem

Para podermos criar pipelines do Azure Data Factory e usá-los para migração, precisamos criar uma nova conta de armazenamento, que consiste em vários contêineres, e carregar os arquivos de dados. No mundo real, esses arquivos e a conexão de armazenamento já estariam preparados. A primeira etapa para criar uma nova conta de armazenamento do Azure é criar um novo grupo de recursos ou escolher um grupo de recursos existente em uma assinatura do Azure.

### Provisionar um novo grupo de recursos

Cada recurso no Azure é associado a um grupo de recursos. Antes de provisionarmos uma conta de armazenamento do Azure, precisamos criar um grupo de recursos que hospedará a conta de armazenamento. As etapas para a criação de um grupo de recursos são fornecidas aqui. Observe que um novo grupo de recursos pode ser criado durante o provisionamento de uma conta de armazenamento do Azure ou um grupo de recursos existente pode ser usado:

1. Navegue até o portal do Azure, faça logon e clique em **+ Criar um recurso**; em seguida, procure **Grupo de recursos**.
2. Selecione **Grupo de recursos** nos resultados da pesquisa e crie um novo grupo de recursos. Forneça um nome e escolha um local apropriado. Observe que todos os recursos devem ser hospedados no mesmo grupo de recursos e local para que seja fácil excluí-los.

Após o provisionamento do grupo de recursos, provisionaremos uma conta de armazenamento dentro dele.

## Provisionar uma conta de armazenamento

Nesta seção, veremos as etapas de criação de uma nova conta de armazenamento do Azure. Essa conta de armazenamento buscará a fonte de dados da qual os dados serão migrados. Execute as seguintes etapas para criar uma conta de armazenamento:

1. Clique em **+ Criar um recurso** e procure **Conta de Armazenamento**. Selecione **Conta de Armazenamento** nos resultados da pesquisa e crie uma nova conta de armazenamento.
2. Forneça um nome e local e, em seguida, selecione uma assinatura com base no grupo de recursos que foi criado anteriormente.
3. Selecione **StorageV2 (uso geral v2)** para **Tipo de conta**, **Padrão** para **Performance** e **Armazenamento com redundância local (LRS)** para **Replicação**, conforme demonstrado na Figura 9.2:

### Create storage account

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

#### PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	<input type="text" value="RiteshSubscription"/>
	<input type="button" value="▼"/>
	<input type="text" value="BigDataSolutions"/>
	<input type="button" value="▼"/>
<a href="#">Create new</a>	

#### INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name <small>?</small>	<input type="text" value="adfsamplesourcedata"/>	<input type="button" value="✓"/>
* Location	<input type="text" value="East US"/>	<input type="button" value="▼"/>
Performance <small>?</small>	<input checked="" type="radio"/> Standard <input type="radio"/> Premium	
Account kind <small>?</small>	<input type="text" value="StorageV2 (general purpose v2)"/>	<input type="button" value="▼"/>
Replication <small>?</small>	<input type="text" value="Locally-redundant storage (LRS)"/>	<input type="button" value="▼"/>
Access tier (default) <small>?</small>	<input type="radio"/> Cool <input checked="" type="radio"/> Hot	

Figura 9.2: configurando a conta de armazenamento

4. Agora, crie alguns contêineres na conta de armazenamento. O contêiner **rawdata** contém os arquivos que serão extraídos pelo pipeline do Data Factory e atuará como o conjunto de dados de origem, enquanto **finaldata** conterá arquivos nos quais os pipelines do Data Factory gravarão dados e atuará como o conjunto de dados de destino:

The screenshot shows the Azure Storage account interface for 'adfsamplesourcedata'. On the left, there's a sidebar with options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Events. The main area shows a list of containers under 'Storage account: adfsamplesourcedata'. There are two containers listed: 'finaldata' and 'rawdata'. The 'rawdata' container is highlighted with a blue dashed border.

Figura 9.3: criando contêineres

5. Carregue um arquivo de dados (esse arquivo está disponível com o código-fonte) no contêiner **rawdata**, como mostrado na Figura 9.4:

The screenshot shows the 'rawdata' container in the Azure Storage account. At the top, there are buttons for Upload, Refresh, Delete, and Acquire lease. Below that, it says 'Location: rawdata'. A search bar says 'Search blobs by prefix (case-sensitive)'. Underneath, there's a table with a single row labeled 'NAME'. The file 'products.csv' is listed in this row with its corresponding icon.

Figura 9.4: carregando um arquivo de dados

Após a conclusão dessas etapas, as atividades de preparação dos dados de origem estão concluídas. Agora podemos nos concentrar na criação de uma instância do Data Lake Storage.

## Provisionar o serviço Data Lake Storage Gen2

Como já sabemos, o serviço Data Lake Storage Gen2 é baseado na conta de armazenamento do Azure. Por esse motivo, criaremos uma nova conta de armazenamento da mesma forma que fizemos anteriormente, com a única diferença sendo a seleção de **Habilitado** para **Namespace hierárquico** na guia **Avançado** da nova conta de armazenamento do Azure. Isso criará o novo serviço Data Lake Storage Gen2:

### Create storage account

<b>Azure Files</b>		
Large file shares <small>(i)</small>	<input type="radio"/> Disabled	<input checked="" type="radio"/> Enabled
<small> ⓘ The current combination of storage account kind, performance, replication and location does not support large file shares.</small>		
<b>Data protection</b>		
Blob soft delete <small>(i)</small>	<input checked="" type="radio"/> Disabled	<input type="radio"/> Enabled
<small> ⓘ Data protection and hierarchical namespace cannot be enabled simultaneously.</small>		
File share soft delete <small>(i)</small>	<input checked="" type="radio"/> Disabled	<input type="radio"/> Enabled
<small> ⓘ Data protection and hierarchical namespace cannot be enabled simultaneously.</small>		
Versioning <small>(i)</small>	<input checked="" type="radio"/> Disabled	<input type="radio"/> Enabled
<small> ⓘ The current combination of subscription, storage account kind, performance, replication and location does not support versioning.</small>		
<b>Data Lake Storage Gen2</b>		
Hierarchical namespace <small>(i)</small>	<input type="radio"/> Disabled	<input checked="" type="radio"/> Enabled
NFS v3 <small>(i)</small>	<input checked="" type="radio"/> Disabled	<input type="radio"/> Enabled
<small> ⓘ Sign up is currently required to utilize the NFS v3 feature on a per-subscription basis. <a href="#">Sign up for NFS v3</a></small>		

Figura 9.5: criando uma nova conta de armazenamento

Após a criação do data lake, vamos nos concentrar na criação de um novo pipeline do Data Factory.

## Provisionar o Azure Data Factory

Agora que provisionamos o grupo de recursos e a conta de armazenamento do Azure, é hora de criar um novo recurso do Data Factory:

1. Crie um novo pipeline do Data Factory selecionando **V2** e fornecendo um nome e local juntamente com uma seleção de grupo de recursos e assinatura.

O Data Factory tem três versões diferentes, conforme mostrado na Figura 9.6.  
Já discutimos a **V1** e a **V2**:

New data factory

Name \*

Version ⓘ

V2

Subscription \*

Microsoft Azure Sponsorship

Resource Group \*

akscluster

Create new

Location \*

West Central US

Enable GIT ⓘ

GIT URL \* ⓘ

Repo name \* ⓘ

testrepo

Branch Name \* ⓘ

master

Root folder \* ⓘ

Pipelines

Figura 9.6: selecionando a versão do Data Factory

- Depois que o recurso do Data Factory for criado, clique no link **Autoria e Monitoramento** no painel central.

Com isso, outra janela será aberta, exibindo o designer do Data Factory para os pipelines.

O código dos pipelines pode ser armazenado em repositórios de controle de versão para que ele possa ser rastreado para alterações de código e promover a colaboração entre desenvolvedores. Se você perdeu a definição das configurações do repositório nessas etapas, isso pode ser feito mais tarde.

A próxima seção se concentrará na configuração relacionada às configurações do repositório de controle de versão se o recurso do Data Factory tiver sido criado sem a definição de nenhuma configuração do repositório.

## Configurações do repositório

Antes de criar qualquer artefato do Data Factory, como conjuntos de dados e pipelines, é recomendável configurar o repositório de código para hospedar arquivos relacionados ao Data Factory:

- Na página **Criação**, clique no botão **Gerenciar** e, em seguida, **Configuração do Git** no menu à esquerda. Isso abrirá outro painel; clique no botão **Configurar repositório de código** nesse painel:

The screenshot shows the Microsoft Azure Data Factory configuration interface. The top navigation bar includes 'Data Factory', 'Publish all', 'Validate all', 'Refresh', 'Discard all', and 'ARM template'. The left sidebar lists 'Connections', 'Linked services', 'Integration runtimes', 'Source control' (which is expanded), 'Git configuration' (selected and highlighted with a blue border), 'Parameterization template', 'Author', 'Triggers', 'Security', and 'Customer managed key'. The main content area is titled 'Git repository' and contains a 'CI/CD best practices' link. It features a large icon of a document with a plus sign and the text 'No Git repository configured'. Below this, it says 'Connect to a repository for source control and collaboration for work on your data factory pipelines.' A blue button at the bottom right says 'Set up code repository'.

Figura 9.7: configurando um repositório Git

2. Na folha resultante, selecione qualquer um dos tipos de repositórios em que você deseja armazenar os arquivos de código do Data Factory. Neste caso, vamos selecionar o **Git do Azure DevOps**:

**Repository settings**

**Repository type \***

Azure DevOps Git

Select a different directory

**Azure Active Directory \***

[REDACTED]

**Azure DevOps Account \***

[REDACTED]

**Project name \***

[REDACTED]

**Git repository name \***  Create new  Use existing  [REDACTED]

**Collaboration branch \***

[REDACTED]

**Root folder \***

/

Import existing Data Factory resources to repository

**Branch to import resources into \***  Use Collaboration  Create new  Use existing

Figura 9.8: selecionando o tipo de repositório Git apropriado

3. Crie um novo repositório ou reutilize um repositório existente do Azure DevOps. Você já deve ter uma conta no Azure DevOps. Se não, acesse <https://dev.azure.com> e use a mesma conta utilizada para o portal do Azure para fazer logon e criar uma nova organização e projeto dentro dela. Consulte o Capítulo 13, *Integrar o Azure DevOps*, para saber mais sobre como criar organizações e projetos no Azure DevOps.

Agora, podemos voltar para a janela de criação do Data Factory e começar a criar artefatos para o nosso novo pipeline.

Na próxima seção, preparamos os conjuntos de dados que serão usados nos nossos pipelines do Data Factory.

## Conjuntos de dados do Data Factory

Agora podemos voltar ao pipeline do Data Factory. Primeiro, crie um novo conjunto de dados que atuará como o conjunto de dados de origem. Ele será a primeira conta de armazenamento para a qual vamos criar e carregar o arquivo de exemplo **product.csv**:

1. Clique em **+ Conjuntos de dados** -> **Novo Conjunto de Dados** no menu à esquerda e selecione **Armazenamento de Blobs do Azure** como o repositório de dados e **delimitedText** como o formato do arquivo de origem. Crie um novo serviço vinculado, fornecendo um nome e selecionando uma conta de armazenamento e assinatura do Azure. Por padrão, **AutoResolveIntegrationRuntime** é usado para o ambiente de tempo de execução, o que significa que o Azure fornecerá o ambiente de tempo de execução na computação gerenciada por ele. Os serviços vinculados fornecem vários métodos de autenticação, e nós estamos usando o método **URL (Uniform Resource Locator) de assinatura de acesso compartilhado (SAS)**. Também é possível usar uma chave de conta, uma entidade de serviço e uma identidade gerenciada como métodos de autenticação:

**New linked service (Azure Blob Storage)**

<b>Name *</b>	AzureBlobStorage1
<b>Description</b>	
<b>Connect via integration runtime *</b>	AutoResolveIntegrationRuntime
<b>Authentication method</b>	Account key
<input checked="" type="radio"/> Connection string <input type="radio"/> Azure Key Vault	
<b>Account selection method</b> <input checked="" type="radio"/> From Azure subscription <input type="radio"/> Enter manually	
<b>Azure subscription</b> <input type="checkbox"/> Select all	
<b>Storage account name *</b> <input type="text"/>	
<b>Additional connection properties</b> <input type="button" value="New"/>	
<b>Test connection</b> <input checked="" type="radio"/> To linked service <input type="radio"/> To file path	

Figura 9.9: implementando o método de autenticação

2. Em seguida, no painel inferior resultante, na guia **Geral**, clique no link **Abrir propriedades** e forneça um nome para o conjunto de dados:

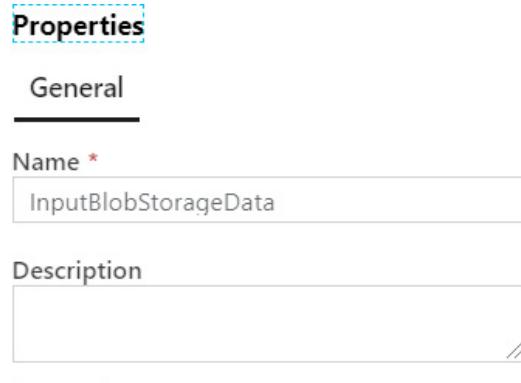


Figura 9.10: nomeando o conjunto de dados

3. Na guia **Conexão**, forneça detalhes sobre o contêiner, o nome do arquivo blob na conta de armazenamento, o delimitador de linhas, o delimitador de colunas e outras informações que ajudarão o Data Factory a ler os dados de origem adequadamente.

A guia **Conexão**, após a configuração, deve ser semelhante à Figura 9.11. Observe que o caminho inclui o nome do contêiner e o nome do arquivo:

Setting	Value
Linked service *	AzureBlobStorage1
File path *	rawdata / Directory / products.csv
Compression type	none
Column delimiter	Comma (,)
Row delimiter	Auto detect (\r,\n, or \r\n)
Encoding	Default(UTF-8)
Escape character	Backslash (\)
Quote character	Double quote (")

Figura 9.11: configurando a conexão

- Neste momento, se você clicar no botão **Dados de visualização**, ele mostrará os dados de visualização do arquivo **product.csv**. Na guia **Esquema**, adicione duas colunas e nomeie-as como **ProductID** e **ProductPrice**. O esquema ajuda a fornecer um identificador para as colunas e também a mapear as colunas de origem no conjunto de dados de origem para as colunas de destino no conjunto de dados de destino quando os nomes não são os mesmos.

Agora que o primeiro conjunto de dados foi criado, vamos criar o segundo.

### Criar o segundo conjunto de dados

Crie um novo conjunto de dados e serviço vinculado para a conta de armazenamento de blobs de destino da mesma maneira que você fez antes. Observe que a conta de armazenamento é igual à origem, mas o contêiner é diferente. Verifique se os dados de entrada têm informações de esquema associadas a eles, como mostrado na Figura 9.12:

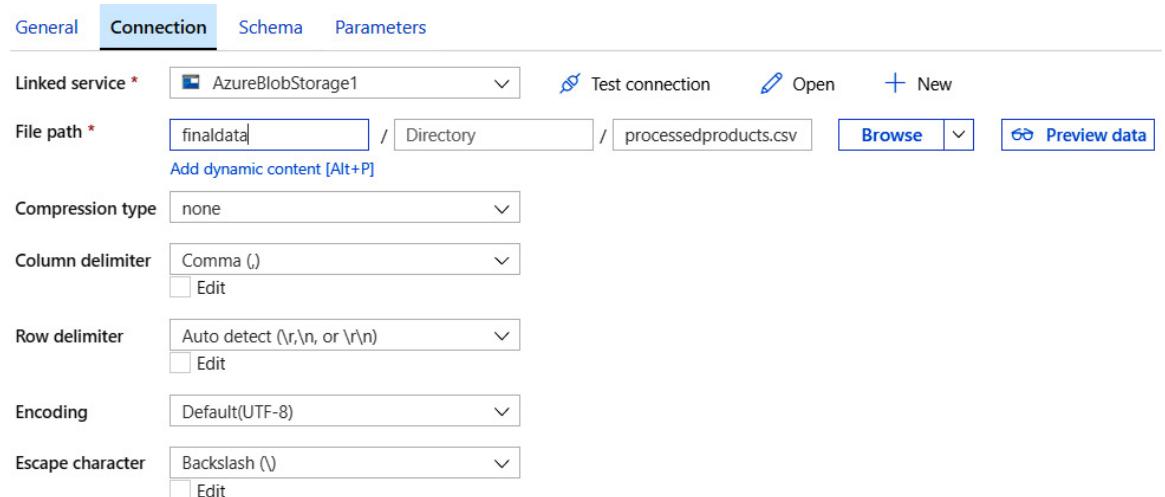


Figura 9.12: criando o segundo conjunto de dados

A seguir, criaremos um terceiro conjunto de dados.

### Criar um terceiro conjunto de dados

Crie um novo conjunto de dados para a instância de armazenamento do Data Lake Storage Gen2 como o conjunto de dados de destino. Para fazer isso, selecione o novo conjunto de dados e, em seguida, selecione **Azure Data Lake Storage Gen2 (Versão Prévia)**.

Dê um nome ao novo conjunto de dados e crie um novo serviço vinculado na guia **Conexão**. Escolha **Usar chave de conta** como o método de autenticação e o restante da configuração será preenchida automaticamente após a seleção do nome da conta de armazenamento. Em seguida, teste a conexão clicando no botão **Testar conexão**. Mantenha a configuração padrão para o restante das guias, conforme mostrado na Figura 9.13:

## New Linked Service (Azure Data Lake Storage Gen2 ... X

Name \*

Description

Connect via integration runtime \*

▼

Authentication method

▼

Account selection method

From Azure subscription ● Enter manually i

Azure subscription

▼

Storage account name \*

▼

[Sign up to the public preview of Azure Data Lake Storage Gen2.](#)

Annotations

+ New

► Advanced i

Cancel Test connection Finish

Figura 9.13: configuração nas guias Conexão

Agora que temos a conexão com os dados de origem e também conexões com os repositórios de dados de origem e de destino, é hora de criar os pipelines que conterão a lógica da transformação de dados.

## Criar um pipeline

Depois que todos os conjuntos de dados forem criados, poderemos criar um pipeline que consumirá esses conjuntos de dados. As etapas para criar um pipeline são fornecidas a seguir:

1. Clique no menu **+ Pipelines => Novo Pipeline** do menu à esquerda para criar um novo pipeline. Em seguida, arraste e solte a atividade **Copiar Dados** do menu **Mover e Transformar**, conforme demonstrado na Figura 9.14:

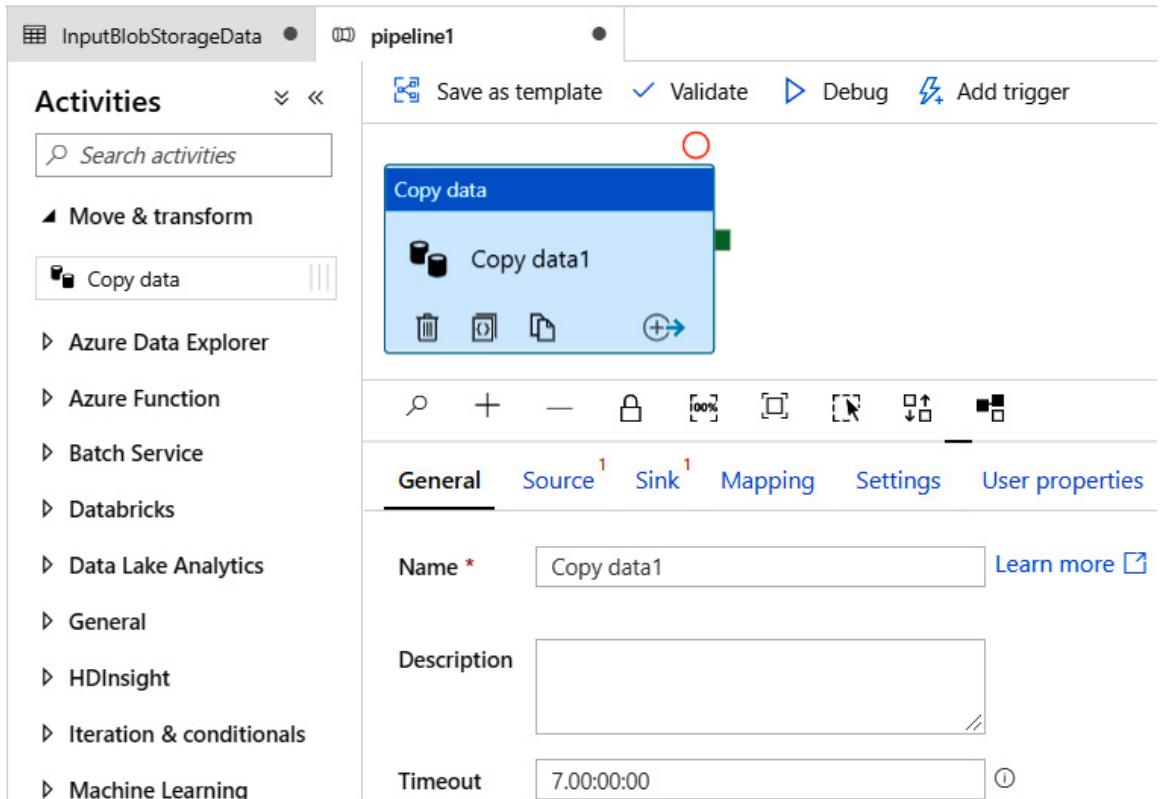


Figura 9.14: menu do pipeline

2. A guia **Geral** resultante pode ser deixada como está, mas a guia **Origem** deve ser configurada para usar o conjunto de dados de origem que configuramos anteriormente:

Figura 9.15: guia Origem

3. A guia **Coletor** é usada para configurar o conjunto de dados e o repositório de dados de destino e deve ser configurada para usar o conjunto de dados de destino que configuramos anteriormente:

Figura 9.16: guia Coletor

4. Na guia **Mapeamento**, mapeie as colunas da origem para as colunas do conjunto de dados de destino, conforme mostrado na Figura 9.17:

Field / Type	Field	Type	Include
ProductID(String)	ProductId	String	<input checked="" type="checkbox"/>
ProductPrice(Decimal)	ProductPrice	Decimal	<input checked="" type="checkbox"/>

Figura 9.17: guia Mapeamento

## Adicionar mais uma atividade Copiar Dados

No nosso pipeline, podemos adicionar várias atividades, cada uma responsável por uma tarefa de transformação específica. A tarefa analisada nesta seção é responsável por copiar dados da conta de armazenamento do Azure para o Azure Data Lake Storage:

1. Adicione outra atividade **Copiar Dados** no menu de atividades à esquerda para migrar dados para o Data Lake Storage. As duas tarefas de cópia serão executadas em paralelo:

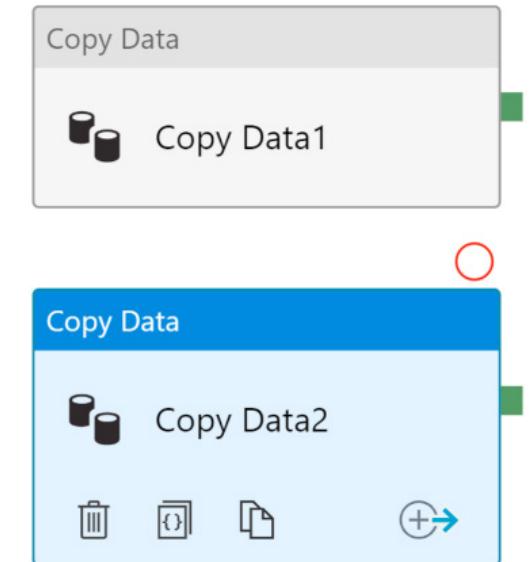


Figura 9.18: atividades Copiar Dados

A configuração para a origem é a conta de Armazenamento de Blobs do Azure que contém o arquivo **product.csv**.

A configuração do coletor escolherá a conta de armazenamento do Data Lake Storage Gen2.

2. O restante da configuração pode ser deixado nas configurações padrão para a segunda atividade Copiar Dados.

Depois que a criação do pipeline for concluída, ele poderá ser publicado em um repositório de controle de versão, como o GitHub.

A seguir, investigaremos a criação de uma solução usando o Databricks e o Spark.

## Criar uma solução usando o Databricks

O Databricks é uma plataforma para usar o Spark como serviço. Não precisamos provisionar nós mestres e de trabalho em máquinas virtuais. Em vez disso, o Databricks nos fornece um ambiente gerenciado que consiste em nós mestres e de trabalho e também os gerencia. Precisamos fornecer as etapas e a lógica para o processamento de dados, e o restante é resolvido pela plataforma do Databricks.

Nesta seção, veremos as etapas de criação de uma solução usando o Databricks. Vamos fazer o download de dados de exemplo para analisar.

O download do CSV de exemplo foi feito em <https://ourworldindata.org/coronavirus-source-data>, embora ele também seja disponibilizado com o código deste livro. A URL mencionada anteriormente terá dados mais atualizados; no entanto, o formato pode ter mudado e, portanto, é recomendável usar o arquivo disponível com os exemplos de código deste livro:

1. A primeira etapa na criação de uma solução do Databricks é provisioná-la no portal do Azure. Há um SKU de avaliação de 14 dias disponível juntamente com dois outros SKUs: padrão e premium. O SKU premium tem o Controle de Acesso Baseado em Função do Azure no nível de blocos de anotações, clusters, trabalhos e tabelas:

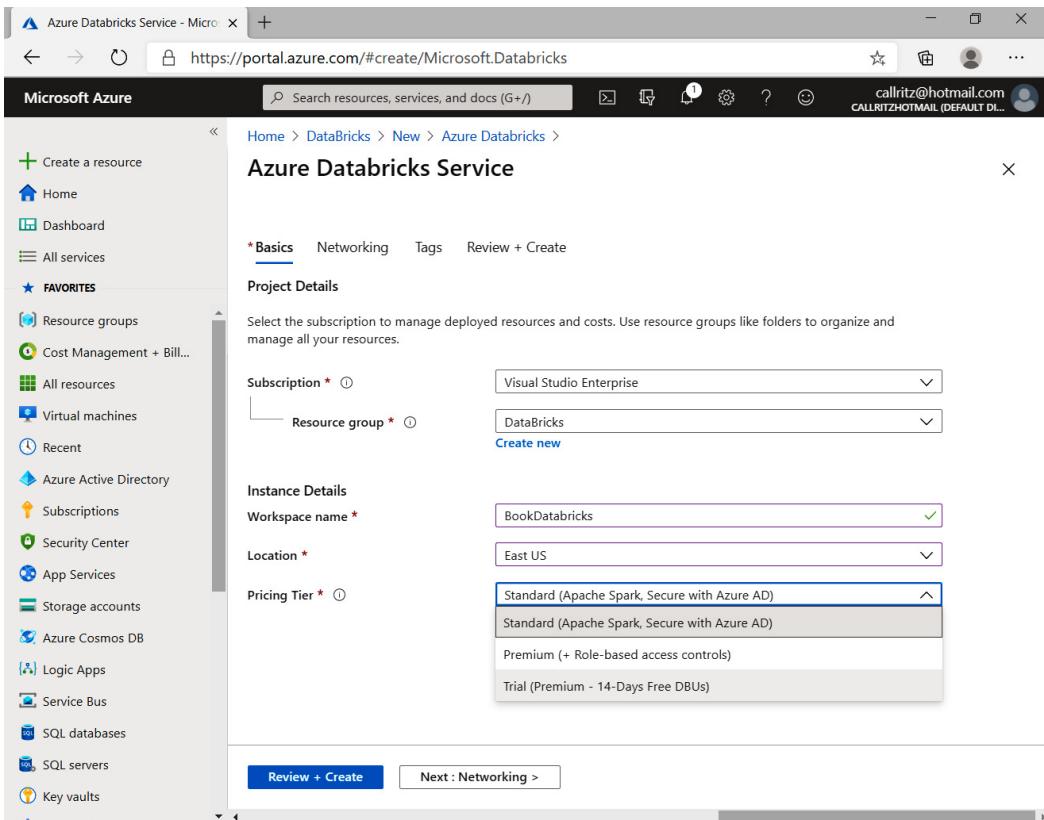


Figura 9.19: portal do Azure – serviço do Databricks

2. Depois que o espaço de trabalho do Databricks for provisionado, clique no botão **Iniciar o workspace** do painel **Visão geral**. Isso abrirá uma nova janela do navegador e, eventualmente, conectará você ao portal do Databricks.
3. No portal do Databricks, selecione **Clusters** no menu à esquerda e crie um novo cluster, conforme mostrado na Figura 9.20:

Create Cluster

New Cluster

[Cancel](#) [Create Cluster](#)

**2-8 Workers:** 28.0-112.0 GB Memory, 8-32 Cores, 1.5-6 DBU  
**1 Driver:** 14.0 GB Memory, 4 Cores, 0.75 DBU

---

Cluster Name

Cluster Mode

Pool

Databricks Runtime Version [Learn more](#)

**New** This Runtime version supports only Python 3.

Autopilot Options  
 Enable autoscaling  
 Terminate after  minutes of inactivity

Worker Type	Min Workers	Max Workers
<input type="text" value="Standard_DS3_v2"/> 14.0 GB Memory, 4 Cores, 0.75 DBU	<input type="text" value="2"/>	<input type="text" value="8"/>

Driver Type  
 14.0 GB Memory, 4 Cores, 0.75 DBU

► Advanced Options

Figura 9.20: criando um novo cluster

4. Forneça o nome, a versão do tempo de execução do Databricks, o número de tipos de trabalho, a configuração de tamanho da máquina virtual e a configuração do servidor do tipo driver.
5. A criação do cluster pode levar alguns minutos. Após a criação do cluster, clique em **Página Inicial**, selecione um usuário do menu de contexto e crie um novo bloco de anotações:

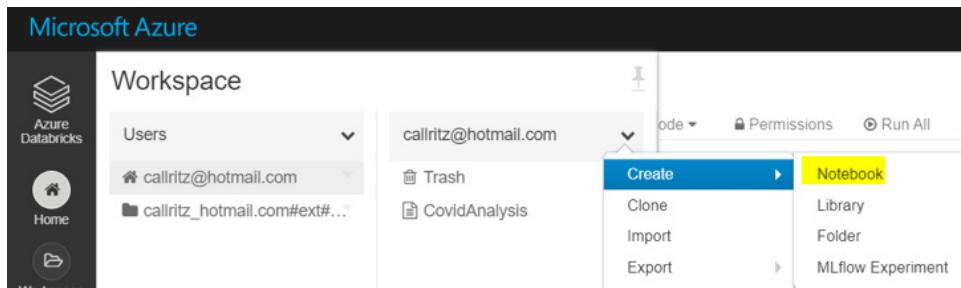


Figura 9.21: selecionando um novo bloco de anotações

- Forneça um nome ao bloco de anotações, como mostrado a seguir:

### Create Notebook

The 'Create Notebook' dialog box contains the following fields:

- Name: CovidAnalysis
- Default Language: Python
- Cluster: BookBigDataCluster

At the bottom are 'Cancel' and 'Create' buttons, with 'Create' being highlighted in blue.

Figura 9.22: criando um bloco de anotações

- Crie uma nova conta de armazenamento, como mostrado a seguir. Isso atuará como armazenamento para os dados brutos da COVID no formato CSV:

### Create storage account

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

The 'Create storage account' dialog box includes the following sections:

- Subscription \***: Visual Studio Enterprise
- Resource group \***: DataBricks (with a 'Create new' link)
- Instance details**:
  - The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model.
- Storage account name \***: coronadastorage
- Location \***: East US
- Performance**: Standard (selected)
- Account kind**: StorageV2 (general purpose v2)
- Replication**: Locally-redundant storage (LRS)
- Access tier (default)**: Hot (selected)

Figura 9.23: criando uma nova conta de armazenamento

- Crie um contêiner para armazenar o arquivo CSV, como mostrado a seguir:

Figura 9.24: criando um contêiner

- Carregue o arquivo `owid-covid-data.csv` nesse contêiner.

Depois de concluir as etapas anteriores, a próxima tarefa é carregar os dados.

## Carregar dados

A segunda etapa importante é carregar os dados da COVID no espaço de trabalho do Databricks. Isso pode ser feito de duas maneiras principais:

- Montar o contêiner de armazenamento do Azure no Databricks e, em seguida, carregar os arquivos disponíveis na montagem.
- Carregar os dados diretamente da conta de armazenamento. Esta abordagem foi usada no exemplo a seguir.

As etapas a seguir devem ser executadas para carregar e analisar dados usando o Databricks:

- A primeira etapa é conectar e acessar a conta de armazenamento. A chave para a conta de armazenamento é necessária e está armazenada na configuração do Spark. Observe que a chave aqui é "`fs.azure.account.key.coronadatastorage.blob.core.windows.net`" e o valor é a chave associada:

```
spark.conf.set("fs.azure.account.key.coronadatastorage.blob.core.windows.net", "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=xxx")
```

- A chave para a conta de armazenamento do Azure pode ser recuperada navegando até as configurações e a propriedade **Chaves de Acesso** da conta de armazenamento no portal.

A próxima etapa é carregar o arquivo e ler os dados no arquivo CSV. O esquema deve ser inferido do próprio arquivo, em vez de ser fornecido explicitamente. Há também uma linha de cabeçalho, que é representada usando a opção no próximo comando.

O arquivo é referenciado usando o seguinte formato: `wasbs://{{container}}@{{storage account name}}.blob.core.windows.net/{{filename}}`.

3. O método **read** do objeto **SparkSession** fornece métodos para ler arquivos. Para ler arquivos CSV, o método **csv** deve ser usado juntamente com os parâmetros necessários, como o caminho para o arquivo CSV. Há parâmetros opcionais adicionais que podem ser fornecidos para personalizar o processo de leitura dos arquivos de dados. Há vários tipos de formatos de arquivo, como JSON, **Optimized Row Columnar (ORC)** e Parquet, e bancos de dados relacionais, como SQL Server e MySQL, repositórios de dados NoSQL, como Cassandra e MongoDB, e plataformas de big data, como Apache Hive, que podem ser usados no Spark. Vamos dar uma olhada no seguinte comando para entender a implementação dos DataFrames do Spark:

```
coviddata = spark.read.format("csv").option("inferSchema", "true").
option("header", "true").load("wasbs://coviddata@coronadatastorage.blob.
core.windows.net/owid-covid-data.csv")
```

O uso desse comando cria um novo objeto do tipo DataFrame no Spark. O Spark fornece objetos **Resilient Distributed Dataset (RDD)** para manipular e trabalhar com dados. Os RDDs são objetos de baixo nível e qualquer código escrito para trabalhar com eles pode não ser otimizado. Os DataFrames são componentes de nível superior aos RDDs e fornecem otimização para acessar e trabalhar com eles. É melhor trabalhar com DataFrames do que com RDDs.

Os DataFrames fornecem dados no formato de linha/coluna, o que facilita a visualização e o trabalho com dados. Os DataFrames do Spark são semelhantes aos DataFrames do Pandas, com a diferença de serem implementações distintas.

4. O comando a seguir mostra os dados em um DataFrame. Ele mostra todas as linhas e colunas disponíveis no DataFrame:

```
coviddata.show()
```

Você obterá uma saída semelhante à da Figura 9.25:

iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million	total_deaths_per_million	new_deaths_per_million	total_tests	new_tests	tests_per_thousand	new_tests_per_thousand	new_tests_smoothed	new_tests_smoothed_per_thousand	tests_units	stringency_index	population	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	extreme_poverty	cvd_death_rate	diabetes_prevalence	female_smokers	male_smokers	handwashing_facilities	hospital_beds_per_100k	+
null	null	2020-03-13 00:00:00	2	2	0	0	18.733	18.733	18.733	0.0	106766.0	584.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
41.2	ABW	Aruba 2020-03-13 00:00:00	13.085	7.452	35973.781	0	null	null	11.62	11.62	11.62	0.0	37.465	37.465	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733	18.733		
41.2	ABW	Aruba 2020-03-28 00:00:00	13.085	7.452	35973.781	4	2	0	0	11.62	11.62	11.62	0.0	30.56	30.56	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	106766.0	
41.2	ABW	Aruba 2020-03-28 00:00:00	13.085	7.452	35973.781	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figura 9.25: os dados brutos em um DataFrame

5. O esquema dos dados carregados é inferido pelo Spark e pode ser verificado usando o seguinte comando:

```
coviddata.printSchema()
```

Isso fornecerá uma saída semelhante a esta:

```

root
|-- iso_code: string (nullable = true)
|-- location: string (nullable = true)
|-- date: timestamp (nullable = true)
|-- total_cases: integer (nullable = true)
|-- new_cases: integer (nullable = true)
|-- total_deaths: integer (nullable = true)
|-- new_deaths: integer (nullable = true)
|-- total_cases_per_million: double (nullable = true)
|-- new_cases_per_million: double (nullable = true)
|-- total_deaths_per_million: double (nullable = true)
|-- new_deaths_per_million: double (nullable = true)

```

Figura 9.26: obtendo o esquema do DataFrame para cada coluna

6. Para contar o número de linhas no arquivo CSV, o comando a seguir pode ser usado, e sua saída mostra que há 19.288 linhas no arquivo:

```
coviddata.count()
```

▶ (1) Spark Jobs

Out[7]: 19288

Figura 9.27: encontrando a contagem de registros em um DataFrame

7. O DataFrame original tem mais de 30 colunas. Também podemos selecionar um subconjunto das colunas disponíveis e trabalhar com elas diretamente, como mostrado a seguir:

```
CovidDataSmallSet = coviddata.select("location", "date", "new_cases",
"new_deaths")
CovidDataSmallSet.show()
```

A saída do código será como mostrado na Figura 9.28:

location	date	new_cases	new_deaths
Aruba	2020-03-13 00:00:00	2	0
Aruba	2020-03-20 00:00:00	2	0
Aruba	2020-03-24 00:00:00	8	0
Aruba	2020-03-25 00:00:00	5	0
Aruba	2020-03-26 00:00:00	2	0
Aruba	2020-03-27 00:00:00	9	0
Aruba	2020-03-28 00:00:00	0	0
Aruba	2020-03-29 00:00:00	0	0
Aruba	2020-03-30 00:00:00	22	0
Aruba	2020-04-01 00:00:00	5	0
Aruba	2020-04-02 00:00:00	0	0

Figura 9.28: selecionando algumas colunas das colunas gerais

8. É possível filtrar dados usando o método **filter**, como mostrado a seguir:

```
CovidDataSmallSet.filter(" location == 'United States' ").show()
```

9. Também é possível adicionar várias condições em conjunto usando os operadores AND (&) ou OR (|):

```
CovidDataSmallSet.filter((CovidDataSmallSet.location == 'United States') | (CovidDataSmallSet.location == 'Aruba')).show()
```

10. Para descobrir o número de linhas e outros detalhes estatísticos, como o desvio médio, máximo, mínimo e padrão, o método **describe** pode ser usado:

```
CovidDataSmallSet.describe().show()
```

Ao usar o comando anterior, você obterá uma saída semelhante a esta:

summary	location	new_cases	new_deaths
count	19288	19288	19288
mean	null	536.6524263790958	35.05174201576109
stddev	null	4828.611755359697	335.3488977234115
min	Afghanistan	-2461	0
max	Zimbabwe	107909	10520

Figura 9.29: mostrando as estatísticas de cada coluna usando o método **describe**

11. Também é possível descobrir a porcentagem de dados nulos ou vazios em colunas especificadas. Alguns exemplos são mostrados a seguir:

```
from pyspark.sql.functions import col
(coviddata.where(col("diabetes_prevalence").isNull()).count() * 100) /
coviddata.count()
```

A saída mostra **5,998548320199087**, o que significa que 95% dos dados são nulos. Devemos remover essas colunas da análise de dados. Da mesma forma, a execução do mesmo comando na coluna **total\_tests\_per\_thousand** retorna **73,62090418913314**, o que é muito melhor do que a coluna anterior.

12. Para descartar algumas das colunas do DataFrame, o próximo comando pode ser usado:

```
coviddatanew=coviddata.drop("iso_code").drop("total_tests").drop("total_tests").drop("new_tests").drop("total_tests_per_thousand").drop("new_tests_per_thousand").drop("new_tests_smoothed").drop("new_tests_smoothed_per_thousand")
```

13. Às vezes, você precisará ter uma agregação de dados. Nesses cenários, você pode executar o agrupamento de dados, como mostrado aqui:

```
coviddatanew = coviddata.groupBy('location').agg({'date': 'max'})
```

Isso exibirá os dados da instrução **groupBy**:

location	max(date)
Chad	2020-05-23 00:00:00
Anguilla	2020-05-23 00:00:00
Paraguay	2020-05-23 00:00:00
Russia	2020-05-23 00:00:00
International	2020-03-10 00:00:00
Yemen	2020-05-23 00:00:00
World	2020-05-23 00:00:00
Senegal	2020-05-23 00:00:00
Sweden	2020-05-23 00:00:00
Guyana	2020-05-23 00:00:00
Eritrea	2020-05-23 00:00:00
Jersey	2020-05-23 00:00:00
Philippines	2020-05-23 00:00:00

Figura 9.30: dados da instrução groupby

14. Como você pode ver na coluna **max (date)**, as datas são praticamente as mesmas para todos os países, podemos usar esse valor para filtrar os registros e obter uma única linha para cada país que representa a data máxima:

```
coviddatauniquecountry = coviddata.filter("date='2020-05-23 00:00:00'")  
coviddatauniquecountry.show()
```

15. Se pegarmos uma contagem de registros para o novo DataFrame, obtemos **209**.

Podemos salvar o novo DataFrame em outro arquivo CSV, que pode ser necessário para outros processadores de dados:

```
coviddatauniquecountry.rdd.saveAsTextFile("dbfs:/mnt/coronadatastorage/  
uniquecountry.csv")
```

Podemos verificar o arquivo recém-criado com o seguinte comando:

```
%fs ls /mnt/coronadatastorage/
```

O caminho montado será exibido conforme mostrado na Figura 9.31:



Figura 9.31: o caminho montado nos nós do Spark

16. Também é possível adicionar os dados ao catálogo do Databricks usando o método `createTempView` ou `createOrReplaceTempView` no catálogo do Databricks. Colocar dados no catálogo disponibiliza-os em um determinado contexto. Para adicionar dados ao catálogo, o método `createTempView` ou `createOrReplaceTempView` do DataFrame pode ser usado, fornecendo uma nova exibição para a tabela no catálogo:

```
coviddatauniquecountry.createOrReplaceTempView("corona")
```

17. Quando a tabela estiver no catálogo, ela será acessada da sua sessão do SQL, conforme mostrado a seguir:

```
spark.sql("select * from corona").show()
```

Os dados da instrução SQL serão exibidos conforme mostrado na Figura 9.32:

location	date	total_cases
Aruba	2020-05-23 00:00:00	101
Afghanistan	2020-05-23 00:00:00	9216
Angola	2020-05-23 00:00:00	60
Anguilla	2020-05-23 00:00:00	3
Albania	2020-05-23 00:00:00	981
Andorra	2020-05-23 00:00:00	762
United Arab Emirates	2020-05-23 00:00:00	27892
Argentina	2020-05-23 00:00:00	10636
Armenia	2020-05-23 00:00:00	5928
Antigua and Barbuda	2020-05-23 00:00:00	25
Australia	2020-05-23 00:00:00	7095
Austria	2020-05-23 00:00:00	16361
Azerbaijan	2020-05-23 00:00:00	3855
Burundi	2020-05-23 00:00:00	42
Belgium	2020-05-23 00:00:00	56511
Benin	2020-05-23 00:00:00	135
Bonaire Sint Eust...	2020-05-23 00:00:00	6
Burkina Faso	2020-05-23 00:00:00	814
Bangladesh	2020-05-23 00:00:00	30205
Bulgaria	2020-05-23 00:00:00	2408

Figura 9.32: dados da instrução SQL

- 
18. É possível executar uma consulta SQL adicional em relação à tabela, como mostrado a seguir:

```
spark.sql("select * from corona where location in ('India','Angola') order by location").show()
```

Esse foi um pequeno resumo das possibilidades com o Databricks. Há muito mais recursos e serviços dentro dele que não poderiam ser abordados em um único capítulo. Leia mais sobre ele em <https://azure.microsoft.com/services/databricks>.

## Resumo

Este capítulo tratou do serviço Azure Data Factory, que é responsável por fornecer serviços ETL no Azure. Por ser uma plataforma como serviço, ele fornece escalabilidade ilimitada, alta disponibilidade e pipelines fáceis de configurar. Sua integração com o Azure DevOps e o GitHub também é perfeita. Também exploramos os recursos e benefícios do uso do Azure Data Lake Storage Gen2 para armazenar qualquer tipo de big data. É um repositório de dados hierárquico, econômico e altamente escalonável para lidar com big data e é compatível com o Azure HDInsight, o Databricks e o ecossistema Hadoop.

De forma alguma houve um aprofundamento completo em todos os tópicos mencionados neste capítulo. O objetivo foi falar sobre as possibilidades no Azure, especialmente com o Databricks e o Spark. Há várias tecnologias no Azure relacionadas a big data, incluindo o HDInsight, o Hadoop, o Spark e seu ecossistema relacionado, e o Databricks, que é um ambiente de Plataforma como Serviço para o Spark com funcionalidade adicionada. No próximo capítulo, você conhecerá os recursos de computação sem servidor no Azure.



# 10

## Sem servidor no Azure – Trabalhar com o Azure Functions

No capítulo anterior, você aprendeu sobre várias soluções de big data disponíveis no Azure. Neste capítulo, você aprenderá como a tecnologia sem servidor pode ajudá-lo a lidar com uma grande quantidade de dados.

Sem servidor é um dos chavões mais usados em tecnologia hoje em dia, e todo mundo quer pegar uma carona. Sem servidor traz muitas vantagens em computação em geral, processos de desenvolvimento de software, infraestrutura e implementação técnica. Há muita coisa acontecendo na indústria: em uma extremidade do espectro está a **Infraestrutura como Serviço (IaaS)** e, na outra, a função sem servidor. Entre as duas estão a **Plataforma como Serviço (PaaS)** e os contêineres. Conheci muitos desenvolvedores, e parece que há uma certa confusão entre eles sobre IaaS, PaaS, contêineres e computação sem servidor. Além disso, há muita confusão sobre casos de uso, aplicabilidade, arquitetura e implementação para o paradigma sem servidor. A função em servidor é um novo paradigma que está mudando não apenas a tecnologia, bem como a cultura e os processos dentro das organizações.

Iniciaremos este capítulo apresentando a tecnologia sem servidor e abordaremos os seguintes tópicos à medida que avançamos:

- Funções como Serviço
- Azure Functions
- Azure Durable Functions
- Grade de Eventos do Azure

## Sem servidor

A função sem servidor refere-se a um modelo de implantação na qual os usuários são responsáveis somente pelo código e pela configuração da sua aplicação.

Na computação sem servidor, os clientes não precisam se preocupar com trazer a sua própria plataforma subjacente e a infraestrutura e, em vez disso, podem se concentrar na resolução dos problemas de negócios.

Sem servidor não significa que não existem servidores. O código e a configuração sempre precisarão de computação, armazenamento e redes para serem executados. No entanto, do ponto de vista do cliente, não há visibilidade dessa computação, armazenamento e redes. Eles não se importam com a plataforma e a infraestrutura subjacentes. Eles não precisam gerenciar nem monitorar a infraestrutura e a plataforma. A função sem servidor fornece um ambiente que pode ser expandido e reduzido, de forma automática, sem o cliente ficar ciente disso. Todas as operações relacionadas a plataformas e infraestruturas ocorrem nos bastidores e são executadas pelo provedor de nuvem. Os clientes recebem contratos de nível de serviço (SLAs) relacionados à performance, e o Azure garante que eles atendam aos **contratos de nível de serviço (SLAs)**, independentemente da demanda total.

Os clientes só precisam trazer seu código. É responsabilidade do provedor de nuvem fornecer a infraestrutura e a plataforma necessárias para executar o código. Vamos avançar e nos aprofundar nas várias vantagens do Azure Functions.

## As vantagens do Azure Functions

A computação sem servidor é um paradigma relativamente novo que ajuda as organizações a converter grandes funcionalidades em funções menores, discretas e sob demanda que podem ser invocadas e executadas por meio de gatilhos automatizados e trabalhos agendados. Elas também são conhecidas como **Funções como Serviço (FaaS)**, nos quais as organizações podem se concentrar nos desafios do domínio, em vez de na plataforma e na infraestrutura subjacentes. As FaaS também ajudam na transferência de arquiteturas de solução para funções reutilizáveis menores, aumentando assim o retorno sobre o investimento.

Há uma infinidade de plataformas de computação sem servidor disponíveis. Algumas das mais importantes são:

- Azure Functions
- AWS Lambda
- IBM OpenWhisk
- Iron.io
- Funções do Google Cloud

De fato, todos os dias parece que uma nova plataforma/estrutura está sendo apresentada, e está se tornando cada vez mais difícil para as empresas decidir a respeito da estrutura que funciona melhor para elas. O Azure fornece um valioso ambiente sem servidor conhecido como Azure Functions. Veja a seguir alguns dos recursos compatíveis com ele:

- Diversas maneiras de invocar uma função: manual, agendada ou com base em um evento.
- Diversos tipos de suporte vinculativo.
- A capacidade de executar funções de forma síncrona e assíncrona.
- A capacidade de executar funções com base em vários tipos de gatilhos.
- A capacidade de executar funções de longa e curta duração. No entanto, as funções grandes e de longa duração não são recomendadas, pois podem causar intervalos inesperados.
- A capacidade de usar recursos de proxy para arquiteturas de função diferentes.
- Vários modelos de uso, incluindo o consumo, bem como o modelo de Serviço de Aplicativo.
- A capacidade de criar funções usando várias linguagens, como JavaScript, Python e C#.
- Autorização baseada em OAuth.
- A extensão Durable Functions ajuda a escrever funções com estado.
- Várias opções de autenticação, incluindo o Azure AD, o Facebook, o Twitter e outros provedores de identidade.
- A capacidade de configurar facilmente parâmetros de entrada e saída.
- Integração do Visual Studio para criação de funções do Azure.
- Paralelismo em massa.

Vamos ver as FaaS e quais funções elas desempenham na arquitetura sem servidor.

## FaaS

O Azure fornece FaaS. Elas são implementações sem servidor do Azure. Com o Azure Functions, o código pode ser escrito em qualquer linguagem com que o usuário se sinta à vontade, e o Azure Functions fornecerá um tempo de execução para executá-lo. Com base na linguagem escolhida, uma plataforma adequada é fornecida aos usuários por trazer o seu próprio código. As funções são uma unidade de implantação e podem ser automaticamente expandidas e reduzidas. Ao lidar com as funções, os usuários não podem exibir as máquinas virtuais e as plataformas subjacentes, mas o Azure Functions fornece uma pequena janela para visualizá-las por meio do **console do Kudu**.

Há dois componentes principais do Azure Functions:

- O tempo de execução do Azure Functions
- Associação e gatilhos do Azure Functions

Vamos aprender sobre esses componentes em detalhes.

### O tempo de execução do Azure Functions

O núcleo do Azure Functions está em seu tempo de execução. O precursor do Azure Functions foi o Azure WebJobs. O código do Azure WebJobs também forma o núcleo do Azure Functions. Existem recursos e extensões adicionais adicionados aos Azure WebJobs para criar o Azure Functions. O tempo de execução do Azure Functions é a mágica que faz as funções funcionarem. O Azure Functions é hospedado no Serviço de Aplicativo do Azure. O Serviço de Aplicativo do Azure carrega o tempo de execução do Azure e aguarda um evento externo ou uma atividade manual ocorrer. Na chegada de uma solicitação ou a ocorrência de um gatilho, o Serviço de Aplicativo carrega a carga de entrada, lê o arquivo **function.json** da função para encontrar associações e o gatilho da função, mapeia os dados de entrada para parâmetros de entrada e invoca a função com valores de parâmetro. Quando a função conclui sua execução, o valor é passado novamente para o tempo de execução do Azure Functions por meio de um parâmetro de saída definido como associação no arquivo **function.json**. O tempo de execução da função retorna os valores para o chamador. O tempo de execução do Azure Functions atua como a ligação que habilita toda a performance das funções.

A versão atual do tempo de execução do Azure é aproximadamente 3. Ela é baseada na estrutura .NET Core 3.1. Antes disso, a versão aproximadamente 2 foi baseada na estrutura .NET Core 2.2. A primeira versão, aproximadamente 1, foi baseada na estrutura .NET 4.7.

Houve alterações significativas da versão 1 para a 2 devido a alterações na própria estrutura subjacente. No entanto, há muito poucas alterações interruptivas da versão 2 para a 3, e a maioria das funções escritas na versão 2 continuarão sendo executadas na versão 3 também. No entanto, é recomendável que os testes adequados sejam feitos após a migração da versão 2 para a 3. Também houve alterações na versão 1 para a 2 em relação a gatilhos e associações. Gatilhos e associações agora estão disponíveis como extensões, cada um em uma montagem diferente nas versões 2 e 3.

## Associação e gatilhos do Azure Functions

Se o tempo de execução é o cérebro do Azure Functions, as associações e os gatilhos são o coração. O Azure Functions promove o acoplamento e a alta coesão entre serviços usando gatilhos e associações. As aplicações escritas visando ambientes que não são sem servidor implementam o código usando sintaxe imperativa para parâmetros de entrada e de saída e valores de retorno. O Azure Functions usa um mecanismo declarativo para invocar funções usando gatilhos e configura o fluxo de dados usando associações.

Vinculação refere-se ao processo de criação de uma conexão entre os dados de entrada e a função do Azure juntamente com o mapeamento dos tipos de dados. A conexão pode ser um único sentido do tempo de execução ao Azure Functions e vice-versa ou vários sentidos – a associação pode transmitir dados entre o tempo de execução do Azure e o Azure Functions nos dois sentidos. O Azure Functions define associações de forma declarativa.

Os gatilhos são um tipo especial de associação por meio da qual as funções podem ser invocadas com base em eventos externos. Além de invocar uma função, os gatilhos também transmitem os dados de entrada, a carga e os metadados para a função.

As associações são definidas no arquivo **function.json**:

```
{
  "bindings": [
    {
      "name": "checkOut",
      "type": "queueTrigger",
      "direction": "in",
      "queueName": "checkout-items",
      "connection": "AzureWebJobsDashboard"
    },
    {
      "name": "Orders",
      "type": "table",
      "direction": "out",
      "tableName": "OrderDetails",
      "connection": "<>Connection to table storage account>>"
    }
  ],
  "disabled": false
}
```

Neste exemplo, um gatilho é declarado e invoca a função sempre que há um novo item na fila de armazenamento. O tipo é **queueTrigger**, o sentido é entrada, **queueName** é **checkout-items** e os detalhes sobre a conexão da conta de armazenamento de destino e o nome da tabela também são mostrados. Todos esses valores são importantes para o funcionamento dessa associação. O nome **checkOut** pode ser usado no código da função como uma variável.

De maneira semelhante, uma associação para o valor de retorno é declarada. Aqui, o valor de retorno é chamado de **Pedidos**, e os dados são a saída do Azure Functions. A associação grava os dados de retorno no Armazenamento de Tabelas do Azure usando a cadeia de conexão fornecida.

Tanto as associações e os gatilhos podem ser modificados e criados usando a guia **Integrar** no Azure Functions. No back-end, o arquivo **function.json** é atualizado. O gatilho **checkOut** é declarado, conforme mostrado aqui:

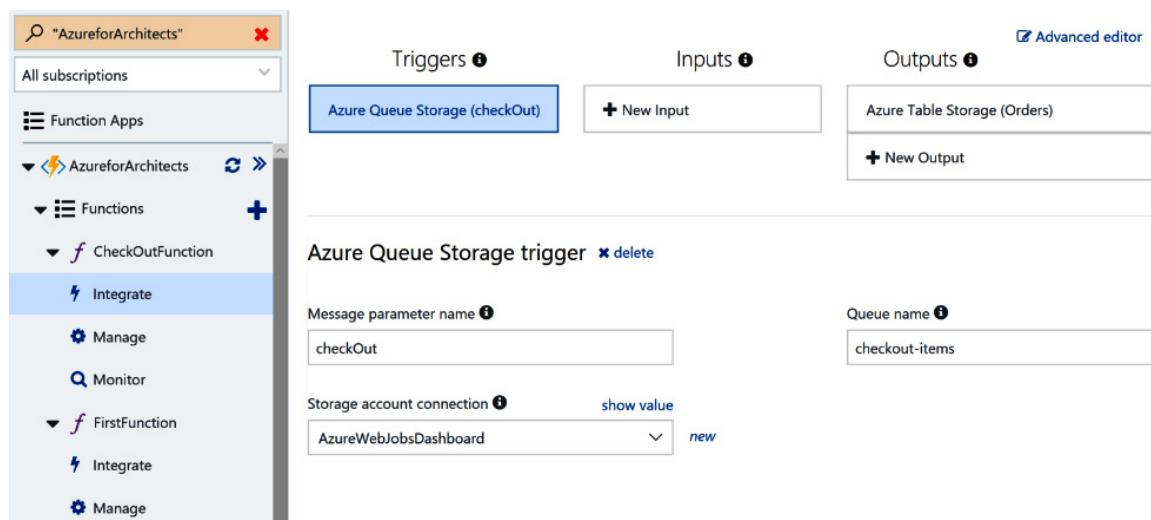


Figura 10.1: a seção Gatilhos da guia Integrar

A saída **Pedidos** é mostrada a seguir:

Figura 10.2: adicionando detalhes de saída para a conta de armazenamento

Os autores de funções do Azure não precisam gravar nenhum código de conexão para obter dados de várias fontes. Eles simplesmente decidem o tipo de dados esperado do tempo de execução do Azure. Isso é mostrado no próximo segmento de código. O check-out está disponível como uma cadeia de caracteres para a função. Vários tipos de dados podem ser usados como associação para funções. Por exemplo, uma associação de fila pode fornecer o seguinte:

- Um antigo objeto CLR (Common Language Runtime) simples (POCO)
- Uma cadeia
- Um byte
- **CloudQueueMessage**

O autor da função pode usar qualquer um desses tipos de dados, e o tempo de execução do Azure Functions garantirá que um objeto adequado como um parâmetro seja enviado para a função: O seguinte é um trecho de código para aceitar dados de cadeia e o tempo de execução do Functions encapsulará dados de entrada em um tipo de dados de **cadeia** antes de invocar a função. Se o tempo de execução não puder lançar os dados de entrada em uma cadeia, ele gerará uma exceção:

```
using System;
public static void Run(string checkOut, TraceWriter log)
{
    log.Info($"C# Queue trigger function processed: { checkOut }");
}
```

Também é importante saber que, na Figura 10.2, os nomes de conta de armazenamento são **AzureWebJobsStorage** e **AzureWebJobsDashboard**. Ambas são chaves definidas na seção **appSettings** e contêm cadeias de conexão da conta de armazenamento. Essas contas de armazenamento são usadas internamente pelo Azure Functions para manter seu estado e o status da execução da função.

Para obter mais informações sobre associações e gatilhos do Azure, consulte <https://docs.microsoft.com/azure/azure-functions/functions-bindings-storage-queue>.

Agora que temos uma boa compreensão das associações e dos gatilhos do Azure, vamos conferir as várias opções de configuração oferecidas pelo Azure Functions.

## Configuração do Azure Functions

O Azure Functions fornece opções de configuração em vários níveis. Ele fornece a configuração para o seguinte:

- A própria plataforma
- Serviços de Aplicativo do Functions

Essas configurações afetam cada função contida por elas. Mais informações sobre essas configurações estão disponíveis em <https://docs.microsoft.com/azure/azure-functions/functions-how-to-use-azure-function-app-settings>.

## Configuração de plataforma

As funções do Azure são hospedadas no Serviço de Aplicativo do Azure e, portanto, elas obtêm todos os recursos. Os logs de diagnóstico e monitoramento podem ser configurados com facilidade usando recursos de plataforma. Além disso, o Serviço de Aplicativo fornece opções para atribuir certificados SSL usando autenticação, autorização e um domínio personalizado como parte de seus recursos de rede.

Embora os clientes não estejam preocupados com a infraestrutura, o sistema operacional, o sistema de arquivos ou a plataforma na qual as funções realmente são executadas, o Azure Functions fornece as ferramentas necessárias para ter uma prévia do sistema subjacente e fazer alterações. O console no portal e o console do Kudu são as ferramentas usadas para essa finalidade. Eles fornecem um editor avançado para criar as funções do Azure e editar sua configuração.

Assim como o Serviço de Aplicativo, o Azure Functions permite o armazenamento de informações de configuração na seção de configuração da aplicação `web.config`, que pode ser lida sob demanda. Alguns dos recursos da plataforma de aplicativos de função são mostrados na *Figura 10.3*:

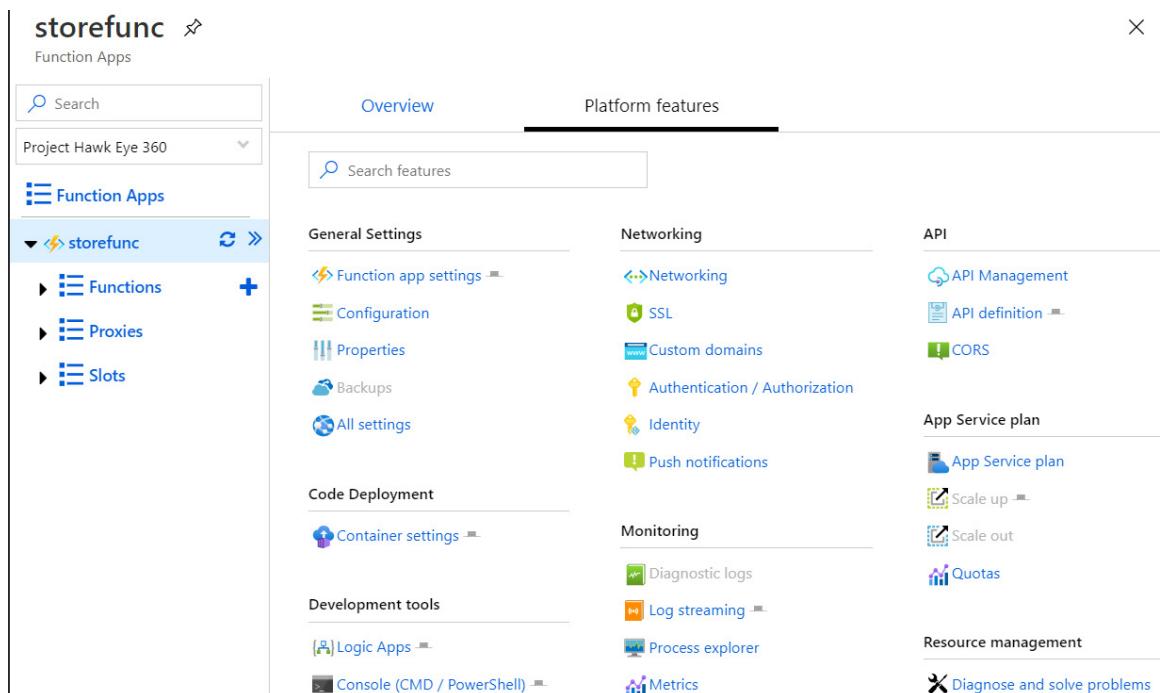


Figura 10.3: recursos da plataforma de um aplicativo de função

Esses recursos de plataforma podem ser usados para configurar autenticação, domínios personalizados, SSL e assim por diante. Além disso, a guia **Recursos da plataforma** fornece uma visão geral das ferramentas de desenvolvimento que podem ser usadas com o aplicativo de função. Na próxima seção, vamos ver as configurações do aplicativo de função que estão disponíveis nos recursos da plataforma.

## Configurações da função de Serviço de Aplicativo

Essas configurações afetam todas as funções. As configurações da aplicação podem ser gerenciadas aqui. Os proxies no Azure Functions podem ser habilitados ou desabilitados. Nós discutiremos os proxies mais à frente neste capítulo. Eles também ajudam na alteração do modo de edição de uma aplicação de função e a implantação em slots:

The screenshot shows the 'Function app settings' tab selected in the top navigation bar. A yellow warning banner at the top left states: '⚠ Application Insights is not configured. Configure Application Insights to capture function logs.' Below this, there are tabs for 'Overview', 'Platform features', and 'Function app settings'. Under 'Function app settings', there is a 'Daily Usage Quota (GB-Sec)' section with a text input field 'Enter value in GB-sec' and a 'Set quota' button. A 'Configuration' section includes a link to 'Manage application settings'. The 'Runtime version' section shows 'Runtime version: 3.0.13107 (~3)'. A yellow warning box titled 'Cannot Upgrade with Existing Functions' states: 'Major version upgrades can introduce breaking changes to languages and bindings. When upgrading major versions of the runtime, consider creating a new function app and migrate your functions to this new app.' Below this, there are three buttons: '~1', '~2', and a blue highlighted button '~3'. The 'Function app edit mode' section allows changing the mode from 'Read/Write' to 'Read Only'. The 'Host Keys (All functions)' section lists two entries: '\_master' and 'default', each with a 'Click to show' link and a 'Copy' button. A blue 'Add new host key' button is at the bottom.

Figura 10.4: configurações do aplicativo de função

O orçamento é um aspecto muito importante do sucesso de qualquer projeto. Vamos explorar os vários planos oferecidos para o Azure Functions.

## Planos de custo do Azure Functions

O Azure Functions baseia-se no Serviço de Aplicativo do Azure e fornece um modelo de custos mais econômico para os usuários. Existem três modelos de custo.

## Um plano de consumo

Isso se baseia no consumo por segundo e na execução de funções. Esse plano calcula o custo com base no uso de computação durante o consumo real e a execução da função. Se uma função não for executada, não há nenhum custo associado a ela. No entanto, isso não significa que a performance esteja comprometida nesse plano. O Azure Functions será expandido e reduzido de forma automática com base na demanda para garantir que os níveis mínimos básicos de performance sejam mantidos. Uma execução de função é permitida 10 minutos para a sua conclusão.

Uma das principais desvantagens desse plano é que, se não houver nenhum consumo de funções por alguns segundos, a função poderá ficar fria e a próxima solicitação que surgir poderá enfrentar um pequeno atraso na obtenção de uma resposta, pois a função está ociosa. Esse fenômeno é chamado de inicialização a frio. No entanto, existem soluções alternativas que podem manter as funções ativas, mesmo quando não há solicitações legítimas. Isso pode ser feito escrevendo uma função agendada que continua invocando a função de destino para mantê-la ativa.

## Um plano Premium

Este é um plano relativamente novo e oferece muitos benefícios em comparação com o Serviço de Aplicativo e um plano de consumo. Nesse plano, não há inicializações a frio para as funções do Azure. As funções podem ser associadas a uma rede privada, e os clientes podem escolher seus próprios tamanhos de máquina virtual para executar funções. Ele fornece inúmeras instalações prontas para uso que não eram possíveis anteriormente com os outros dois tipos de planos.

## Um plano do Serviço de Aplicativo

Este plano fornece funções com máquinas virtuais totalmente dedicadas no back-end e, por isso, o custo é diretamente proporcional ao custo da máquina virtual e seu tamanho. Há um custo associado fixo a esse plano, mesmo se as funções não forem invocadas. O código de função pode ser executado durante o tempo necessário. Embora não haja nenhuma restrição de tempo, o limite padrão é definido como 30 minutos. Isso pode ser alterado com a alteração do valor no arquivo `hosts.json`. No âmbito do plano de Serviço de Aplicativo, o tempo de execução da função fica ocioso se não utilizado por alguns minutos e pode ser ativado somente usando um gatilho de HTTP. Há uma configuração **Always On** que pode ser usada para impedir que o tempo de execução da função fique ocioso. A escalabilidade é manual ou baseado em configurações de escalabilidade automática.

Juntamente com a opção de preços flexível, o Azure também oferece várias opções de hospedagem para implantação de arquitetura.

## Hosts de destino do Azure Functions

O tempo de execução do Azure Functions pode ser hospedado no Windows, bem como em hosts do Linux. As funções baseadas em .NET Core, Python, Java, Node.js e PowerShell Core podem ser executadas em sistemas operacionais Windows e Linux. É importante saber qual tipo de sistema operacional subjacente é necessário para as funções porque essa definição de configuração está vinculada ao aplicativo de função e a todas as funções contidas nele.

Além disso, é possível executar funções dentro de contêineres do Docker. Isso ocorre porque o Azure fornece imagens do Docker que têm um tempo de execução de função pré-criado instalado nelas, e as funções podem ser hospedadas usando essas imagens. Agora, as imagens do Docker podem ser usadas para criar contêineres nos pods do Kubernetes e hospedadas no Serviço de Kubernetes do Azure, Instâncias de Contêiner do Azure ou em clusters do Kubernetes não gerenciados. Essas imagens podem ser armazenadas no Hub do Docker, no Registro de Contêiner do Azure ou em quaisquer outros repositórios de imagens globais, bem como privados.

Para ter uma compreensão mais clara, vamos ver alguns dos casos de uso mais proeminentes para o Azure Functions.

## Casos de uso do Azure Functions

O Azure Functions têm muitas implementações. Vamos ver alguns desses casos de uso.

### Implementação de microsserviços

O Azure Functions ajuda na divisão de aplicações grandes em unidades de código funcional discretas e menores. Cada unidade é tratada de forma independente das outras e evolui em seu próprio ciclo de vida. Cada unidade de código tem sua própria computação, hardware e requisitos de monitoramento. Cada função pode ser conectada a todas as outras funções. Essas unidades são tecidas juntas por orquestradores para criar funcionalidade completa. Por exemplo, em uma aplicação de comércio eletrônico, pode haver funções individuais (unidades de código), cada uma responsável por listagem de catálogos, recomendações, categorias, subcategorias, compras, carrinhos, check-outs, tipos de pagamento, gateways de pagamento, endereços de entrega, endereços de faturamento, impostos, frete, cancelamentos, devoluções, emails, mensagens SMS e assim por diante. Algumas dessas funções são reunidas para criar casos de uso para aplicações de comércio eletrônico, como navegação de produtos e fluxo de check-out.

### Integração entre vários pontos de extremidade

O Azure Functions pode criar funcionalidade geral de aplicação ao integrar várias funções. A integração pode se basear no disparo de eventos ou pode ser feita por push. Ela ajuda na decomposição de grandes aplicações monolíticas em pequenos componentes.

## Processamento de dados

O Azure Functions pode ser usado para o processamento de dados de entrada em lotes. Ele pode ajudar no processamento de dados vem em vários formatos, como XML, CSV, JSON e TXT. Ele também pode executar algoritmos de conversão, enriquecimento, limpeza e filtragem. Na verdade, várias funções podem ser usadas, cada uma fazendo conversão ou enriquecimento, limpeza ou filtragem. O Azure Functions também pode ser usado para incorporar serviços cognitivos avançados, como **reconhecimento óptico de caracteres (OCR)**, visão computacional e manipulação e conversão de imagem. Isso é ideal se você quiser processar as respostas de API e convertê-las.

## Integração de aplicações herdadas

O Azure Functions pode ajudar na integração de aplicações herdadas com novos protocolos e aplicações modernas. É possível que as aplicações herdadas não estejam usando protocolos e formatos padrão da indústria. O Azure Functions pode atuar como um proxy para essas aplicações herdadas, aceitar solicitações de usuários ou outras aplicações, converter os dados em um formato compreendido por uma aplicação herdada e se comunicar com ela com protocolos que ela entenda. Isso possibilita muitas oportunidades para integrar e trazer aplicações antigas e herdadas para o portfólio principal.

## Trabalhos agendados

O Azure Functions pode ser usado para executar de forma contínua ou periodicamente para determinadas funções da aplicação. Essas funções de aplicação podem realizar tarefas como fazer backups periódicos, restaurar, executar trabalhos de lote, exportar e importar dados e enviar emails em massa.

## Gateways de comunicação

O Azure Functions pode ser usado em gateways de comunicação ao usar hubs de notificação, mensagens SMS e assim por diante. Por exemplo, você pode usar o Azure Functions para enviar uma notificação por push para dispositivos Android e iOS usando Hubs de Notificações do Azure.

As funções do Azure estão disponíveis em diferentes tipos, que devem ser selecionadas com base em sua relação com a otimização de workloads. Vamos vê-las em mais detalhes.

## Tipos de funções do Azure

As funções do Azure podem ser categorizadas em três tipos diferentes:

- **Funções sob demanda:** são executadas quando explicitamente chamadas ou invocadas. Os exemplos de tais funções incluem funções e webhooks baseados em HTTP.
- **Funções agendadas:** são como trabalhos de timer e executam as funções em intervalos fixos.
- **Funções baseadas em evento:** são executadas com base em eventos externos. Por exemplo, o upload de um novo arquivo para o Armazenamento de Blobs do Azure gera um evento que pode iniciar a execução de funções do Azure.

Na seção a seguir, você aprenderá a criar uma função orientada a eventos que será vinculada a uma conta de armazenamento do Azure.

## Criar uma função orientada a eventos

Neste exemplo, uma função do Azure será criado e conectado a uma conta do armazenamento do Azure. A conta de armazenamento tem um contêiner para armazenar todos os arquivos de blob. O nome da conta de armazenamento é **incomingfiles** e o contêiner é **orders**, conforme mostrado na Figura 10.5:

The screenshot shows the Azure Blob service interface for the 'incomingfiles' storage account. At the top, it displays the storage account name and a 'Container' button. Below this, it shows the following details:

- Storage account: incomingfiles
- Status: Primary: Available
- Location: West Central US
- Subscription: RiteshSubscription
- Subscription ID: 9755ffce-e94b-4332-9be8-1ade15e78909

At the bottom, there is a search bar labeled 'Search containers by prefix' and a table showing the single container named 'orders'.

NAME
orders

Figura 10.5: detalhes da conta de armazenamento

Execute as seguintes etapas para criar uma nova função do Azure no portal do Azure:

1. Clique no botão + ao lado do menu **Funções** à esquerda.
2. Selecione **No portal** na tela resultante e clique no botão **Continuar**.
3. Selecione **Gatilho de Armazenamento de Blobs do Azure**, conforme mostrado na **Figura 10.6**:

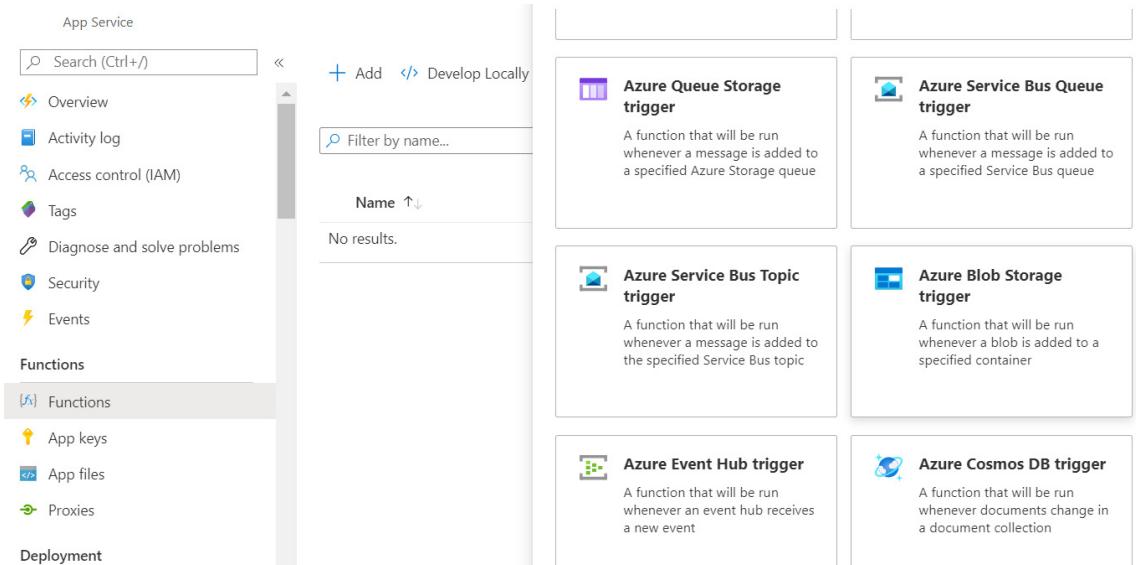


Figura 10.6: selecionando o gatilho de Armazenamento de Blobs do Azure

Agora, essa função do Azure não tem conectividade com a conta de armazenamento. As funções do Azure precisam de informações de conexão para a conta de armazenamento e está disponível na guia **Teclas de acesso** na conta de armazenamento. A mesma informação pode ser obtida usando o ambiente do editor do Azure Functions. Na verdade, esse ambiente permite a criação de uma nova conta de armazenamento do mesmo ambiente de editor.

O gatilho de Armazenamento de Blobs do Azure pode ser adicionado usando o botão **Novo** ao lado do tipo de entrada de **Conexão de conta de armazenamento**.

Ele permite selecionar uma conta de armazenamento existente ou criar uma nova. Como já tenho algumas contas de armazenamento, estou reutilizando-as, mas você deve criar uma conta de armazenamento do Azure diferente. Ao selecionar uma conta de armazenamento, você atualizará as configurações na seção **appSettings** com a cadeia de conexão adicionada a ela.

Verifique se um contêiner já existe no serviço de blob da conta de armazenamento do Azure de destino. A entrada do caminho se refere ao caminho para o contêiner. Neste caso, o contêiner **orders** já existe na conta de armazenamento. O botão **Criar** mostrado aqui provisionará a nova função que monitora o contêiner de conta de armazenamento:

Figura 10.7: criando uma função que monitora o contêiner da conta de armazenamento

O código para a função **storagerelatedfunctions** é da seguinte forma:

```
public static void Run(Stream myBlob, TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n \n Size {myBlob.Length} Bytes");
}
```

As associações são mostradas aqui:

```
{
  "bindings": [
    {
      "name": "myBlob",
      "type": "blobTrigger",
      "direction": "in",
      "path": "orders",
      "connection": "azureforarchitead2b_STORAGE"
    }
  ],
  "disabled": false
}
```

Agora, o upload de qualquer arquivo blob no contêiner **orders** deve acionar a função:

The screenshot shows the Azure Functions developer portal interface. At the top, there is a toolbar with a 'Save' button and a 'Run' button. Below the toolbar is the code editor window containing the following C# code:

```

1 public static void Run(Stream myBlob, TraceWriter log)
2 {
3     log.Info($"C# Blob trigger function Processed blob\n \n Size: {myBlob.Length}");
4 }
5

```

Below the code editor is the 'Logs' section. It displays the following log entries:

- 2017-09-20T10:24:12.504 Function started (Id=ac68a8f8-712f-4df8-975e-238b0ecf5b05)
- 2017-09-20T10:24:12.536 C# Blob trigger function Processed blob
- Size: 2480471 Bytes
- 2017-09-20T10:24:12.536 Function completed (Success, Id=ac68a8f8-712f-4df8-975e-238b0ecf5b05, Dur

Figura 10.8: blob processado pela função de gatilho de blob C#

Na próxima seção, vamos nos aprofundar no Azure Function Proxies, que ajudarão você a lidar com as solicitações e respostas das suas APIs de forma eficiente.

## Function Proxies

O Azure Function Proxies é uma adição relativamente nova ao Azure Functions. O Function Proxies ajuda a ocultar os detalhes das funções do Azure e expor pontos de extremidade completamente diferentes aos clientes. O Function Proxies pode receber solicitações em pontos de extremidade, modificar o conteúdo, o corpo, os cabeçalhos e a URL da solicitação alterando os valores e aumentá-los com dados adicionais e passá-los internamente para as funções do Azure. Assim que receberem uma resposta dessas funções, podem novamente converter, substituir e aumentar a resposta e devolvê-la ao cliente.

Ele também ajuda a invocar diferentes funções para operações CRUD (criação, leitura, exclusão e atualização) usando cabeçalhos diferentes, assim dividindo grandes funções em menores. Ele fornece um nível de segurança ao não expor o ponto de extremidade da função original e também ajuda a alterar a implementação e os pontos de extremidade da função interna sem afetar seu chamador. O Function Proxies ajuda ao fornecer aos clientes uma única URL de função e, em seguida, invocar várias funções do Azure nos back-end para concluir fluxos de trabalho. Mais informações sobre o Azure Function Proxies podem ser encontradas em <https://docs.microsoft.com/azure/azure-functions/functions-proxies>.

Na próxima seção, vamos abordar o Durable Functions em detalhes.

## Durable Functions

O Durable Functions é uma das adições mais recentes ao Azure Functions. Ele permite que os arquitetos criem fluxos de trabalho com estado em uma função de orquestrador, que é um novo tipo de função. Como desenvolvedor, você pode optar por codificar ou usar qualquer forma de IDE. Algumas vantagens do uso do Durable Functions são:

- A saída da função pode ser salva para variáveis locais, e você pode chamar outras funções de forma síncrona e assíncrona.
- O estado é preservado para você.

O seguinte é o mecanismo básico para invocar o Durable Functions:

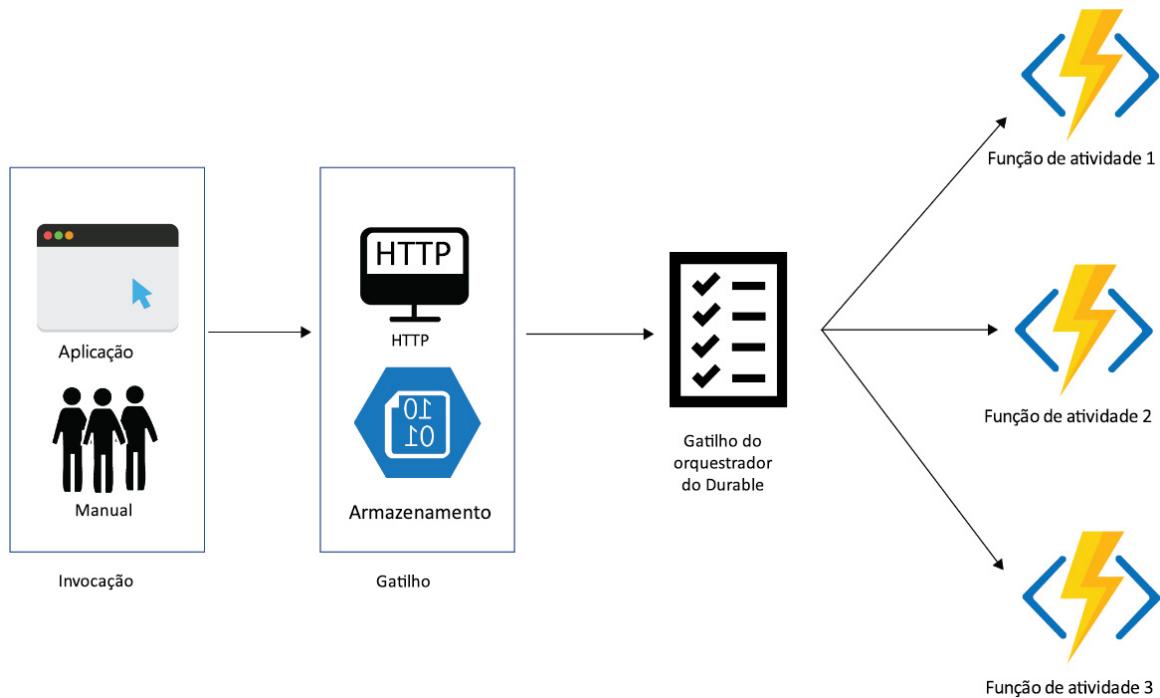


Figura 10.9: mecanismo para invocar o Durable Functions

O Azure Durable Functions pode ser invocado por qualquer gatilho fornecido pelo Azure Functions. Esses gatilhos incluem HTTP, Armazenamento de Blobs, Armazenamento de Tabelas, filas do Barramento de Serviço e muito mais. Eles podem ser acionados manualmente por alguém com acesso a eles ou por uma aplicação. A Figura 10.9 mostra alguns gatilhos como exemplo. Eles também são conhecidos como Durable Functions iniciais. O Durable Functions inicial invoca o **gatilho do orquestrador do Durable**, que contém a lógica principal para a orquestração e orquestra a invocação das funções da atividade.

O código escrito no orquestrador durável deve ser determinístico. Isso significa que, independentemente do número de vezes que o código é executado, os valores retornados por ele devem permanecer os mesmos. A função de orquestrador é uma função de longa duração por natureza. Isso significa que ele pode ser hidratado, serializado em estado e hibernar depois que ele chamar uma função de atividade durável. Isso ocorre porque ele não sabe quando a função de atividade durável será concluída e não quer esperar por ela. Quando a função de atividade durável termina sua execução, a função de orquestrador é executada novamente. A execução da função começa do início e executada até que ela chame outra função de atividade durável ou termine a execução da função. Ela precisa executar novamente as linhas de código que já executou anteriormente e deve obter os mesmos resultados que obteve anteriormente. Observe que o código escrito no orquestrador durável deve ser determinístico. Isso significa que, independentemente do número de vezes que o código é executado, os valores retornados por ele devem permanecer os mesmos.

Vou explicar isso com a ajuda de um exemplo. Se usarmos uma classe geral `DateTime` do Core .NET Core e retornarmos à data e hora atual, isso resultará em um novo valor toda vez que executarmos a função. O objeto de contexto do Durable Functions fornece `CurrentUtcDateTime`, que retornará o mesmo valor de data e hora durante a nova execução que retornou na primeira vez.

Essas funções de orquestração também podem aguardar eventos externos e habilitar cenários relacionados à transferência humana. Esse conceito será explicado mais adiante nesta seção.

Essas funções de atividade podem ser chamadas com ou sem um mecanismo de repetição. O Durable Functions pode ajudar a resolver muitos desafios e a fornecer recursos para escrever funções que podem:

- Executar funções de longa duração
- Manter o estado
- Executar funções filhas em paralelo ou em sequência
- Recuperar-se da falha facilmente
- Orquestrar a execução de funções em um fluxo de trabalho

Agora que você tem uma boa compreensão do funcionamento interno de uma função durável, vamos explorar como criar uma função durável no Visual Studio.

## Etapas para criar uma função durável usando o Visual Studio

Veja a seguir as etapas para criar uma função durável:

1. Navegue até o portal do Azure e clique em **Grupos de recursos** no menu à esquerda.
2. Clique no botão **+Adicionar** no menu superior para criar um novo grupo de recursos.
3. Forneça as informações do grupo de recursos no formulário resultante e clique no botão **Criar**, conforme mostrado aqui:

Create a resource group

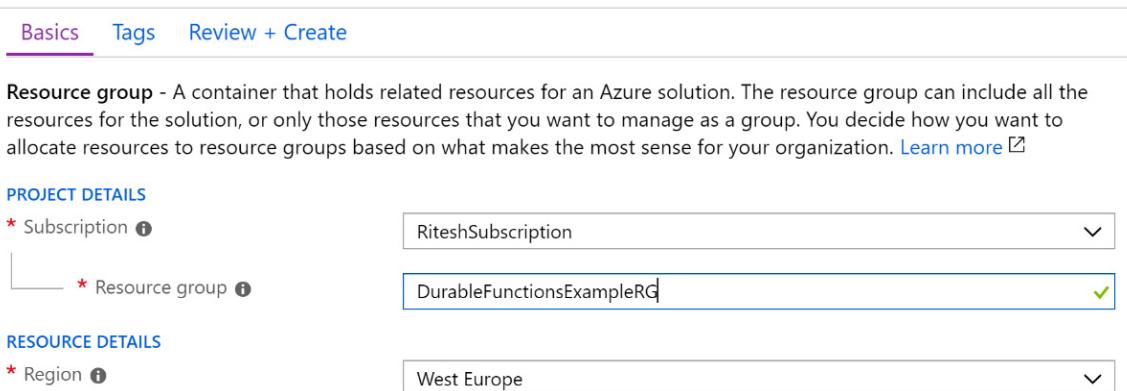


Figura 10.10: criando um grupo de recursos

4. Navegue até o grupo de recursos recém-criado e adicione um novo aplicativo de função. Para isso, clique no botão **+Adicionar** no menu superior e procure **aplicativo de função** na caixa de pesquisa resultante.
5. Selecione **Aplicativo de função** e clique no botão **Criar**. Preencha o formulário do aplicativo de função resultante e clique no botão **Criar**. Você também pode reutilizar o aplicativo de função que criamos anteriormente.
6. Depois que o aplicativo de função for criado, entraremos em nosso ambiente de desenvolvimento local com o Visual Studio 2019 instalado nele. Vamos começar com o Visual Studio e criar um novo projeto do tipo **funções do Azure**, fornecê-lo com um nome e selecionar **Azure Functions v3 (.NET Core)** para o **Tempo de execução da função**.
7. Depois que o projeto for criado, precisaremos adicionar o pacote NuGet **DurableTask** ao projeto para trabalhar com o Durable Functions. A versão usada no momento da redação deste capítulo é 2.2.2:



Figura 10.11: adicionando um pacote NuGet DurableTask

8. Agora, podemos codificar nossas funções duráveis no Visual Studio. Adicione uma nova função, forneça um nome e selecione o tipo de gatilho **Orquestração do Durable Functions**:

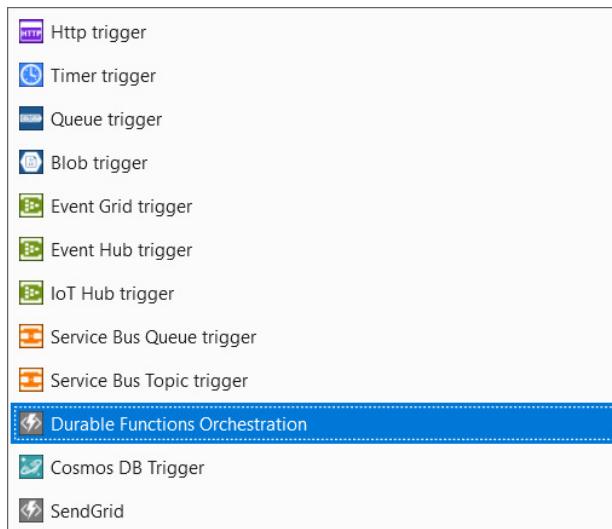


Figura 10.12: selecionando um gatilho de orquestração do Durable Functions

9. O Visual Studio gera o código clichê para o Durable Functions, e vamos usá-lo para saber mais sobre o Durable Functions. As atividades do Durable Functions são funções invocadas pela função de orquestrador principal. Geralmente, há uma função de orquestrador principal e várias atividades do Durable Functions. Após a instalação da extensão, forneça um nome para a função e escreva um código que faça algo útil, como enviar um email ou mensagem SMS, conectar-se a sistemas externos e executar lógica ou executar serviços usando seus pontos de extremidade, como os serviços cognitivos.

O Visual Studio gera três conjuntos de funções em uma única linha de código:

- **HttpStart:** esta é a função inicial. Isso significa que ele é responsável por iniciar a orquestração da função durável. O código gerado consiste em uma função inicial do gatilho de HTTP. No entanto, pode ser qualquer função baseada em gatilho, como **BlobTrigger**, uma fila **ServiceBus** ou uma função baseada em gatilho.
- **RunOrchestrator:** esta é a principal função de orquestração durável. Ela é responsável por aceitar parâmetros da função inicial e invocar várias funções de tarefa duráveis. Cada função de tarefa durável é responsável por uma funcionalidade, e essas tarefas duráveis podem ser invocadas em paralelo ou em sequência, dependendo da necessidade.
- **SayHello:** esta é a função de tarefa durável que é invocada no orquestrador de função durável para realizar um trabalho específico.

10. O código da função inicial (**HttpStart**) é mostrado a seguir. Essa função tem um gatilho do tipo HTTP e aceita uma associação adicional do tipo **DurableClient**. Esse objeto **DurableClient** ajuda a invocar a função de orquestrador:

```
[FunctionName("Function1_HttpStart")]
0 references
public static async Task<HttpResponseMessage> HttpStart(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")] HttpRequestMessage req,
    [DurableClient] IDurableOrchestrationClient starter,
    ILogger log)
{
    // Function input comes from the request content.
    string instanceId = await starter.StartNewAsync("Function1", null);

    log.LogInformation($"Started orchestration with ID = '{instanceId}'.");

    return starter.CreateCheckStatusResponse(req, instanceId);
}
```

Figura 10.13: código para a função inicial

11. O código da função de orquestrador (**RunOrchestrator**) é mostrado a seguir. Essa função tem um gatilho de tipo **OrchestrationTrigger** e aceita um parâmetro do tipo **IDurableOrchestrationContext**. Esse objeto de contexto ajuda a invocar tarefas duráveis:

```
[FunctionName("Orchestrator")]
0 references
public static async Task<List<string>> RunOrchestrator(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    var outputs = new List<string>();

    // Replace "hello" with the name of your Durable Activity Function.
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Tokyo"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Seattle"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "London"));

    // returns ["Hello Tokyo!", "Hello Seattle!", "Hello London!"]
    return outputs;
}
```

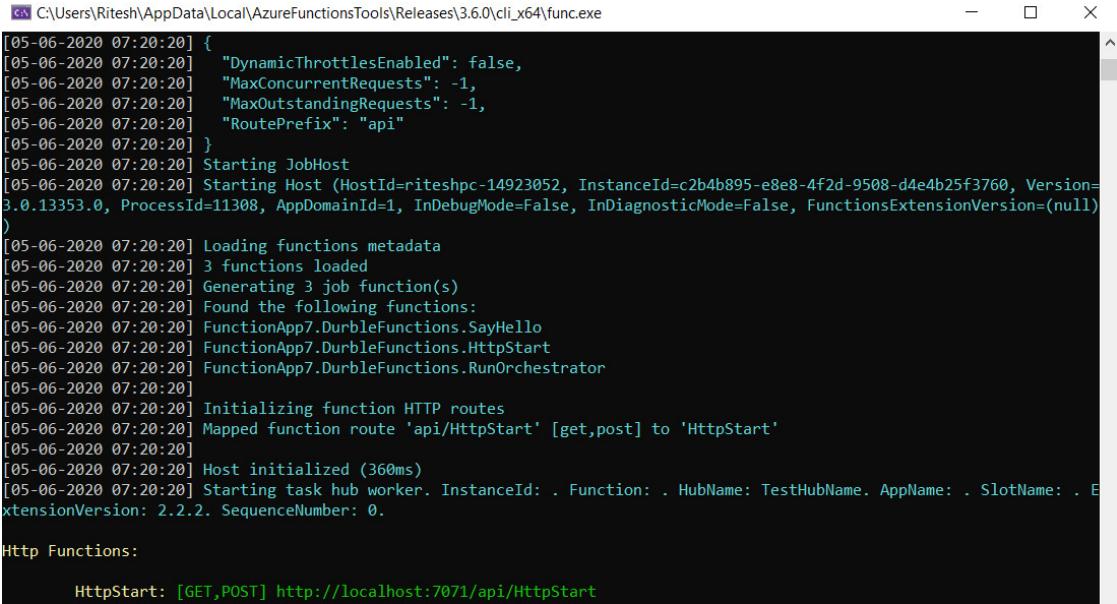
Figura 10.14: código para a função de gatilho do orquestrador

12. O código da função de tarefa durável (**HelloFunction**) é mostrado a seguir. Essa função tem um gatilho do tipo **ActivityTrigger** e aceita um parâmetro que pode ser qualquer tipo necessário para que ele execute sua funcionalidade. Ele tem um valor de retorno do tipo **cadeia**, e a função é responsável por retornar um valor de cadeia para a função de orquestração:

```
[FunctionName("HelloFunction")]
0 references
public static string SayHello([ActivityTrigger] string name, ILogger log)
{
    log.LogInformation($"Saying hello to {name}.");
    return $"Hello {name}!";
}
```

Figura 10.15: código para a função de tarefa durável

Em seguida, podemos executar a função localmente, que iniciará um emulador de armazenamento se ainda não foi iniciado e fornecerá uma URL para a função de gatilho de HTTP:



```
C:\Users\Ritesh\AppData\Local\AzureFunctionsTools\Releases\3.6.0\cli_x64\func.exe
[05-06-2020 07:20:20] {
[05-06-2020 07:20:20]   "DynamicThrottlesEnabled": false,
[05-06-2020 07:20:20]   "MaxConcurrentRequests": -1,
[05-06-2020 07:20:20]   "MaxOutstandingRequests": -1,
[05-06-2020 07:20:20]   "RoutePrefix": "api"
[05-06-2020 07:20:20] }
[05-06-2020 07:20:20] Starting JobHost
[05-06-2020 07:20:20] Starting Host (HostId=riteshpc-14923052, InstanceId=c2b4b895-e8e8-4f2d-9508-d4e4b25f3760, Version=3.0.13353.0, ProcessId=11308, AppDomainId=1, InDebugMode=False, InDiagnosticMode=False, FunctionsExtensionVersion=(null))
[05-06-2020 07:20:20] Loading functions metadata
[05-06-2020 07:20:20] 3 functions loaded
[05-06-2020 07:20:20] Generating 3 job function(s)
[05-06-2020 07:20:20] Found the following functions:
[05-06-2020 07:20:20] FunctionApp7.DurableFunctions.SayHello
[05-06-2020 07:20:20] FunctionApp7.DurableFunctions.HttpStart
[05-06-2020 07:20:20] FunctionApp7.DurableFunctions.RunOrchestrator
[05-06-2020 07:20:20]
[05-06-2020 07:20:20] Initializing function HTTP routes
[05-06-2020 07:20:20] Mapped function route 'api/HttpStart' [get,post] to 'HttpStart'
[05-06-2020 07:20:20]
[05-06-2020 07:20:20] Host initialized (360ms)
[05-06-2020 07:20:20] Starting task hub worker. InstanceId: . Function: . HubName: TestHubName. AppName: . SlotName: . ExtensionVersion: 2.2.2. SequenceNumber: 0.

Http Functions:

HttpStart: [GET,POST] http://localhost:7071/api/HttpStart
```

Figura 10.16: iniciando o emulador de armazenamento

Vamos invocar essa URL usando uma ferramenta conhecida como **Postman** (de que é possível fazer o download em <https://www.getpostman.com/>). Só precisamos copiar a URL e executá-la no Postman. Essa atividade é mostrada na Figura 10.17:

The screenshot shows the Postman interface with a GET request to `http://localhost:7071/api/HttpStart`. The response status is `202 Accepted`, time `1581 ms`, and size `1.3 KB`. The response body is a JSON object containing five URLs:

```

1  "id": "5d9943bba7c943f8acd28d1c29df9617",
2  "statusQueryGetUri": "http://localhost:7071/runtime/webhooks/durabletask/instances/5d9943bba7c943f8acd28d1c29df9617?taskHub=TestHubName&connection=Storage&code=SopCOYBJTYCMkXhrKFJ2CHmanf1yHtx8EdhqVe7KELrRwaS0U2mtGw==",
3  "sendEventPostUri": "http://localhost:7071/runtime/webhooks/durabletask/instances/5d9943bba7c943f8acd28d1c29df9617/raiseEvent/{eventName}?taskHub=TestHubName&connection=Storage&code=SopCOYBJTYCMkXhrKFJ2CHmanf1yHtx8EdhqVe7KELrRwaS0U2mtGw==",
4  "terminatePostUri": "http://localhost:7071/runtime/webhooks/durabletask/instances/5d9943bba7c943f8acd28d1c29df9617/terminate?reason={text}&taskHub=TestHubName&connection=Storage&code=SopCOYBJTYCMkXhrKFJ2CHmanf1yHtx8EdhqVe7KELrRwaS0U2mtGw==",
5  "purgeHistoryDeleteUri": "http://localhost:7071/runtime/webhooks/durabletask/instances/5d9943bba7c943f8acd28d1c29df9617?taskHub=TestHubName&connection=Storage&code=SopCOYBJTYCMkXhrKFJ2CHmanf1yHtx8EdhqVe7KELrRwaS0U2mtGw=="
    
```

Figura 10.17: invocando URLs usando o Postman

Lembre-se de que cinco URLs são geradas quando você inicia o orquestrador:

- A URL **statusQueryGetUri** é usada para localizar o status atual do orquestrador. Ao clicar nessa URL no Postman, uma nova guia será aberta e, se executarmos essa solicitação, o status do fluxo de trabalho será exibido:

The screenshot shows the Postman interface with a GET request to `/statusQueryGetUri`. The response body is a JSON object representing the orchestrator's status:

```

{
  "name": "Orchestrator",
  "instanceId": "5e3af8a1f19e4a31967a1ae08fea47ba",
  "runtimeStatus": "Completed",
  "input": null,
  "customStatus": null,
  "output": [
    "Hello Tokyo!",
    "Hello Seattle!",
    "Hello London!"
  ],
  "createdTime": "2020-06-05T07:24:12Z",
  "lastUpdatedTime": "2020-06-05T07:24:42Z"
}
    
```

Figura 10.18: status atual do orquestrador

- O URL **terminatePostUri** é usado para interromper uma função de orquestrador já em execução.
- A URL **sendEventPostUri** é usada para postar um evento em uma função durável suspensa. As funções duráveis poderão ser suspensas se estiverem aguardando um evento externo. Essa URL é usada nesses casos.

- A URL `purgeHistoryDeleteUri` é usada para excluir o histórico mantido pelo Durable Functions para uma invocação específica na conta de Armazenamento de Tabelas.

Agora que você sabe como trabalhar com o Durable Functions usando o Visual Studio, vamos abordar outro aspecto das funções do Azure: encadeá-las.

## Criar uma arquitetura conectada com funções

Uma arquitetura conectada com funções refere-se à criação de várias funções, em que a saída de uma função aciona outra função e fornece dados para a próxima função executar sua lógica. Nesta seção, vamos continuar com o cenário anterior da conta de armazenamento. Nesse caso, a saída da função acionada usando arquivos de Armazenamento de Blobs do Azure gravará o tamanho do arquivo no Azure Cosmos DB.

A configuração do Cosmos DB é mostrada a seguir. Por padrão, não há nenhuma coleção criada no Cosmos DB.

Uma coleção será criada automaticamente ao criar uma função que será acionada quando o Cosmos DB obtiver dados:

### Create Azure Cosmos DB Account

The screenshot shows the 'Create Azure Cosmos DB Account' wizard. It has three main sections: 'Project Details', 'Instance Details', and 'Location'. In 'Project Details', there's a note about creating a new account with multi-region writes by February 29, 2020, to receive up to 33% off. The 'Subscription' dropdown is set to 'Microsoft Azure Sponsorship'. In 'Instance Details', the 'Account Name' is 'azureforarchitectthirdedition', the 'API' is 'Core (SQL)', and the 'Location' is '(US) West US'. Under 'Geo-Redundancy', 'Enable' is selected. At the bottom, a note states that up to 33% off is available for qualifying new accounts created between December 1, 2019, and February 29, 2020, with limitations on account locations and geo-redundancy.

**Project Details**

Create a new Azure Cosmos DB account with multi-region writes in any region by February 29, 2020 and receive up to 33% off for the life of your account.

**Subscription \*** Microsoft Azure Sponsorship

**Resource Group \*** AzureForArchitectsThirdEdition [Create new](#)

**Instance Details**

**Account Name \*** azureforarchitectthirdedition

**API \*** Core (SQL)

**Apache Spark** Notebooks (preview) Notebooks with Apache Spark (preview) [None](#) [Sign up for Apache Spark preview](#)

**Location \*** (US) West US

**Geo-Redundancy** [Enable](#) [Disable](#)

**Multi-region Writes** [Enable](#) [Disable](#)

\*Up to 33% off multi-region writes is available to qualifying new accounts only. Accounts must be created between December 1, 2019 and February 29, 2020. Offer limited to accounts with both account locations and geo-redundancy, and applies only to multi-region writes in those same regions. Both Geo-Redundancy and Multi-region Writes must be enabled in account settings. Actual discount will vary based on number of qualifying regions selected.

[Review + create](#) [Previous](#) [Next: Networking](#)

Figura 10.19: criando uma conta do Azure Cosmos DB

Vamos seguir as etapas abaixo para obter dados para a próxima função da saída de uma função.

1. Crie um novo banco de dados, **testdb**, no Cosmos DB e uma nova coleção chamada **testcollection** dentro dele. Você precisa do nome do banco de dados e da coleção ao configurar as funções do Azure:

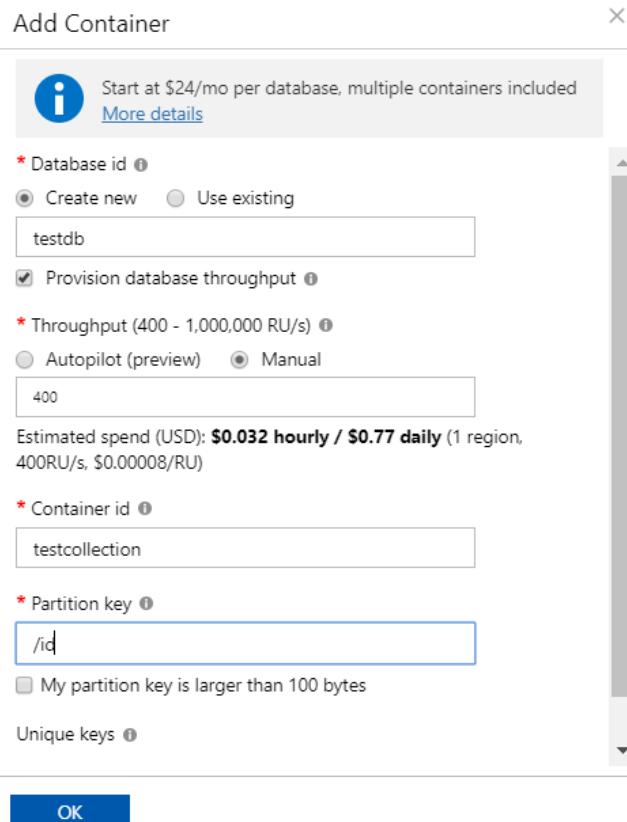


Figura 10.20: adicionando um contêiner

2. Crie uma nova função que terá um gatilho do Armazenamento de Blobs e uma associação de saída do CosmosDB. O valor retornado da função será o tamanho dos dados para o arquivo carregado. Esse valor retornado será gravado no Cosmos DB. A associação de saída será gravada na coleção do Cosmos DB. Navegue até a guia **Integrar**, clique no botão **Nova Saída** abaixo do rótulo **Saídas** e selecione **Azure Cosmos DB**:

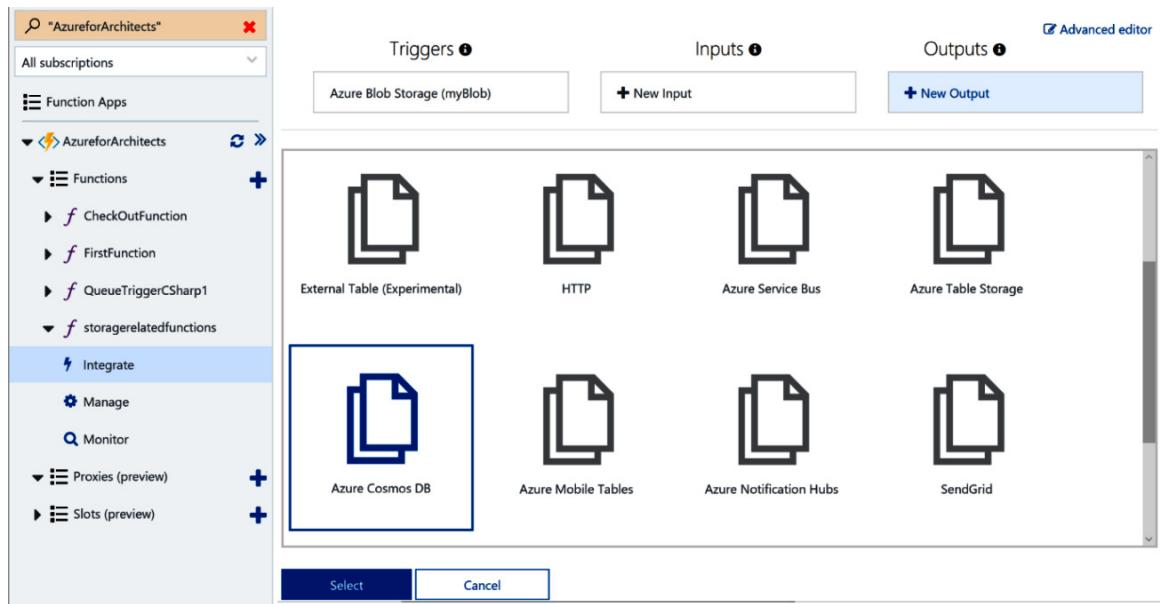


Figura 10.21: associando saída ao Azure Cosmos DB

- Forneça os nomes apropriados para o banco de dados e a coleção (marque a caixa de seleção para criar a coleção, se ela não existir), clique no botão **Novo** para selecionar nosso recém-criado Azure Cosmos DB e deixe o nome do parâmetro como **outputDocument**:

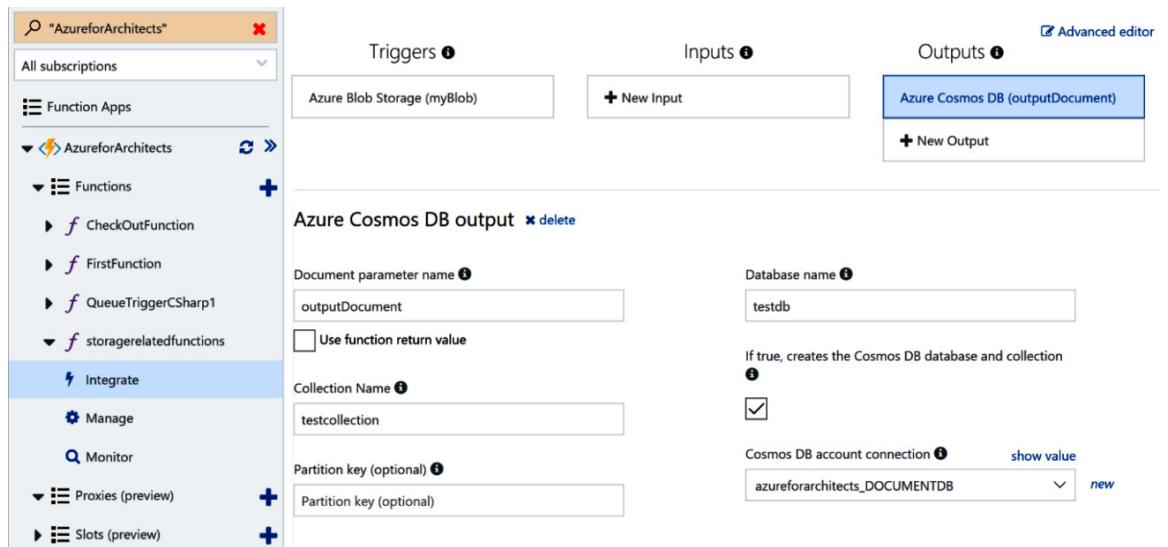
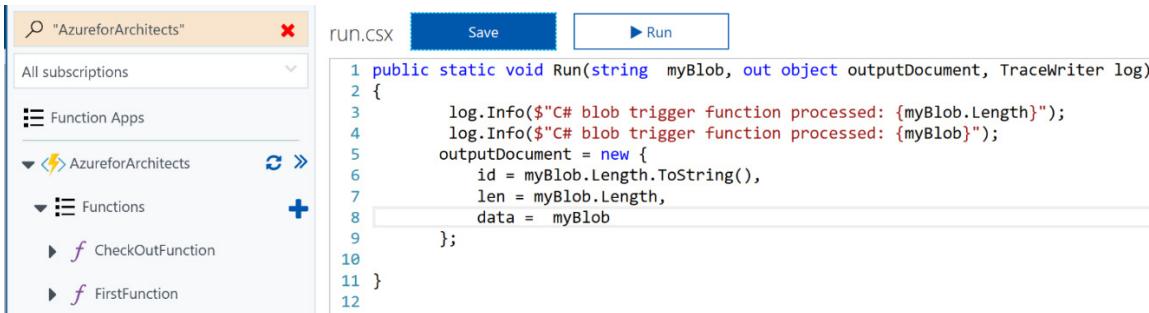


Figura 10.22: Azure Cosmos DB recém-criado

4. Modifique a função, conforme mostrado na Figura 10.23:



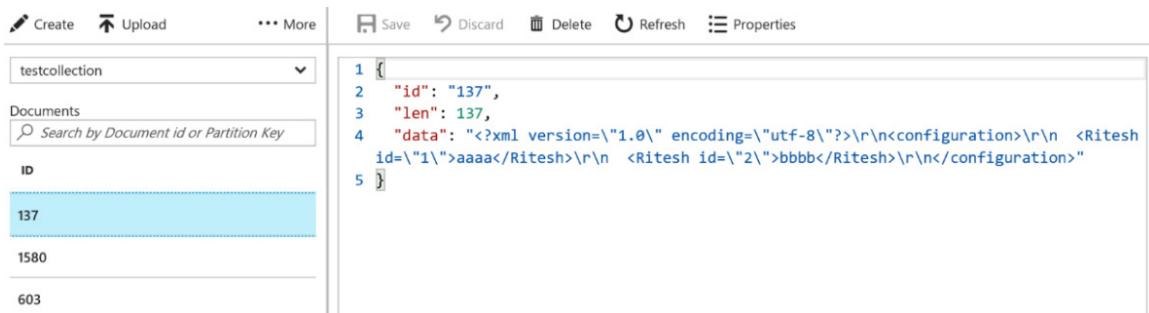
```

    public static void Run(string myBlob, out object outputDocument, TraceWriter log)
    {
        log.Info($"C# blob trigger function processed: {myBlob.Length}");
        log.Info($"C# blob trigger function processed: {myBlob}");
        outputDocument = new {
            id = myBlob.Length.ToString(),
            len = myBlob.Length,
            data = myBlob
        };
    }
}

```

Figura 10.23: modificando a função

5. Agora, o upload de um arquivo novo para a coleção de pedidos na conta de armazenamento do Azure executará uma função que gravará na coleção do Azure Cosmos DB. Outra função pode ser escrita com a recém-criada conta do Azure Cosmos DB como uma associação de gatilho. Ela fornecerá o tamanho dos arquivos, e a função poderá agir em relação a isso. Isso é mostrado aqui:



```

1 {
2   "id": "137",
3   "len": 137,
4   "data": "<?xml version=\\"1.0\\" encoding=\\"utf-8\\"?>\r\n<configuration>\r\n  <Ritesh>\r\n    id=\\"1\\">aaaa</Ritesh>\r\n  <Ritesh id=\\"2\\">bbbb</Ritesh>\r\n</configuration>"
5 }

```

Figura 10.24: gravando uma função de associação de gatilho

Esta seção abordou como a saída de uma função pode ser usada para obter dados para a próxima função. Na próxima seção, você aprenderá a habilitar os eventos sem servidor, compreendendo a Grade de Eventos do Azure.

## Grade de Eventos do Azure

A Grade de Eventos do Azure é um serviço relativamente novo. Também foi chamado de uma plataforma de eventos sem servidor. Ele ajuda na criação de aplicações baseadas em eventos (também conhecido como **design controlado por eventos**). É importante entender o que são eventos e como lidamos com eles antes da Grade de Eventos. Um evento é algo que aconteceu, ou seja, uma atividade que mudou o estado de um assunto. Quando um assunto passa por uma alteração em seu estado, geralmente gera um evento.

Os eventos normalmente seguem o padrão de publicação/assinatura (também conhecido popularmente como **padrão de publicação/assinatura**), no qual um assunto gera um evento devido à sua alteração de estado, e esse evento pode ser inscrito por várias stakeholders, também conhecidas como **assinantes**. O trabalho do evento é notificar os assinantes de tais mudanças e também fornecê-los com dados como parte de seu contexto. Os assinantes podem tomar qualquer ação que considerem necessárias, o que varia de assinante para assinante.

Antes da Grade de Eventos, não havia nenhum serviço que poderia ser descrito como uma plataforma de eventos em tempo real. Havia serviços separados, e cada um forneceu seu próprio mecanismo para manipular eventos.

Por exemplo, o Log Analytics, também conhecido como **Operations Management Suite (OMS)**, fornece uma infraestrutura para capturar logs de ambiente e telemetria em que os alertas podem ser gerados. Esses alertas podem ser usados para executar um runbook, um webhook ou uma função. Isso é próximo ao tempo real, mas eles não são completamente tempo real. Além disso, foi bastante complicado capturar logs individuais e agir sobre eles. Da mesma forma, há o Application Insights, que fornece recursos semelhantes para o Log Analytics, mas para aplicações.

Há outros logs, como os logs de atividades e logs de diagnóstico, mas, novamente, eles dependem de princípios semelhantes aos de outros recursos relacionados ao log. As soluções são implantadas em vários grupos de recursos em várias regiões, e os eventos gerados de qualquer um deles devem estar disponíveis para os recursos implantados em outro lugar.

A Grade de Eventos remove todas as barreiras e, como resultado, os eventos podem ser gerados pela maioria dos recursos (cada vez mais estão se tornando disponíveis) e até mesmo eventos personalizados podem ser gerados. Esses eventos podem então ser assinados por qualquer recurso, em qualquer região e em qualquer grupo de recursos dentro da assinatura.

A Grade de Eventos já está estabelecida como parte da infraestrutura do Azure, junto com data centers e redes. Os eventos gerados em uma região podem ser facilmente assinados por recursos em outras regiões, e uma vez que essas redes estejam conectadas, será extremamente eficiente para a entrega de eventos aos assinantes.

## Grade de Eventos

A Grade de Eventos permite criar aplicações com arquitetura baseada em eventos. Há editores de eventos e há consumidores de eventos. No entanto, pode haver vários assinantes para o mesmo evento.

O editor de um evento pode ser um recurso do Azure, como Armazenamento de Blobs, hubs da **Internet das Coisas (IoT)** e muitos outros. Esses editores também são conhecidos como origens de eventos. Esses editores usam tópicos do Azure prontos para uso para enviar seus eventos para a Grade de Eventos. Não é necessário configurar o recurso ou o tópico. Os eventos gerados por recursos do Azure já estão usando internamente tópicos para enviar seus eventos para a Grade de Eventos. Assim que o evento atingir a grade, poderá ser consumido pelos assinantes.

Os assinantes, ou consumidores, são recursos interessados em eventos e desejam executar uma ação com base nesses eventos. Esses assinantes fornecem um manipulador de eventos quando eles assinam o tópico. Os manipuladores de eventos podem ser funções do Azure, webhooks personalizados, aplicativos lógicos ou outros recursos. As origens de eventos e os assinantes que executam manipuladores de eventos são mostrados na Figura 10.25:

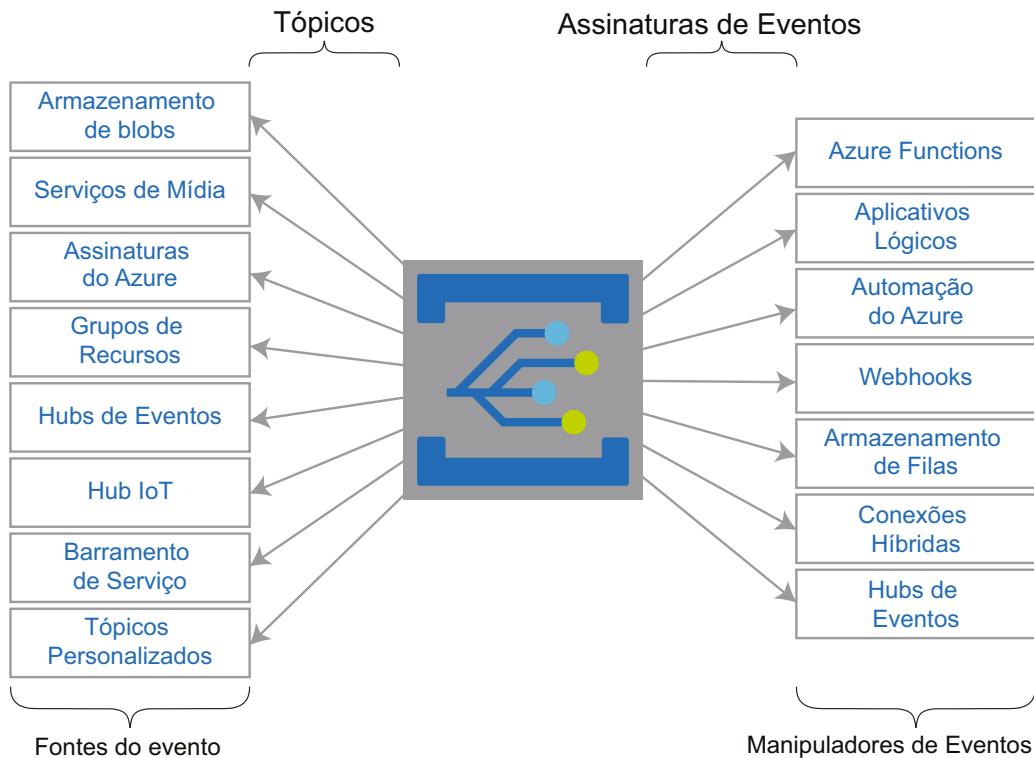


Figura 10.25: a arquitetura da Grade de Eventos

Quando um evento atinge um tópico, vários manipuladores de eventos podem ser executados simultaneamente, cada um executando sua própria ação.

Também é possível criar um evento personalizado e enviar um tópico personalizado para a Grade de Eventos. A Grade de Eventos fornece recursos para a criação de tópicos personalizados, e esses tópicos são anexados automaticamente à Grade de Eventos. Esses tópicos conhecem o armazenamento da Grade de Eventos e enviam automaticamente suas mensagens para ele. Os tópicos personalizados têm duas propriedades importantes, da seguinte maneira:

- **Um ponto de extremidade:** é o ponto de extremidade do tópico. Os editores e as origens de eventos usam esse ponto de extremidade para enviar e publicar seus eventos na Grade de Eventos. Em outras palavras, os tópicos são reconhecidos usando seus pontos de extremidade.

- **Chaves:** os tópicos personalizados fornecem algumas chaves. Essas chaves permitem a segurança para o consumo do ponto de extremidade. Somente os editores com essas chaves podem enviar e publicar suas mensagens na Grade de Eventos.

Cada evento tem um tipo de evento e é reconhecido por ele. Por exemplo, o Armazenamento de Blobs fornece tipos de eventos, como **blobAdded** e **blobDeleted**. Os tópicos personalizados podem ser usados para enviar um evento personalizado definido, como um evento personalizado do tipo **KeyVaultSecretExpired**.

Por outro lado, os assinantes têm a capacidade de aceitar todas as mensagens ou apenas obter eventos com base em filtros. Esses filtros podem ser baseados no tipo de evento ou em outras propriedades dentro da carga do evento.

Cada evento tem pelo menos estas cinco propriedades:

- **id:** é o identificador exclusivo para o evento.
- **eventType:** é o tipo de evento.
- **eventTime:** é a data e hora em que o evento foi gerado.
- **subject:** é uma breve descrição do evento.
- **data:** é um objeto de dicionário e contém dados específicos de recursos ou quaisquer dados personalizados (para tópicos personalizados).

Atualmente, as funcionalidades da Grade de Eventos não estão disponíveis com todos os recursos. No entanto, o Azure está constantemente adicionando mais recursos com a funcionalidade de Grade de Eventos.

Para saber mais sobre os recursos que podem gerar eventos relacionados à Grade de Eventos e manipuladores que podem manipular esses eventos, acesse <https://docs.microsoft.com/azure/event-grid/overview>.

## Eventos de recursos

Nesta seção, as etapas a seguir são fornecidas para criar uma solução na qual os eventos que são gerados pelo Armazenamento de Blobs são publicados na Grade de Eventos e, por fim, roteados para uma função do Azure:

1. Faça logon no portal do Azure e em sua assinatura usando as credenciais adequadas e crie uma nova conta de armazenamento em um grupo de recursos existente ou novo. A conta de armazenamento deve ser **StorageV2** ou **Armazenamento de Blobs**. Conforme demonstrado na Figura 10.26, a Grade de Eventos não funcionará com **StorageV1**:

## Create storage account

Basics Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

### PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	<input type="text" value="RiteshSubscription"/>	▼
* Resource group	<input type="text" value="azureclitest"/>	▼
	<a href="#">Create new</a>	

### INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name <a href="#">?</a>	<input type="text" value="storageforeventgrid"/>	✓
* Location	<input type="text" value="West Europe"/>	▼
Performance <a href="#">?</a>	<input checked="" type="radio"/> Standard <input type="radio"/> Premium	
Account kind <a href="#">?</a>	<input type="text" value="StorageV2 (general purpose v2)"/>	▼
Replication <a href="#">?</a>	<input type="text" value="Read-access geo-redundant storage (RA-GRS)"/>	▼
Access tier (default) <a href="#">?</a>	<input type="radio"/> Cool <input checked="" type="radio"/> Hot	

[Review + create](#)

[Previous](#)

[Next : Advanced >](#)

Figura 10.26: criando uma nova conta de armazenamento

- Crie um novo aplicativo de função ou reutilize um existente para criar uma função do Azure. A função do Azure será hospedada no aplicativo de função.
- Crie uma nova função usando o modelo de **gatilho da Grade de Eventos do Azure**. Instale a extensão **Microsoft.Azure.WebJobs.Extensions.EventGrid** se ainda não estiver instalada, conforme mostrado na Figura 10.27:

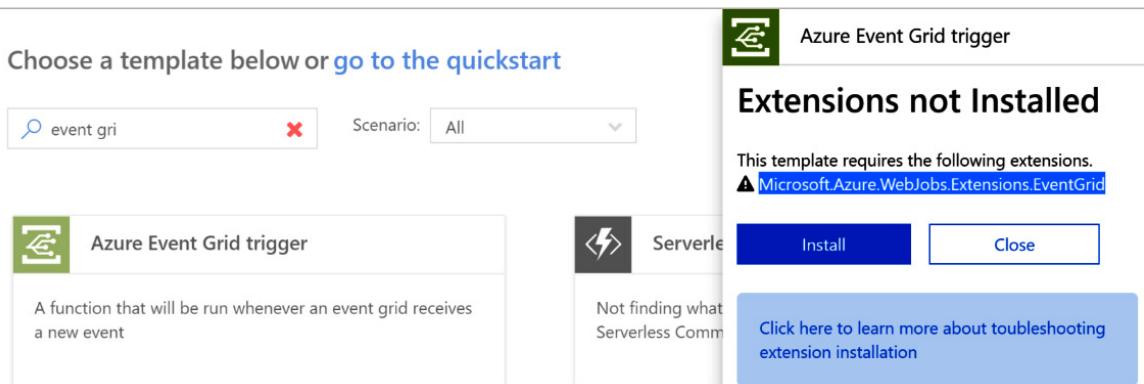


Figura 10.27: instalando extensões para um gatilho da Grade de Eventos do Azure

4. Chame a função de **StorageEventHandler** e crie-a. O seguinte código padrão gerado será usado como o manipulador de eventos:

```

run.csx      Save      Run      Add Event Grid subscription

1 #r "Microsoft.Azure.EventGrid"
2 using Microsoft.Azure.EventGrid.Models;
3
4 public static void Run(EventGridEvent eventGridEvent, ILogger log)
5 {
6     log.LogInformation(eventGridEvent.Data.ToString());
7 }
8

```

Figura 10.28: código do manipulador de eventos

A assinatura de eventos de armazenamento pode ser configurada na **interface do usuário (UI)** do Azure Functions clicando em **Adicionar assinatura da Grade de Eventos** ou na própria conta de armazenamento.

5. Clique no link **Adicionar assinatura de Grade de Eventos** na interface do usuário do Azure Functions para adicionar uma assinatura aos eventos gerados pela conta de armazenamento criada na etapa anterior. Forneça um nome significativo para a assinatura e escolha **Esquema de Eventos** seguido por **Esquema da Grade de Eventos**. Defina **Tipos de Tópico** como **Contas de Armazenamento**, defina uma **Assinatura** apropriada e o grupo de recursos que contém a conta de armazenamento:

The screenshot shows the 'Create Event Subscription' page in the Azure Functions portal. At the top, there's a breadcrumb navigation: Home > DurableFunctionDemoApp - StorageEventHandler > Create Event Subscription. Below the breadcrumb is a title 'Create Event Subscription' with a 'Event Grid' icon. There are three tabs: 'Basic' (selected), 'Filters', and 'Additional Features'. A note below the tabs says: 'Event Subscriptions listen for events emitted by the topic resource and send them to the endpoint resource.' with a 'Learn more' link.

**EVENT SUBSCRIPTION DETAILS**

Name	StorageEventSubscription	<input checked="" type="checkbox"/>
Event Schema	Event Grid Schema	<input type="button" value="▼"/>

**TOPIC DETAILS**

Pick a topic resource for which events should be generated and pushed. [Learn more](#)

Topic Types	Storage Accounts	<input type="button" value="▼"/>
Subscription	RiteshSubscription	<input type="button" value="▼"/>
Resource Group	azureclitest	<input type="button" value="▼"/>
Resource	someoddstoreacc1 storageforeventgrid someoddstoreacc	<input type="button" value="^"/>

**EVENT TYPES**

Pick which event types get pushed to your d

Subscribe to all event types

Figura 10.29: criando uma assinatura da Grade de Eventos

Verifique se a caixa de seleção **Assinar todos os tipos de evento** está marcada e clique no botão **Criar** (ele deverá ser habilitado assim que uma conta de armazenamento for selecionada).

6. Se agora navegamos para a conta de armazenamento no portal do Azure e clicamos no link **Eventos** no menu à esquerda, a assinatura da conta de armazenamento deverá estar visível:

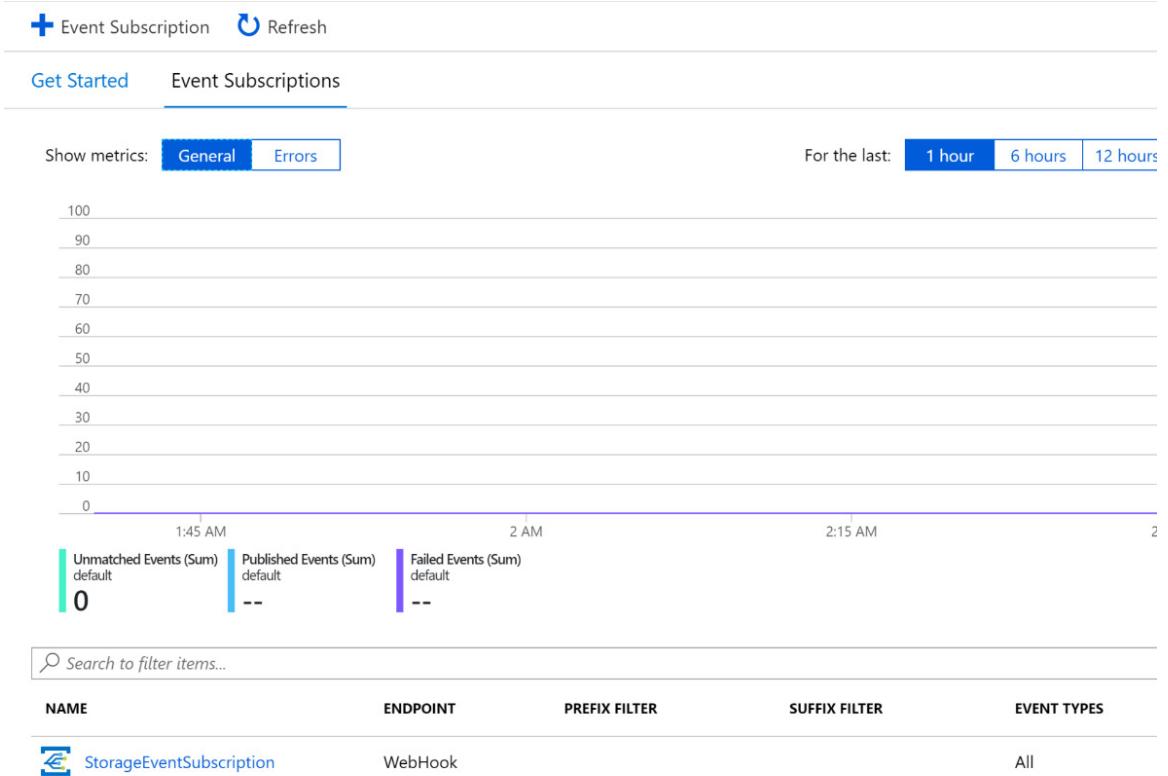


Figura 10.30: lista de assinaturas de eventos

7. Faça upload de um arquivo para o Armazenamento de Blobs depois de criar um contêiner, e a função do Azure deverá ser executada. A ação de upload acionará um novo evento do tipo **blobAdded**, e o enviará para o tópico da Grade de Eventos para contas de armazenamento. Conforme mostrado na Figura 10.31, a assinatura já está definida para obter todos os eventos deste tópico, e a função é executada como parte do manipulador de eventos:

The screenshot shows the Azure Functions developer portal interface. At the top, there are buttons for 'Save' and 'Run'. Below that is the code editor with the following C# code:

```

1 #r "Microsoft.Azure.EventGrid"
2 using Microsoft.Azure.EventGrid.Models;
3
4 public static void Run(EventGridEvent eventGridEvent, ILogger log)
5 {
6     log.LogInformation(eventGridEvent.Data.ToString());
7 }

```

Below the code editor is a section titled 'Logs' with a 'Console' tab selected. The logs pane displays the following output:

```

2019-01-10T07:48:40 welcome, you are now connected to log-streaming service.
2019-01-10T07:48:51.428 [Information] Executing 'Functions.StorageEventHandler' (Reason='EventGrid trigger fired at 2019-01-10T07:48:51.4287500+00:00', Id=5c4e2121-f903-40bb-a45a-e49e674348ec)
2019-01-10T07:48:51.429 [Information] [
  "api": "PutBlockList",
  "clientRequestId": "010e48eb-a88e-4a48-bec1-44da8e8b503c",
  "requestId": "a0217c1f-f01e-0032-53b8-a83345000000",
  "eTag": "0x8d676000FA07895",
  "contentType": "application/pdf",
  "contentLength": 126822,
  "blobType": "BlockBlob",
  "url": "https://storageforeventgrid.blob.core.windows.net/filecontainer/sup",
  "sequencer": "0000000000000000000000000000000042d8000000000002d532",
  "storageProperties": {
    ...
  }
]

```

At the bottom of the logs pane are buttons for 'Reconnect', 'Copy logs', 'Pause', 'Clear', and 'Expand'.

Figura 10.31: acionando um novo evento

Nesta seção, você aprendeu como os eventos gerados pelo Armazenamento de Blobs podem ser roteados para uma função do Azure. Na próxima seção, você aprenderá a aproveitar eventos personalizados.

## Eventos personalizados

Neste exemplo, em vez de usar recursos prontos para uso para gerar eventos, serão usados eventos personalizados. Usaremos o PowerShell para criar essa solução e reutilizar a mesma função do Azure que foi criada no último exercício como o manipulador:

1. Faça logon e conecte-se à assinatura do Azure à sua escolha usando o cmdlet **Login-AzAccount** e **Set-AzContext**.
2. A próxima etapa é criar um novo tópico de Grade de Eventos no Azure em um grupo de recursos. O cmdlet **New-AzEventGridTopic** é usado para criar um novo tópico:

```
New-AzEventGridTopic -ResourceGroupName CustomEventGridDemo -Name
"KeyVaultAssetsExpiry" -Location "West Europe"
```

3. Depois que o tópico for criado, sua URL de ponto de extremidade e a chave deverão ser recuperadas, já que são necessárias para enviar e publicar o evento para ele. Os cmdlets **Get-AzEventGridTopic** e **Get-AzEventGridTopicKey** são usados para recuperar esses valores. Observe que **Key1** é recuperada para se conectar ao ponto de extremidade:

```
$topicEndpoint = (Get-AzEventGridTopic -ResourceGroupName containers -Name KeyVaultAssetsExpiry).Endpoint
```

```
$keys = (Get-AzEventGridTopicKey -ResourceGroupName containers -Name KeyVaultAssetsExpiry).Key1
```

4. Uma nova tabela de hash é criada com todas as cinco propriedades de evento da Grade de Eventos importantes. Uma nova propriedade **id** é gerada para a ID, a propriedade **subject** é definida como **Expiração de Evento de cofre de chaves**, **eventType** é definido como **Certificate Expiry**, **eventTime** é definido com a hora atual e **data** contém informações sobre o certificado:

```
$eventgridDataMessage = @{
    id = [System.guid]::NewGuid()
    subject = "Key Vault Asset Expiry"
    eventType = "Certificate Expiry"
    eventTime = [System.DateTime]::UtcNow
    data = @{
        CertificateThumbprint = "sdfervdserwetsgfhgdg"
        ExpiryDate = "1/1/2019"
        Createdon = "1/1/2018"
    }
}
```

5. Como os dados da Grade de Eventos devem ser publicados na forma de uma matriz JSON, a carga é convertida na matriz JSON. Os colchetes "[ , ]" representam uma matriz JSON:

```
$finalBody = "[" + $(ConvertTo-Json $eventgridDataMessage) + "]"
```

6. O evento será publicado usando o protocolo HTTP e as informações de cabeçalho apropriadas devem ser adicionadas à solicitação. A solicitação é enviada usando o tipo de conteúdo application/JSON, e a chave pertencente ao tópico é atribuída ao cabeçalho **aeg-sas-key**. É obrigatório chamar o cabeçalho e o conjunto de chaves de **aeg-sas-key**:

```
$header = @{
    "contentType" = "application/json"
    "aeg-sas-key" = $keys}
```

7. Uma nova assinatura é criada para o tópico personalizado com um nome, o grupo de recursos que contém o tópico, o nome do tópico, o ponto de extremidade do webhook e o ponto de extremidade real que atua como o manipulador de eventos. Nesse caso, o manipulador de eventos é a função do Azure:

```
New-AzEventGridSubscription -TopicName KeyVaultAssetsExpiry
-EventSubscriptionName "customtopicsubscriptionautocar" -ResourceGroupName
CustomEventGridDemo -EndpointType webhook '
-Endpoint "https://durablefunctiondemoapp.
azurewebsites.net/runtime/webhooks/
EventGrid?functionName=StorageEventHandler&code=0aSw6sxvtFmafXHvt7i0w/
Dsb8o1M9RKKagzVchTUKwe9EIkz14mCg=="
-Verbose
```

A URL da função do Azure está disponível na guia **Integrar**, conforme mostrado na Figura 10.31:

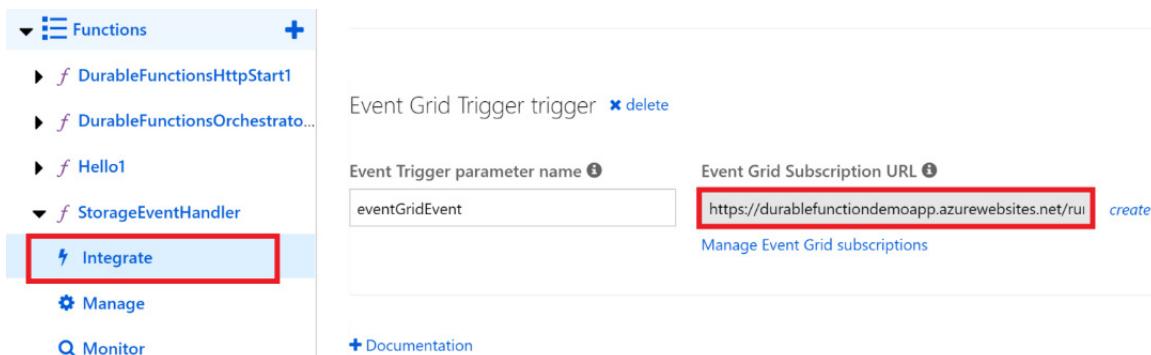


Figura 10.32: URL da assinatura da Grade de Eventos na guia Integrar

8. Até agora, o assinante (manipulador de eventos) e o editor foram configurados. A próxima etapa é enviar e publicar um evento para o tópico personalizado. Os dados do evento já foram criados na etapa anterior e, usando o cmdlet **Invoke-WebRequest**, a solicitação é enviada ao ponto de extremidade junto com o corpo e o cabeçalho:

```
Invoke-WebRequest -Uri $topicEndpoint -Body $finalBody -Headers $header
-Method Post
```

A chamada da API acionará o evento, e a Grade de Eventos enviará uma mensagem para o ponto de extremidade que configuramos, que é o aplicativo de função. Com essa atividade, estamos terminando este capítulo.

## Resumo

A evolução das funções dos métodos tradicionais levou à criação da arquitetura de baixa associação, de evolução independente, autossuficiente sem servidor que era apenas um conceito antigamente. As funções são uma unidade de implantação e fornecem um ambiente que não precisa ser gerenciado pelo usuário. Ele só precisa se preocupar com o código escrito para a funcionalidade. O Azure fornece uma plataforma madura para hospedagem de funções e as integra perfeitamente, com base em eventos ou sob demanda. Quase todos os recursos no Azure podem participar de uma arquitetura composta de funções do Azure. O futuro é das funções, pois cada vez mais organizações querem se afastar do gerenciamento de infraestruturas e plataformas. Elas desejam descarregar isso nos provedores de nuvem. O Azure Functions é um recurso essencial a ser dominado por todos os arquitetos que lidam com o Azure.

Este capítulo explicou em detalhes o Azure Functions, Funções como Serviço, o Durable Functions e a Grade de Eventos. O próximo capítulo se concentrará em Aplicativos Lógicos do Azure, e vamos criar uma solução completa que combina vários serviços sem servidor, juntamente com outros serviços do Azure, como o Azure Key Vault e a Automação do Azure.



# 11

# Soluções do Azure usando Aplicativos Lógicos do Azure, Grade de Eventos e Functions

Este capítulo continua a partir do capítulo anterior e mostrará mais detalhes sobre os serviços sem servidor disponíveis no Azure. No capítulo anterior, você conheceu detalhadamente o Azure Functions, Funções como Serviço, Durable Functions e Grade de Eventos. De agora em diante, este capítulo se concentrará na compreensão dos Aplicativos Lógicos e, em seguida, passará à criação de uma solução completa sem servidor de ponta a ponta que combina vários serviços sem servidor, entre outros, como Key Vault e Automação do Azure.

Neste capítulo, exploraremos ainda mais os serviços do Azure abordando os seguintes tópicos:

- Aplicativos Lógicos do Azure
- Criação de uma solução de ponta a ponta usando tecnologias sem servidor

## Aplicativos Lógicos do Azure

Aplicativos Lógicos é uma oferta de fluxo de trabalho sem servidor do Azure. Ele tem todos os recursos de tecnologias sem servidor, como custeio baseado no consumo e escalabilidade ilimitada. O Aplicativos Lógicos nos ajuda a criar um processo de negócios e uma solução de fluxo de trabalho com facilidade usando o portal do Azure. Ele fornece uma interface do usuário de arrastar e soltar para criar e configurar fluxos de trabalho.

O uso de Aplicativos Lógicos é a maneira preferencial de integrar serviços e dados, criar projetos de negócios e criar um fluxo completo de lógica. Há diversos conceitos importantes que devem ser entendidos antes de criar um Aplicativo Lógico.

### Atividades

Uma atividade é uma única unidade de trabalho. Exemplos de atividades incluem a conversão de XML em JSON, a leitura de blobs do Armazenamento do Azure e a gravação em uma coleção de documentos do Cosmos DB. O Aplicativos Lógicos fornece uma definição de fluxo de trabalho que consiste em várias atividades relacionadas em uma sequência. Há dois tipos de atividade no Aplicativos Lógicos:

- **Gatilho:** um gatilho refere-se à iniciação de uma atividade. Todos os Aplicativos Lógicos têm um único gatilho que forma a primeira atividade. É o gatilho que cria uma instância do aplicativo lógico e inicia a execução. Exemplos de gatilhos são a chegada de mensagens da Grade de Eventos, um email, uma solicitação HTTP ou um agendamento.
- **Ações:** qualquer atividade que não seja um gatilho é uma atividade de etapa, e cada uma delas é responsável por executar uma tarefa. As etapas estão conectadas entre si em um fluxo de trabalho. Cada etapa terá uma ação que precisa ser concluída antes de passar para a próxima etapa.

### Conectores

Os conectores são recursos do Azure que ajudam a conectar um Aplicativo Lógico a serviços externos. Esses serviços podem estar na nuvem ou na infraestrutura local. Por exemplo, há um conector para conectar Aplicativos Lógicos à Grade de Eventos. Da mesma forma, há outro conector para se conectar ao Office 365 Exchange. Quase todos os tipos de conectores estão disponíveis em Aplicativos Lógicos, e eles podem ser usados para se conectar a serviços. Os conectores contêm informações de conexão e também lógica para conectar a serviços externos usando essas informações de conexão.

A lista inteira de conectores está disponível em <https://docs.microsoft.com/connectors>.

Agora que você já conhece os conectores, precisa entender como eles podem ser alinhados passo a passo para fazer com que o fluxo de trabalho funcione conforme esperado. Na próxima seção, vamos nos concentrar no funcionamento de um Aplicativo Lógico.

## O funcionamento de um Aplicativo Lógico

Vamos criar um fluxo de trabalho de Aplicativos Lógicos que seja acionado quando uma conta de email recebe um email. Ele responde ao remetente com um email padrão e executa a análise de sentimento sobre o conteúdo do email. Para análise de sentimento, o recurso de Análise de Texto de Serviços Cognitivos deve ser provisionado antes da criação do Aplicativo Lógico:

1. Navegue até o portal do Azure, faça logon na sua conta e crie um recurso de **Análise de Texto** em um grupo de recursos. A Análise de Texto faz parte dos Serviços Cognitivos e tem recursos como análise de sentimento, extração de frases-chave e detecção de linguagem. Você pode encontrar o serviço no portal do Azure, como mostrado na Figura 11.1:

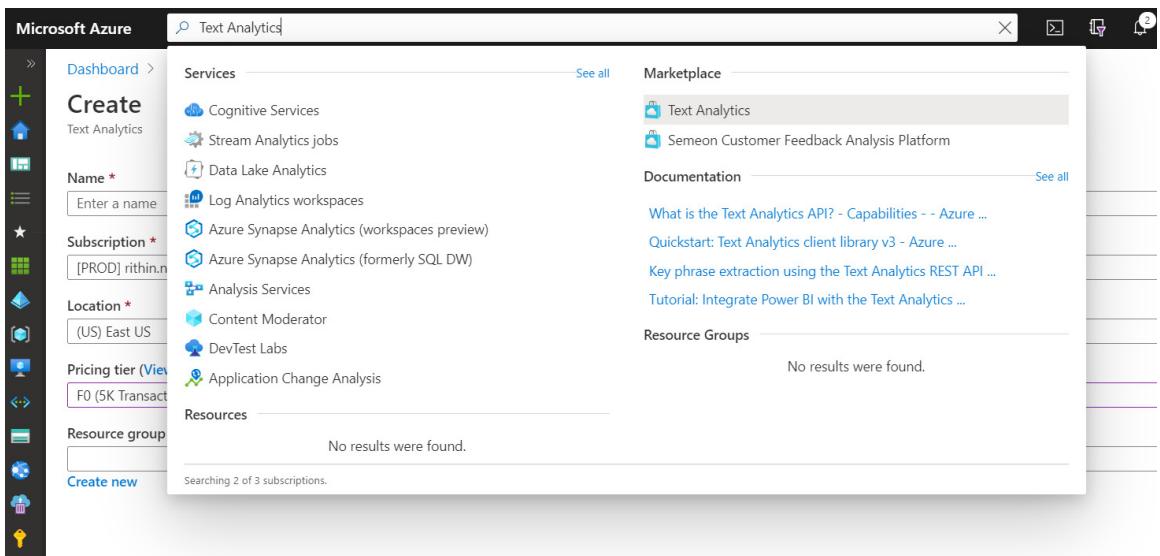


Figura 11.1: navegar até o serviço de Análise de Texto a partir do portal do Azure

2. Informe os valores **Nome**, **Local**, **Assinatura**, **Grupos de recursos** e **Camada de preços**. Utilizaremos a camada gratuita (camada F0) deste serviço para esta demonstração.
3. Depois que o recurso for provisionado, navegue até a página **Visão geral** e copie o URL do ponto de extremidade. Guarde-o em um local temporário. Esse valor será necessário na configuração do Aplicativo Lógico.

4. Navegue até a página **Chaves**, copie o valor de **Chave 1** e armazene-o em um local temporário. Esse valor será necessário na configuração do Aplicativo Lógico.
5. A próxima etapa é criar um Aplicativo Lógico. Para criar um Aplicativo Lógico, navegue até o grupo de recursos no portal do Azure no qual ele deverá ser criado. Procure o Aplicativo Lógico e crie-o fornecendo os valores **Nome**, **Local**, **Grupos de recursos** e **Assinatura**.
6. Depois que o Aplicativo Lógico tiver sido criado, navegue até o recurso, clique em **Designer de Aplicativo Lógico** no menu à esquerda e, em seguida, selecione o modelo **Quando um novo email é recebido em Outlook.com** para criar um novo fluxo de trabalho. O modelo fornece um início, adicionando cópias de gatilhos e atividades. Isso adicionará um gatilho do Office 365 Outlook automaticamente ao fluxo de trabalho.
7. Clique no botão **Entrar** no gatilho; ele abrirá uma nova janela do Internet Explorer. Em seguida, entre em sua conta. Após a conexão bem-sucedida, um novo conector de email do Office 365 será criado com as informações de conexão para a conta.
8. Clique no botão **Continuar** e configure o gatilho com uma frequência de sondagem de 3 minutos, conforme mostrado na Figura 11.2:

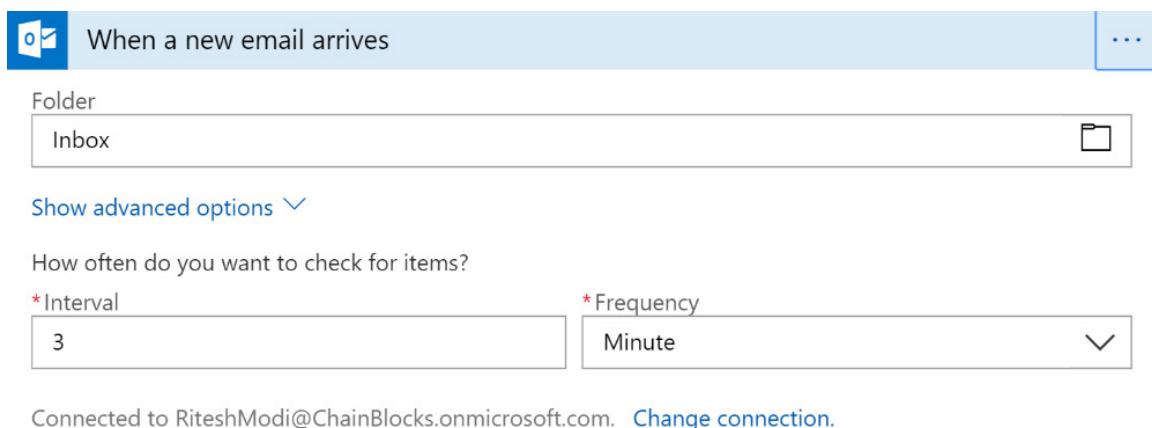


Figura 11.2: configurando o gatilho com uma frequência de enquete de 3 minutos

9. Clique na **Próxima etapa** para adicionar outra ação e digite a **variável** de palavra-chave na barra de pesquisa. Em seguida, selecione a ação **Iniciar variável**, conforme demonstrado na Figura 11.3:

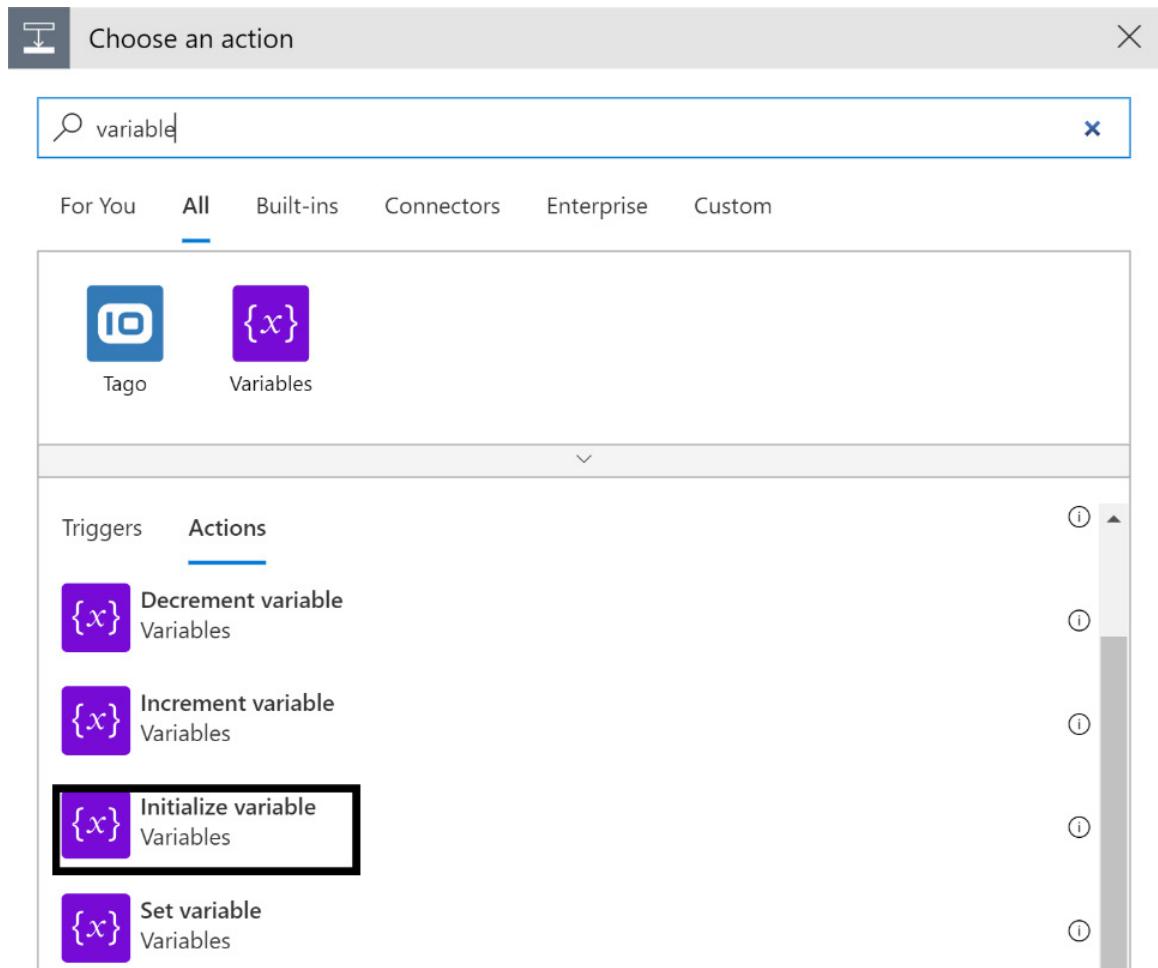


Figura 11.3: adicionando a ação Iniciar variável

10. Depois, configure a ação de variável. Quando a caixa **Valor** for clicada, aparecerá uma janela pop-up mostrando **Conteúdo dinâmico** e **Expressão**. O Conteúdo dinâmico refere-se a propriedades que estão disponíveis para a ação atual e são preenchidas com valores de tempo de execução de ações anteriores e gatilhos. As variáveis ajudam a manter os fluxos de trabalho genéricos. Nessa janela, selecione **Corpo** em **Conteúdo dinâmico**:

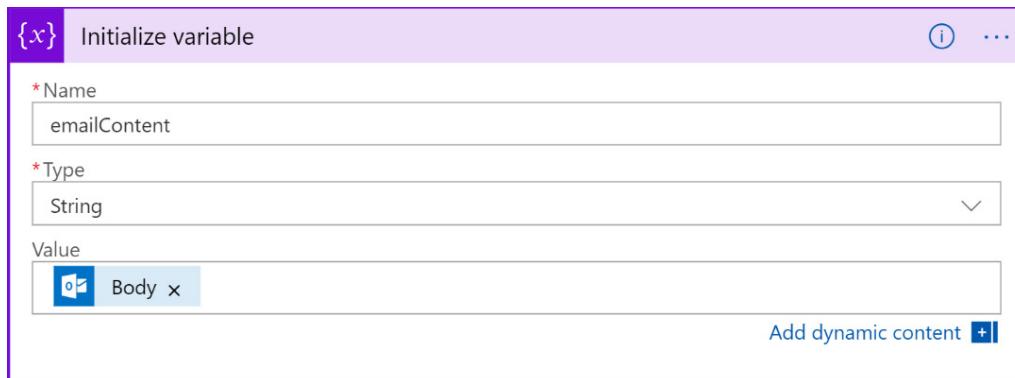


Figura 11.4: configurando a ação variável

11. Para incluir outra ação, clique em **Adicionar etapa**, digite **Outlook** na barra de pesquisa e selecione a ação **Responder ao email**:

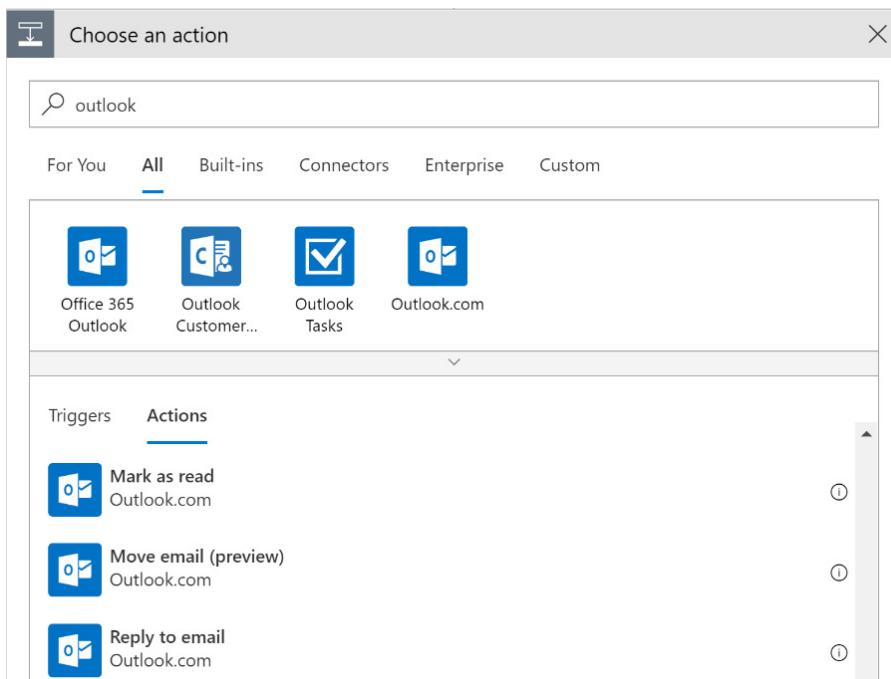


Figura 11.5: adicionando a resposta à ação de email

12. Configure a nova ação. Verifique se **Id da Mensagem** está definida com o conteúdo dinâmico, **Id da Mensagem** e, em seguida, digite a resposta que gostaria de enviar ao destinatário na caixa **Comentário**:

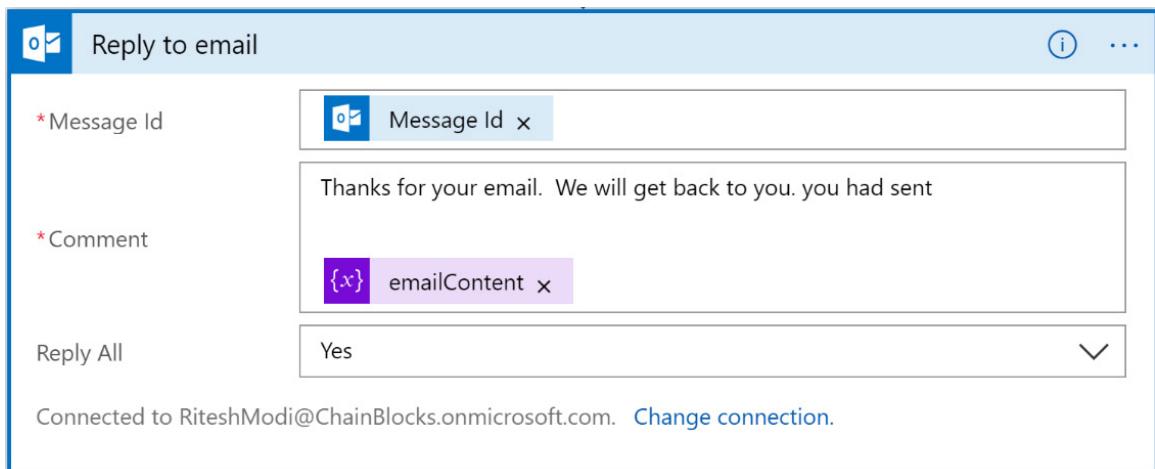


Figura 11.6: configurando a resposta à ação de email

13. Adicione outra ação, digite **análise de texto** na barra de pesquisa e, em seguida, selecione **Detectar Sentimento (visualização)**:

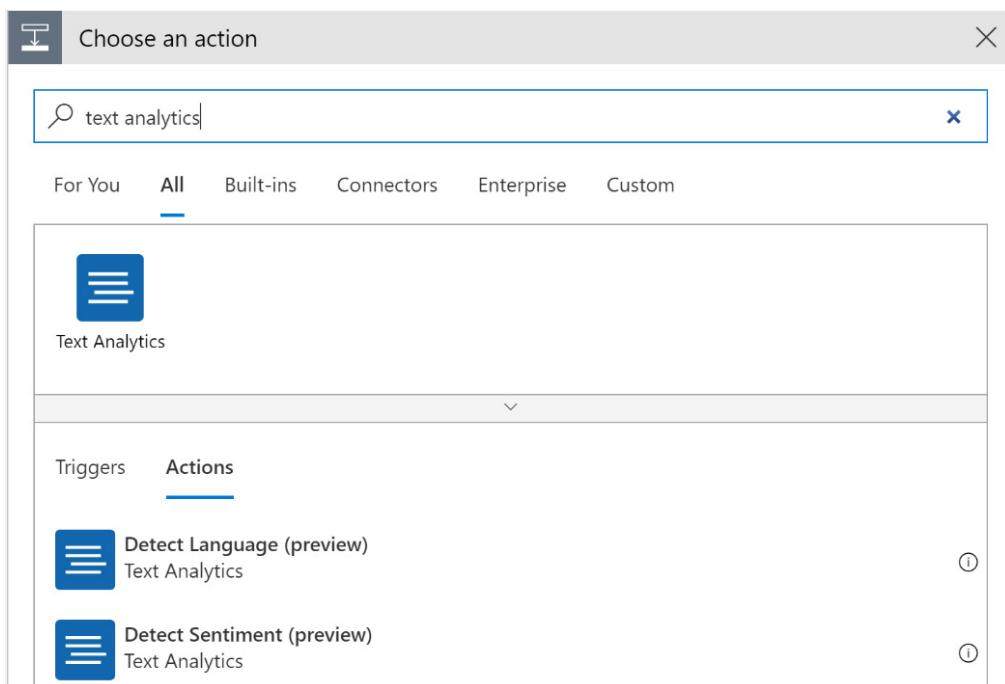
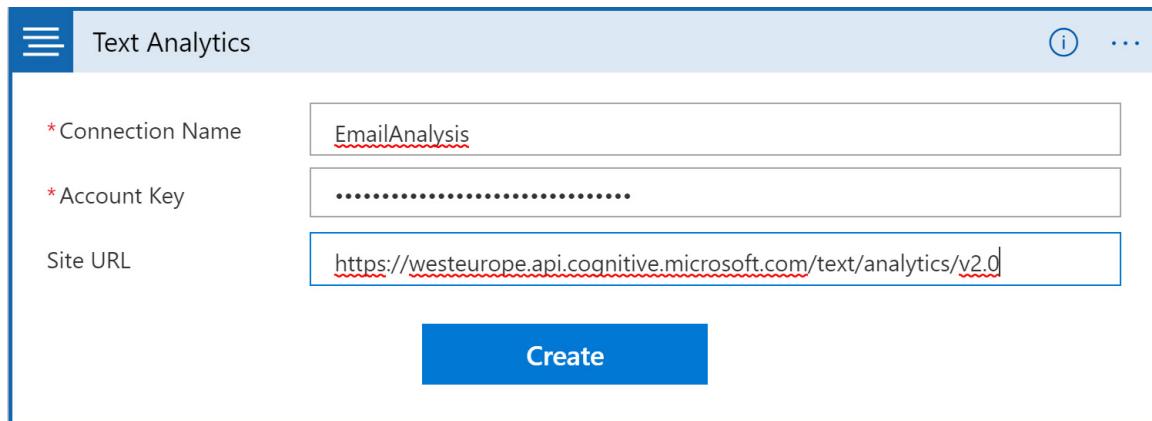


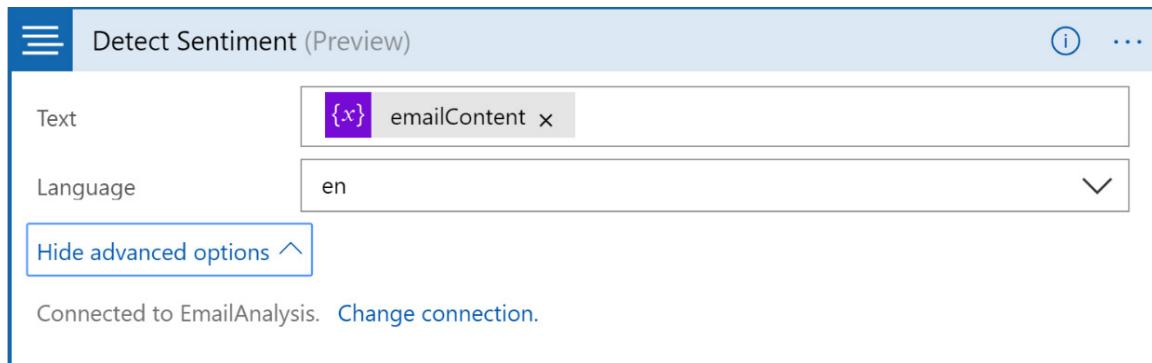
Figura 11.7: adicionando a ação Detectar Sentimento (visualização)

14. Configure a ação de sentimento, conforme mostrado na *Figura 11.8* – os valores de ponto de extremidade e chave devem ser usados aqui. Agora clique no botão **Criar**, como demonstrado na *Figura 11.8*:



**Figura 11.8:** configurando a ação Detectar Sentimento (visualização)

15. Forneça o texto à ação adicionando conteúdo dinâmico e selecionando a variável criada anteriormente, **emailContent**. Em seguida, clique em **Mostrar opções avançadas** e selecione **pt** para **Idioma**:



**Figura 11.9:** selecionando a linguagem para a ação de sentimento

16. Em seguida, adicione uma nova ação selecionando **Outlook**. Em seguida, selecione **Enviar um email**. Essa ação envia ao destinatário original o conteúdo do email com a pontuação de sentimento em seu assunto. Ele deve ser configurado como mostrado na *Figura 11.10*. Se a pontuação não estiver visível na janela de conteúdo dinâmico, clique no link **Ver mais** ao lado dele:

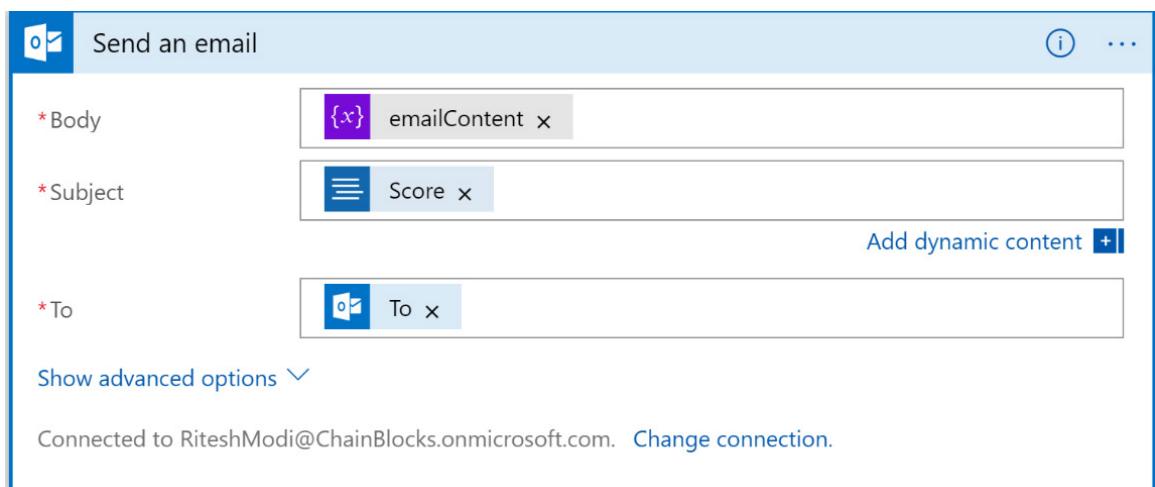


Figura 11.10: adicionando a ação Enviar um email

17. Salve o Aplicativo Lógico, navegue de volta para a página de visão geral e clique no **gatilho Executar**. O gatilho verificará novos emails a cada 3 minutos, responderá aos remetentes, executará a análise de sentimento e enviará um email para o destinatário original. Um email de exemplo com conotações negativas é enviado para a ID de email fornecida:

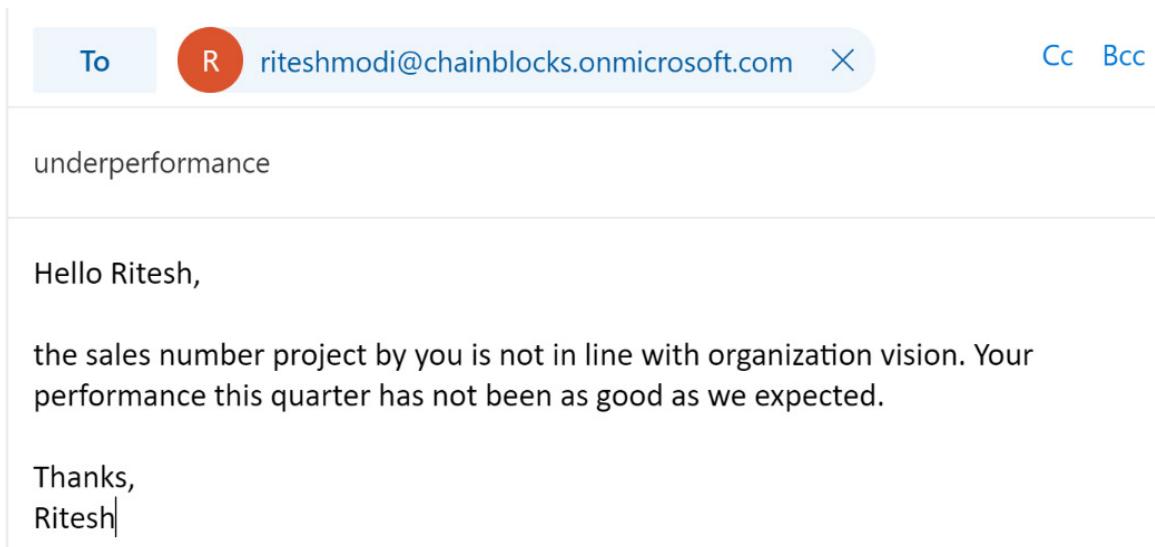


Figura 11.11: exemplo de email

18. Depois de alguns segundos, o Aplicativo Lógico é executado e o remetente recebe a seguinte resposta:

The screenshot shows an email interface. At the top, it says "underperformance". Below that, there's a green circular profile picture with the letters "RM". To the right of the picture, the recipient's name is listed as "Ritesh Modi <RiteshModi@ChainBlock.s.onmicrosoft.com>" and the timestamp "Thu 1/10/2019, 7:33 PM". There are three blue dots on the right side of the message body. The message content is as follows:

Thanks for your email. We will get back to you. you had sent  
Hello Ritesh,

the sales number project by you is not in line with organization vision.  
Your performance this quarter has not been as good as we expected.

Thanks,  
Ritesh

...

Figura 11.12: enviar resposta por email ao remetente original

19. O destinatário original recebe um email com a pontuação de sentimento e o texto de email original, conforme mostrado na Figura 11.13:

0.731263041496277

The screenshot shows an email interface. On the left, there's a green circular profile picture with the letters "RM". Next to it, the recipient's name is "Ritesh Modi" and the timestamp "Today, 7:33 PM". Below that, the name "Ritesh Modi" appears again with a dropdown arrow. A gray box contains the text "This message was sent with low importance." Below this, the original message content is shown in HTML format:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta content="text/html; charset=iso-8859-1">
<style type="text/css" style="display:none">
<!--
p
{margin-top:0;
```

Figura 11.13: visão HTML da mensagem de email

Na atividade, conseguimos entender o funcionamento de um Aplicativo Lógico. O aplicativo foi acionado quando um email foi recebido na caixa de entrada do usuário, e o processo seguiu a sequência de etapas apresentadas no Aplicativo Lógico. Na próxima seção, você aprenderá a criar uma solução de ponta a ponta usando tecnologias sem servidor.

## Criação de uma solução de ponta a ponta usando tecnologias sem servidor

Nesta seção, criaremos uma solução de ponta a ponta compreendendo tecnologias sem servidor que discutimos nas seções anteriores. O exemplo a seguir dará a você uma ideia de como os fluxos de trabalho podem ser implementados de forma inteligente para evitar sobrecarga de gerenciamento. Na próxima atividade, criaremos um fluxo de trabalho para notificar os usuários quando as chaves, os segredos e os certificados forem armazenados no Azure Key Vault. Vamos tomar isso como uma declaração de problema, descobrir uma solução, arquitetar a solução e implementá-la.

### A declaração do problema

O problema que vamos resolver aqui é que os usuários e organizações não são notificados sobre a expiração dos segredos em seu cofre de chaves, e as aplicações param de funcionar quando expiram. Os usuários estão reclamando que o Azure não fornece a infraestrutura para monitorar segredos, chaves e certificados do Key Vault.

### Solução

A solução para esse problema é combinar vários serviços do Azure e integrá-los para que os usuários possam ser notificados proativamente sobre a expiração de segredos. A solução enviará notificações usando dois canais – email e SMS.

Os serviços do Azure usados para criar essa solução incluem o seguinte:

- Azure Key Vault
- **Azure Active Directory (Azure AD)**
- Grade de Eventos do Azure
- Automação do Azure
- Aplicativos Lógicos
- Azure Functions
- SendGrid
- SMS do Twilio

Agora que você conhece os serviços que serão usados como parte da solução, vamos avançar e criar uma arquitetura para essa solução.

## Arquitetura

Na seção anterior, exploramos a lista de serviços que serão usados na solução. Se quisermos implementar a solução, os serviços deverão ser definidos na ordem apropriada. A arquitetura nos ajudará a desenvolver o fluxo de trabalho e a chegar mais perto da solução.

A arquitetura da solução compreende vários serviços, conforme mostrado na Figura 11.14:



Figura 11.14: arquitetura de solução

Vamos passar por cada um desses serviços e entender seus papéis e a funcionalidade que eles fornecem na solução geral.

## Automação do Azure

A Automação do Azure fornece runbooks e estes runbooks podem ser usados para executar a lógica utilizando o PowerShell, Python e outras linguagens de script. Os scripts podem ser executados na infraestrutura local ou na nuvem, o que proporciona infraestrutura e instalações avançadas para criar scripts. Esses tipos de script são conhecidos como **runbooks**. Normalmente, os runbooks implementam um cenário como interromper ou iniciar uma máquina virtual, ou criar e configurar contas de armazenamento. É muito fácil conectar-se ao ambiente do Azure desde runbooks com a ajuda de ativos, como variáveis, certificados e conexões.

Na solução atual, queremos conectar-se ao Azure Key Vault, ler todos os segredos e chaves armazenados nele e buscar suas datas de expiração. Essas datas de expiração devem ser comparadas com a data de hoje e, se a data de expiração estiver dentro de um mês, o runbook deve disparar um evento personalizado na Grade de Eventos usando um tópico personalizado de Grade de Eventos.

Um runbook da Automação do Azure utilizando um script do PowerShell será implementado para conseguir isso. Junto com o runbook, um agendador também será criado que executará o runbook uma vez por dia à meia-noite.

## Um tópico personalizado de Grade de Eventos do Azure

Depois que o runbook identificar que um segredo ou chave vai expirar dentro de um mês, ele gerará um novo evento personalizado e o publicará no tópico personalizado criado especificamente para essa finalidade. Novamente, veremos os detalhes da implementação na próxima seção.

## Aplicativos Lógicos do Azure

Um Aplicativo Lógico é um serviço sem servidor que fornece recursos de fluxo de trabalho. Nosso aplicativo lógico será configurado para ser disparado quando um evento for publicado no tópico de Grade de Eventos personalizado. Depois de disparado, ele invocará o fluxo de trabalho e executará todas as atividades contidas nele, uma após a outra. Geralmente, há várias atividades, mas para a finalidade deste exemplo, invocaremos uma função do Azure que enviará mensagens de email e SMS. Em uma implementação completa, essas funções de notificação devem ser implementadas separadamente em funções separadas do Azure.

## Azure Functions

O Azure Functions é usado para notificar os usuários e as stakeholders sobre a expiração de segredos e chaves usando email e SMS. SendGrid é usado para enviar emails, enquanto o Twilio é usado para enviar mensagens SMS do Azure Functions.

Na próxima seção, vamos dar uma olhada nos pré-requisitos antes de implementar a solução.

## Pré-requisitos

No mínimo, você precisará de uma assinatura do Azure com direitos de colaborador. Como estamos apenas implantando serviços no Azure e não há serviços externos implantados, a assinatura é o único pré-requisito. Vamos continuar e implementar a solução.

## Implementação

Um Key Vault já deve existir. Caso contrário, ele deverá ser criado.

Esta etapa deverá ser seguida se uma nova instância do Azure Key Vault precisar ser provisionada. O Azure permite provisionar recursos de várias maneiras. Entre eles, destacam-se o Azure PowerShell e a CLI do Azure. A CLI do Azure é uma interface de linha de comando que funciona em várias plataformas. A primeira tarefa será provisionar um cofre de chaves no Azure. Nesta implementação, utilizaremos o Azure PowerShell para provisionar o cofre de chaves.

Antes que o Azure PowerShell possa ser usado para criar um cofre de chaves, é importante fazer logon no Azure para que os comandos subsequentes possam ser executados com êxito para criar o cofre de chaves.

## Etapa 1: provisionar uma instância do Azure Key Vault

A primeira etapa é preparar o ambiente para a amostra. Isso envolve o logon no portal do Azure, selecionando uma assinatura apropriada e, em seguida, criando um novo grupo de recursos do Azure e um novo recurso do Azure Key Vault:

1. Execute o comando **Connect-AzAccount** para fazer logon no Azure. Ele solicitará credenciais em uma nova janela.
2. Após um logon bem-sucedido, se houver várias assinaturas disponíveis para a ID de logon fornecida, elas serão todas listadas. É importante selecionar uma assinatura apropriada – isso pode ser feito executando o cmdlet **Set-AzContext**:

```
Set-AzContext -SubscriptionId xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

3. Crie um novo grupo de recursos no seu local preferencial. Nesse caso, o nome do grupo de recursos é **IntegrationDemo** e é criado na região **Europa Ocidental**:

```
New-AzResourceGroup -Name IntegrationDemo -Location "West Europe" -Verbose
```

4. Crie um novo recurso do Azure Key Vault – o nome do cofre, nesse caso, é **keyvaultbook**, e ele está habilitado para implantação, implantação de modelo, criptografia de disco, exclusão suave e proteção contra remoção:

```
New-AzKeyVault -Name keyvaultbook -ResourceGroupName  
IntegrationDemo -Location "West Europe" -EnabledForDeployment  
-EnabledForTemplateDeployment -EnabledForDiskEncryption  
-EnablePurgeProtection -Sku Standard - Verbose
```

O nome do cofre de chaves precisa ser exclusivo. Você não conseguirá usar o mesmo nome para dois cofres de chave. O comando anterior, quando executado com êxito, criará um novo recurso do Azure Key Vault. A próxima etapa é fornecer acesso a uma entidade de serviço no Key Vault.

## Etapa 2: criar uma entidade de serviço

Em vez de usar uma conta individual para se conectar ao Azure, o Azure fornece entidades de serviço, que são, em essência, contas de serviço que podem ser usadas para se conectar ao Azure Resource Manager e executar atividades. Adicionar um usuário a um diretório/locatário do Azure os torna disponíveis em todos os lugares, inclusive em todos os grupos de recursos e recursos, devido à natureza da herança de segurança no Azure. O acesso deve ser explicitamente revogado de grupos de recursos para usuários se eles não tiverem permissão para acessá-los. As entidades de serviço ajudam a atribuir acesso e controle granular a grupos de recursos e recursos e, se necessário, podem receber acesso ao escopo da assinatura. Eles também podem ser atribuídos permissões granulares, como leitor, colaborador ou permissões de proprietário.

Em suma, as entidades de serviço devem ser o mecanismo preferencial para consumir serviços do Azure. Eles podem ser configurados com uma senha ou com uma chave de certificado. As entidades de serviço podem ser criadas usando o comando **New-AzAdServicePrincipal**, como mostrado aqui:

```
$sp = New-AzADServicePrincipal -DisplayName "keyvault-book" -Scope "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -Role Owner -StartDate ([datetime]::Now) -EndDate $([datetime]::now.AddYears(1)) -Verbose
```

Os valores de configuração importantes são o escopo e a função. O escopo determina a área de acesso para a aplicação de serviço – atualmente é mostrado no nível da assinatura. Os valores válidos para o escopo são os seguintes:

```
/subscriptions/{subscriptionId}
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}
/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/providers/{resourceProviderNamespace}/{resourceType}/{resourceName}
/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/providers/{resourceProviderNamespace}/{parentResourcePath}/{resourceType}/{resourceName}
```

A função fornece permissões para o escopo atribuído. Os valores válidos são os seguintes:

- Proprietário
- Colaborador
- Leitor
- Permissões específicas de recursos

No comando anterior, as permissões de proprietário foram fornecidas para a entidade de serviço recém-criada.

Também podemos usar certificados, se necessário. Para simplificar, procederemos à senha.

Com a entidade de serviço que criamos, o segredo ficará oculto. Para descobrir o segredo, você pode testar os seguintes comandos:

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($sp.Secret)
$UnsecureSecret = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
```

**\$UnsecureSecret** terá sua chave secreta.

Junto com a entidade de serviço, uma aplicação do Diretório de Aplicações será criada. A aplicação atua como a representação global da nossa aplicação em diretórios, e a entidade é como uma representação local da aplicação. Podemos criar várias entidades usando a mesma aplicação em outro diretório. Podemos obter os detalhes da aplicação criada usando o comando **Get-AzAdApplication**. Salvaremos a saída desse comando em uma variável, **\$app**, pois precisaremos dela posteriormente:

```
$app = Get-AzAdApplication -DisplayName $sp.DisplayName
```

Agora criamos uma entidade de serviço usando um segredo; outra maneira segura de criá-la é usando certificados. Na próxima seção, criaremos uma entidade de serviço usando certificados.

### **Etapa 3: criar uma entidade de serviço usando certificados**

Para criar uma entidade de serviço usando certificados, as seguintes etapas devem ser seguidas:

- Criar um certificado autoassinado ou comprar um certificado:** um certificado autoassinado é usado para criar este exemplo de aplicação de ponta a ponta. Para implantações da vida real, um certificado válido deve ser comprado de uma autoridade de certificação.

Para criar um certificado autoassinado, o comando a seguir pode ser executado. O certificado autoassinado é exportável e armazenado em uma pasta pessoal no computador local – ele também tem uma data de expiração:

```
$currentDate = Get-Date
$expiryDate = $currentDate.AddYears(1)
$finalDate = $expiryDate.AddYears(1)
$servicePrincipalName = "https://automation.book.com"
$automationCertificate = New-SelfSignedCertificate -DnsName
$servicePrincipalName -KeyExportPolicy Exportable -Provider "Microsoft
Enhanced RSA and AES Cryptographic Provider" -NotAfter $finalDate
-CertStoreLocation "Cert:\LocalMachine\My"
```

- Exportar o certificado recém-criado:** o novo certificado deve ser exportado para o sistema de arquivos para que, posteriormente, ele possa ser carregado para outros destinos, como o Azure AD, para criar uma entidade de serviço.

Os comandos usados para exportar o certificado para o sistema de arquivos local são mostrados a seguir. Note que este certificado tem chaves públicas e privadas e, por isso, enquanto é exportado, tem de ser protegido utilizando uma senha e a senha tem de ser uma cadeia de caracteres segura:

```
$securepfxpwd = ConvertTo-SecureString -String 'password' -AsPlainText
-Force # Password for the private key PFX certificate
$cert1 = Get-Item -Path Cert:\LocalMachine\My\$($automationCertificate.Thumbprint)
Export-PfxCertificate -Password $securepfxpwd -FilePath " C:\azureautomation.pfx" -Cert $cert1
```

O cmdlet **Get-Item** lê o certificado do repositório de certificados e o armazena na variável **\$cert1**. O cmdlet **Export-PfxCertificate** exporta o certificado no repositório de certificados para o sistema de arquivos. Nesse caso, ele está na pasta **C:\book**.

3. **Ler o conteúdo do arquivo PFX recém-gerado:** um objeto **X509Certificate2** é criado para armazenar o certificado In-memory, e os dados são convertidos em uma cadeia de caracteres de Base64 usando a função **System.Convert**:

```
$newCert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2 -ArgumentList "C:\azureautomation.pfx", $securepfxpwd
$newcertdata = [System.Convert]::ToBase64String($newCert.GetRawCertData())
```

Nós usaremos essa mesma entidade na conexão ao Azure da conta da Automação do Azure. É importante que a ID da aplicação, a ID do locatário, a ID da assinatura e os valores de impressão digital do certificado sejam armazenados em um local temporário para que possam ser usados para configurar recursos subsequentes:

```
$adAppName = "azure-automation-sp"
$ServicePrincipal = New-AzADServicePrincipal -DisplayName $adAppName
-CertValue $newcertdata -StartDate $newCert.NotBefore -EndDate $newCert.NotAfter
Sleep 10
New-AzRoleAssignment -ServicePrincipalName $ServicePrincipal.ApplicationId
-RoleDefinitionName Owner -Scope /subscriptions/xxxxx-xxxxxx-xxxxxx-xxxxxx
```

Nossa entidade de serviço está pronta. O cofre de chaves que criamos não tem um conjunto de políticas de acesso, o que significa que nenhum usuário ou aplicação poderá acessá-lo. Na próxima etapa, concedemos permissões para a aplicação do Diretório de Aplicações que criamos para acessar o cofre de chaves.

#### **Etapa 4: criar uma política de cofre de chaves**

Nessa fase, criamos a entidade de serviço e o cofre de chaves. No entanto, a entidade de serviço ainda não tem acesso ao cofre de chaves. Essa entidade de serviço será usada para consultar e listar todos os segredos, chaves e certificados do cofre de chaves, e ele deve ter as permissões necessárias para fazer isso.

Para fornecer a permissão de entidade de serviço recém-criada para acessar o cofre de chaves, voltaremos ao console do Azure PowerShell e executaremos o seguinte comando:

```
Set-AzKeyVaultAccessPolicy -VaultName keyvaultbook -ResourceGroupName IntegrationDemo -ObjectId $ServicePrincipal.Id -PermissionsToKeys get,list,create -PermissionsToCertificates get,list,import -PermissionsToSecrets get,list -Verbose
```

Referindo-se ao bloco de comando anterior, dê uma olhada nos seguintes pontos:

- **Set-AzKeyVaultAccessPolicy** fornece permissões de acesso a usuários, grupos e entidades de serviço. Ele aceita o nome do cofre de chaves e a ID do objeto de entidade de serviço. Esse objeto é diferente da ID da aplicação. A saída da entidade de serviço contém uma propriedade de **Id**, como mostrado aqui:

```
PS C:\windows\system32> $ServicePrincipal

ServicePrincipalNames : {f48850c9-580a-41d4-a062-77cd623e519e, http://azure-automation-sp}
ApplicationId          : f48850c9-580a-41d4-a062-77cd623e519e
ObjectType             : ServicePrincipal
DisplayName            : azure-automation-sp
Id                     : c95cdd04-5906-4bd7-ab5a-d4e617e98946
Type                  :
```

Figura 11.15: encontrando o ID do objeto da entidade de serviço

- **PermissionsToKeys** fornece acesso a chaves no cofre de chaves e as permissões **get**, **list** e **create** são fornecidas a essa entidade de serviço. Não há nenhuma permissão de gravação ou atualização fornecida a essa entidade.
- **PermissionsToSecrets** fornece acesso a segredos no cofre de chaves e as permissões **get** e **list** são fornecidas a essa entidade de serviço. Não há nenhuma permissão de gravação ou atualização fornecida a essa entidade.
- **PermissionsToCertificates** fornece acesso a chaves no cofre de chaves e as permissões **get**, **list** e **create** são fornecidas a essa entidade de serviço. Não há nenhuma permissão de gravação ou atualização fornecida a essa entidade.

Nesse ponto, configuramos a entidade de serviço para funcionar com o Azure Key Vault. A próxima parte da solução é criar uma conta de automação.

### **Etapa 5: criar uma conta da Automação**

Assim como antes, nós usaremos o Azure PowerShell para criar uma nova conta da Automação do Azure em um grupo de recursos. Antes de criar um grupo de recursos e uma conta da Automação, uma conexão com o Azure deve ser estabelecida. No entanto, desta vez, utilize as credenciais para que a entidade de serviço se conecte ao Azure. As etapas são as seguintes:

- O comando para se conectar ao Azure usando a aplicação de serviço é mostrado a seguir. O valor é obtido das variáveis que inicializamos nas etapas anteriores:

```
Login-AzAccount -ServicePrincipal -CertificateThumbprint $newCert.  
Thumbprint -ApplicationId $ServicePrincipal.ApplicationId -Tenant "xxxx-  
xxxxxx-xxxxx-xxxx"
```

- Certifique-se de que você tenha acesso verificando **Get-AzContext** como mostrado aqui. Anote o ID da assinatura, pois ele será necessário em comandos subsequentes:

```
Get-AzContext
```

- Depois de se conectar ao Azure, um novo recurso com os recursos para a solução e uma nova conta da Automação do Azure devem ser criados. Você está chamando o grupo de recursos de **VaultMonitoring** e criando-o na região **Europa Ocidental**. Você criará o restante dos recursos neste grupo de recursos também:

```
$IntegrationResourceGroup = "VaultMonitoring"  
$rgLocation = "West Europe"  
$automationAccountName = "MonitoringKeyVault"  
New-AzResourceGroup -name $IntegrationResourceGroup -Location $rgLocation  
New-AzAutomationAccount -Name $automationAccountName -ResourceGroupName  
$IntegrationResourceGroup -Location $rgLocation -Plan Free
```

- Em seguida, crie três variáveis de automação. Os valores para eles, ou seja, ID de assinatura, ID de locatário e ID de aplicação, já devem estar disponíveis seguindo as etapas anteriores:

```
New-AzAutomationVariable -Name "azuresubscriptionid"  
-AutomationAccountName $automationAccountName -ResourceGroupName  
$IntegrationResourceGroup -Value "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx "  
-Encrypted $true
```

```
New-AzAutomationVariable -Name "azuretenantid" -AutomationAccountName  
$automationAccountName -ResourceGroupName $IntegrationResourceGroup -Value  
"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx " -Encrypted $true
```

```
New-AzAutomationVariable -Name "azureappid" -AutomationAccountName  
$automationAccountName -ResourceGroupName $IntegrationResourceGroup -Value  
"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx " -Encrypted $true
```

5. Agora é hora de fazer upload de um certificado, que será usado para conexão com o Azure a partir da Automação do Azure:

```
$securepfxpwd = ConvertTo-SecureString -String 'password' -AsPlainText  
-Force # Password for the private key PFX certificate  
New-AzAutomationCertificate -Name "AutomationCertificate" -Path "C:\book\azureautomation.pfx" -Password $securepfxpwd -AutomationAccountName $automationAccountName -ResourceGroupName $IntegrationResourceGroup
```

6. A próxima etapa é instalar os módulos do PowerShell relacionados ao Key Vault e à Grade de Eventos na conta da Automação do Azure, pois esses módulos não são instalados por padrão.
7. No portal do Azure, navegue até o grupo de recursos **VaultMonitoring** já criado clicando em **Grupos de Recursos** no menu à esquerda.
8. Clique na conta da Automação do Azure já provisionada, **MonitoringKeyVault** e, então, clique em **Módulos** no menu à esquerda. O módulo Grade de Eventos depende do módulo **Az.profile** e, portanto, temos que instalá-lo antes do módulo Grade de Eventos.
9. Clique em **Procurar na Galeria** no menu superior e digite **Az.profile** na caixa de pesquisa, como mostrado na Figura 11.16:

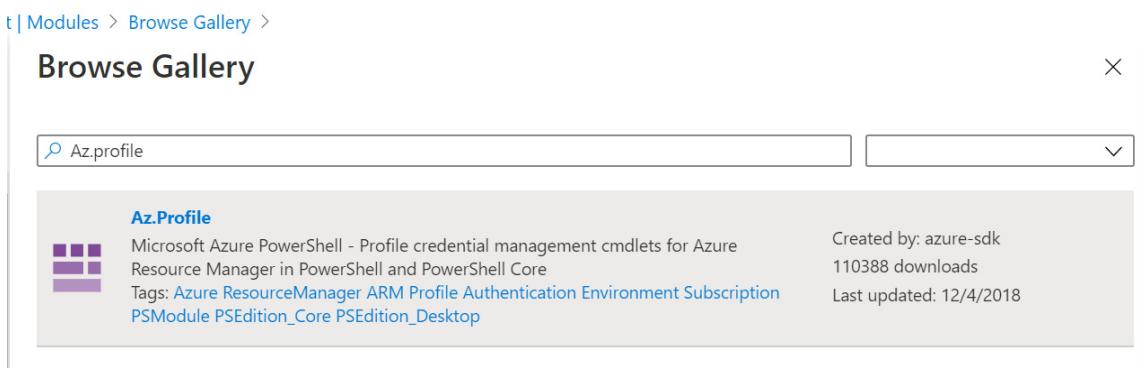


Figura 11.16: o módulo Az.Profile na galeria de módulos

10. Nos resultados da pesquisa, selecione **Az.Profile** e clique no botão **Importar** no menu superior. Por fim, clique no botão **OK**. Esta etapa leva alguns segundos para ser concluída. Depois de alguns segundos, o módulo deve ser instalado.

11. O status da instalação pode ser verificado no item de menu **Módulo**. A Figura 11.17 demonstra como podemos importar um módulo:

The screenshot shows the PowerShell Gallery page for the **Az.Profile** module. At the top, it says "PowerShell Module". Below that is a "Import" button with a downward arrow icon. The main content area has the following details:

- Created by:** azure-sdk
- Tags:** Azure ResourceManager ARM Profile Authentication Environment Subscription PSModule PSEdition\_Core PSEdition/Desktop
- Version:** 0.7.0
- Downloads:** 110,155
- Last updated:** 12/4/2018

Below these details are links: "Learn more", "View in PowerShell Gallery", "Documentation", and "Licensing information".

The "Content" section has a search bar with placeholder text "Search to filter items...". Below the search bar is a table with two columns: "Type" and "Name". The table lists the following cmdlets:

Type	Name
Cmdlet	Disable-AzDataCollection
Cmdlet	Disable-AzContextAutosave
Cmdlet	Enable-AzDataCollection
Cmdlet	Enable-AzContextAutosave
Cmdlet	Remove-AzEnvironment
Cmdlet	Get-AzEnvironment
Cmdlet	Set-AzEnvironment
Cmdlet	Add-AzEnvironment

Figura 11.17: status do módulo Az.Profile

12. Siga as etapas 9, 10 e 11 novamente para importar e instalar o módulo **Az.EventGrid**. Se você for avisado para instalar quaisquer dependências antes de prosseguir, vá em frente e instale as dependências primeiro.
13. Siga as etapas 9, 10 e 11 novamente para importar e instalar o módulo **Az.KeyVault**. Se você for avisado para instalar quaisquer dependências antes de prosseguir, vá em frente e instale a dependência primeiro.

Como importamos os módulos necessários, vamos prosseguir e criar o tópico de Grade de Eventos.

## Etapa 6: criar um tópico da Grade de Eventos

Se você se lembra da arquitetura que usamos, precisamos de um tópico de Grade de Eventos. Vamos criar um.

O comando usado para criar um tópico de Grade de Eventos usando o PowerShell é o seguinte:

```
New-AzEventGridTopic -ResourceGroupName VaultMonitoring -Name  
azureforarchitects-topic -Location "West Europe"
```

O processo de criação de um tópico de Grade de Eventos usando o portal do Azure é o seguinte:

1. No portal do Azure, navegue até o grupo de recursos **Vaultmonitoring** já criado clicando em **Grupos de Recursos** no menu à esquerda.
2. Em seguida, clique no botão **+Adicionar** e procure **Tópico de Grade de Eventos** na caixa de pesquisa. Selecione-o e clique no botão **Criar**.
3. Preencha os valores apropriados no formulário resultante fornecendo um nome, selecionando uma assinatura, o grupo de recursos recém-criado, o local e o esquema do evento.

Como já discutimos, o tópico de Grade de Eventos fornece um ponto de extremidade em que a fonte enviará os dados. Já que temos nosso tópico pronto, vamos preparar a conta da Automação de origem.

## Etapa 7: criar o runbook

Esta etapa se concentrará na criação de uma conta da Automação do Azure e Runbooks do PowerShell que conterá a lógica principal de ler Azure Key Vaults e recuperar segredos armazenados nele. As etapas necessárias para configurar a Automação do Azure são:

1. **Criar o runbook da Automação do Azure:** no portal do Azure, navegue até o grupo de recursos **Vaultmonitoring** já criado clicando no ícone **Grupos de Recursos** no menu à esquerda.
2. Clique na conta da Automação do Azure já provisionada, **MonitoringKeyVault**. Em seguida, clique em **Runbooks** no menu à esquerda e clique em **+Adicionar um Runbook** no menu superior.
3. Clique em **Criar um novo Runbook** e forneça um nome. Vamos chamar este runbook de **CheckExpiredAssets** e então definir o **Tipo de runbook** como **PowerShell**:

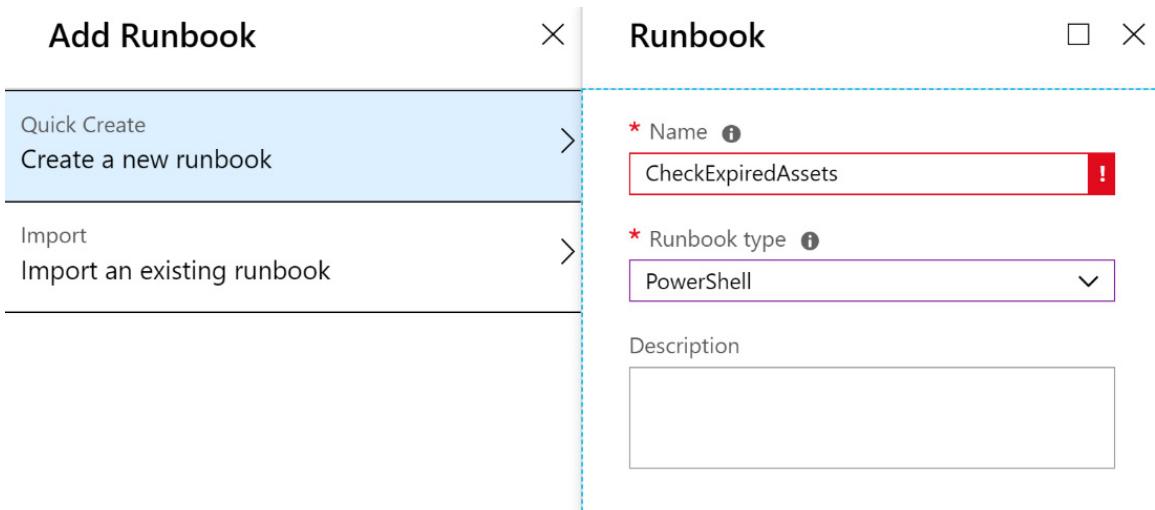


Figura 11.18: criar um runbook

4. **Codificar o runbook:** declare algumas variáveis para manter a ID da assinatura, ID do locatário, ID da aplicação e as informações de impressão digital do certificado. Esses valores devem ser armazenados em variáveis da Automação do Azure e o upload do certificado deve ser feito nos certificados da Automação. A chave usada para o certificado obtido por upload é **AutomationCertificate**. Os valores são recuperados desses armazenamentos e são atribuídos às variáveis, conforme mostrado a seguir:

```
$subscriptionID = get-AutomationVariable "azuresubscriptionid"
$tenantID = get-AutomationVariable "azuretenantid"
$applicationId = get-AutomationVariable "azureappid"
$cert = get-AutomationCertificate "AutomationCertificate"
$certThumbprint = ($cert.Thumbprint).ToString()
```

5. O próximo código no runbook ajuda a fazer logon no Azure usando a entidade de serviço com valores de variáveis declaradas anteriormente. Além disso, o código seleciona uma assinatura apropriada. O código é mostrado a seguir:

```
Login-AzAccount -ServicePrincipal -CertificateThumbprint $certThumbprint
-ApplicationId $applicationId -Tenant $tenantID
Set-AzContext -SubscriptionId $subscriptionID
```

Como a Grade de Eventos do Azure foi provisionada na etapa 6 desta seção, seu ponto de extremidade e suas chaves são recuperados usando os cmdlets **Get-AzEventGridTopic** e **Get-AzEventGridTopicKey**.

A Grade de Eventos do Azure gera duas chaves – uma primária e uma secundária. A primeira referência fundamental é tomada da seguinte maneira:

```
$eventGridName = "ExpiredAssetsKeyVaultEvents"
$eventGridResourceGroup = "VaultMonitoring"
$topicEndpoint = (Get-AzEventGridTopic -ResourceGroupName
$eventGridResourceGroup -Name $eventGridName).Endpoint
$keys = (Get-AzEventGridTopicKey -ResourceGroupName
$eventGridResourceGroup -Name $eventGridName ).Key1
```

- Em seguida, todos os cofres de chaves que foram provisionados na assinatura são recuperados usando a iteração. Durante o loop, todos os segredos são recuperados usando o cmdlet **Get-AzKeyVaultSecret**.

A data de expiração de cada segredo é comparada à data atual e, se a diferença for inferior a um mês, gerará um evento de Grade de Eventos e o publicará usando o comando **Invoke-WebRequest**.

As mesmas etapas são executadas para certificados armazenados no cofre de chaves. O cmdlet usado para recuperar todos os certificados é **Get-AzKeyVaultCertificate**.

O evento que é publicado na Grade de Eventos deve estar na matriz JSON. A mensagem gerada é convertida em JSON usando o cmdlet **ConvertTo-Json** e, em seguida, convertida em uma matriz por meio da adição de [ e ] como um prefixo e sufixo.

Para se conectar à Grade de Eventos do Azure e publicar o evento, o remetente deverá fornecer a chave em seu cabeçalho. A solicitação falhará se esses dados estiverem ausentes na carga da solicitação:

```
$keyvaults = Get-AzureRmKeyVault
foreach($vault in $keyvaults) {
    $secrets = Get-AzureKeyVaultSecret -VaultName $vault.VaultName
    foreach($secret in $secrets) {
        if( ![string]::IsNullOrEmpty($secret.Expires) ) {
            if($secret.Expires.AddMonths(-1) -lt [datetime]::Now)
            {
                $secretDataMessage = @{
                    id = [System.guid]::NewGuid()
                    subject = "Secret Expiry happening soon !!"
                    eventType = "Secret Expiry"
                    eventTime = [System.DateTime]::UtcNow
                    data = @{
                        "ExpiryDate" = $secret.Expires
                        "SecretName" = $secret.Name.ToString()
                    }
                }
                $secretDataMessage | ConvertTo-Json | Add-Content -Path "C:\temp\SecretExpiry.json"
            }
        }
    }
}
```

```

    "VaultName" = $secret.VaultName.ToString()
    "SecretCreationDate" = $secret.Created.ToString()
    "IsSecretEnabled" = $secret.Enabled.ToString()
    "SecretId" = $secret.Id.ToString()
}
}
...
Invoke-WebRequest -Uri $topicEndpoint -Body $finalBody -Headers $header
-Method Post -UseBasicParsing
}
}
Start-Sleep -Seconds 5
}
}

```

7. Publique o runbook clicando no botão **Publicar**, conforme mostrado na Figura 11.19:

[Home](#) > [Automation Accounts](#) | ee > [MonitoringKeyVault](#) | Runbooks > [CheckExpiredAssets \(MonitoringKeyVault/CheckExpiredAssets\)](#) >

```

Edit PowerShell Runbook*
CheckExpiredAssets

Save Publish Revert to published Test pane Feedback

CMDLETS
1 $subscriptionID = get-AutomationVariable "azuresubscriptionid"
2 $tenantID = get-AutomationVariable "azuretenantid"
3 $applicationId = get-AutomationVariable "azureappid"
4 $cert = get-AutomationCertificate "AutomationCertificate"
5 $certThumbprint = ($cert.Thumbprint).ToString()
6 Login-AzAccount -ServicePrincipal -CertificateThumbprint $certThumbprint -ApplicationId $applicationId -Tenant $tenantID
7 Set-AzContext -SubscriptionId $subscriptionID
8 $eventGridName = "azureforarchitects-topic"
9 $eventGridResourceGroup = "VaultMonitoring"
10 $topicEndpoint = (Get-AzEventGridTopic -ResourceGroupName $eventGridResourceGroup -Name $eventGridName).Endpoint
11 $keys = (Get-AzEventGridTopicKey -ResourceGroupName $eventGridResourceGroup -Name $eventGridName).Key1
12
13 $keyvaults = Get-AzKeyVault
14 foreach($vault in $keyvaults) {
15 $secrets = Get-AzKeyVaultSecret -VaultName $vault.VaultName

```

Figura 11.19: publicando o runbook

8. **Agendador:** crie um ativo de agendador da Automação do Azure para executar esse runbook uma vez por dia à meia-noite. Clique em **Agendamentos** no menu à esquerda da Automação do Azure e clique em **+Adicionar um agendamento** no menu superior.
9. Forneça informações de agendamento no formulário resultante.

Isso deve concluir a configuração da conta da Automação do Azure.

## Etapa 8: trabalhar com SendGrid

Nesta etapa, criaremos um novo recurso SendGrid. O recurso SendGrid é usado para enviar emails da aplicação sem a necessidade de instalar um servidor **Simple Mail Transfer Protocol (SMTP)**. Ele fornece uma API REST e um **Kit de Desenvolvimento de Software (SDK) do C#**, por meio do qual é muito fácil enviar emails em massa. Na solução atual, o Azure Functions será usado para invocar as APIs do SendGrid para enviar emails e, portanto, esse recurso precisa ser provisionado. Esse recurso tem custos separados e não é abordado como parte do custo do Azure – há um nível gratuito disponível que pode ser usado para enviar emails:

1. Um recurso **SendGrid** é criado como qualquer outro recurso do Azure. Procure **sendgrid**, e nós obteremos **Entrega de Email de SendGrid** nos resultados.
2. Selecione o recurso e clique no botão **Criar** para abrir seu formulário de configuração.
3. Selecione uma camada de preços apropriada.
4. Forneça os detalhes de contato apropriados.
5. Marque a caixa de seleção **Termos de Uso**.
6. Preencha o formulário e clique no botão **Criar**.
7. Depois que o recurso for provisionado, clique no botão **Gerenciar** no menu superior – isso abrirá o site do SendGrid. O site pode solicitar configuração de email. Em seguida, selecione **Chaves de API** na seção **Configurações** e clique no botão **Criar Chave de API**:

The screenshot shows the SendGrid interface with the user 'Ritesh Modi' logged in. The left sidebar has navigation links: Dashboard, Marketing, Templates, Stats, Activity, Suppressions, and Settings. Under Settings, 'API Keys' is selected. The main content area is titled 'API Keys' and features a large key icon. Below it, the text reads: 'Get started creating API Keys. API keys help protect the sensitive areas of your SendGrid account (e.g. contacts and account settings). To control and limit access of API users, you can create multiple API keys, each with different permissions.' At the top right of this section is a blue button labeled 'Create API Key'.

Figura 11.20: criando chaves de API para o SendGrid

8. Na janela resultante, selecione **Acesso Completo** e clique no botão **Criar e Exibir**. Isso criará a chave para o recurso SendGrid; anote essa chave, pois ela será usada com a configuração do Azure Functions para SendGrid:

## Create API Key

The screenshot shows the 'Create API Key' interface. At the top, there's a field labeled 'API Key Name' with a red asterisk indicating it's required. Below it is a section titled 'API Key Permissions' with a red asterisk and an information icon. Three access level options are listed in boxes: 'Full Access' (selected), 'Restricted Access', and 'Billing Access'. Each option has a description and an information icon. At the bottom right are 'Cancel' and 'Create & View' buttons.

API Key Name • ⓘ

API Key Permissions • ⓘ

Full Access

Allows the API key to access GET, PATCH, PUT, DELETE, and POST endpoints for all parts of your account, excluding billing.

Restricted Access

Customize levels of access for all parts of your account, excluding billing.

Billing Access

Allows the API key to access billing endpoints for the account. (This is especially useful for Enterprise or Partner customers looking for more advanced account management.)

Cancel Create & View

Figura 11.21: configurando o nível de acesso no portal SendGrid

Agora que configuramos os níveis de acesso para o SendGrid, vamos configurar outro serviço de terceiros, que é chamado de Twilio.

### Etapa 9: introdução ao Twilio

Nesta etapa, criaremos uma nova conta do Twilio. O Twilio é usado para enviar mensagens SMS em massa. Para criar uma conta com o Twilio, navegue até [twilio.com](https://www.twilio.com) e crie uma nova conta. Depois de criar uma conta com êxito, um número de celular será gerado e poderá ser usado para enviar mensagens SMS para receptores:



Figura 11.22: escolhendo um número do Twilio

A conta do Twilio fornece chaves de produção e teste. Copie a chave de teste e o token para um local temporário, como o Bloco de Notas, pois eles serão necessários posteriormente no Azure Functions:

ritesh.modi@outlook.com's Account Dashboard

The screenshot shows the Twilio Account Dashboard under the heading 'Project Info'. It displays the following information:

- TRIAL BALANCE:** \$14.44 (highlighted in green)
- TRIAL NUMBER:** [REDACTED]
- ACCOUNT SID:** [REDACTED] (with a copy icon to its right)
- AUTH TOKEN:** [REDACTED] (with a copy icon to its right)
- Need more numbers?** (with a question mark icon)

Figura 11.23: configurando o Twilio

Temos o SendGrid e o Twilio para o serviço de notificação; no entanto, precisamos de algo que possa assumir o evento e notificar os usuários. Aqui entra o papel de um aplicativo de função. Na próxima seção, criaremos um aplicativo de função que ajudará no envio de SMS e emails.

## Etapa 10: configurar um aplicativo de função

Nesta etapa, criaremos um novo aplicativo de função responsável pelo envio de emails e notificações por SMS. A finalidade do aplicativo de função na solução é enviar mensagens de notificação aos usuários sobre a expiração de segredos no cofre de chaves. Uma única função será responsável pelo envio de emails e mensagens SMS – note que isso poderia ter sido dividido em duas funções separadas. A primeira etapa é criar um novo aplicativo de função e hospedar uma função dentro dele:

1. Como fizemos antes, navegue até o seu grupo de recursos, clique no botão **+Adicionar** no menu superior e procure o recurso de **aplicativo de função**. Em seguida, clique no botão **Criar** para obter o formulário **Aplicativo de Função**.
2. Preencha o formulário **Aplicativo de Função** e clique no botão **Criar**. O nome do aplicativo de função deve ser exclusivo em todo o Azure.
3. Depois que o aplicativo de função for provisionado, crie uma nova função chamada **SMSandEMailFunction** clicando no botão + ao lado do item **Funções** no menu à esquerda. Em seguida, selecione **In-portal** no painel central.
4. Selecione **Gatilho HTTP** e chame-o de **SMSandEMailFunction**. Em seguida, clique no botão **Criar**; a opção **Nível de autorização** pode ter qualquer valor.
5. Remova o código padrão, substitua-o pelo código mostrado na lista a seguir e clique no botão **Salvar** no menu superior:

```
#r "SendGrid"
#r "Newtonsoft.Json"
#r "Twilio.Api"
using System.Net;
using System;
using SendGrid.Helpers.Mail;
using Microsoft.Azure.WebJobs.Host;
using Newtonsoft.Json;
using Twilio;
using System.Configuration;
public static HttpResponseMessage Run(HttpRequestMessage req, TraceWriter log, out Mail message,out SMSMessage sms)
{
    log.Info("C# HTTP trigger function processed a request.");
    string alldata = req.Content.ReadAsStringAsync().GetAwaiter().GetResult();
    message = new Mail();
    var personalization = new Personalization();
    personalization.AddBcc(new Email(ConfigurationManager.AppSettings["bccStakeholdersEmail"]));
}
```

```

personalization.AddTo(new Email(ConfigurationManager.
AppSettings["toStakeholdersEmail"]));
var messageContent = new Content("text/html", alldata);
message.AddContent(messageContent);
message.AddPersonalization(personalization);
message.Subject = "Key Vault assets Expiring soon..";
message.From = new Email(ConfigurationManager.AppSettings["serviceEmail"]);
string msg = alldata;
sms = new SMSMessage();
sms.Body = msg;
sms.To = ConfigurationManager.AppSettings["adminPhone"];
sms.From = ConfigurationManager.AppSettings["servicePhone"];
return req.CreateResponse(HttpStatusCode.OK, "Hello ");
}

```

6. Clique no nome do aplicativo de função no menu à esquerda e clique novamente no link **Configurações da aplicação** na janela principal:

**NotificationFunctionAppBook**

Function Apps

Overview      Platform features

Status: Running      Subscription: RiteshSubscription      Resource group: VaultMonitoring

Subscription ID: [REDACTED]      Location: West Europe

Configured features

- Function app settings
- Application settings**
- Application Insights

Figura 11.24: navegando até as configurações da aplicação

7. Navegue até a seção **Configurações da aplicação**, conforme mostrado na Figura 11.24, e adicione algumas entradas clicando em **+ Adicionar nova configuração** para cada entrada.

Observe que as entradas estão no formato de pares de chave–valor e os valores devem ser valores reais em tempo real. **adminPhone** e **servicePhone** já devem estar configurados no site do Twilio. **servicePhone** é o número de telefone gerado pelo Twilio que é usado para enviar mensagens SMS, e **adminPhone** é o número de telefone do administrador a quem o SMS deve ser enviado.

Lembre-se também de que o Twilio espera que o número de telefone de destino esteja em um formato específico, dependendo do país (para a Índia, o formato é **+91 xxxxx xxxx**). Observe os espaços e o código do país no número.

Também precisamos adicionar as chaves ao SendGrid e Twilio nas configurações da aplicação. Essas configurações são mencionadas na lista a seguir. Você já pode ter esses valores úteis devido a atividades executadas nas etapas anteriores:

- O valor de **SendGridAPIKeyAsAppSetting** é a chave para SendGrid.
- **TwilioAccountSid** é o identificador do sistema para a conta do Twilio. Esse valor já foi copiado e armazenado em um local temporário na Etapa 9: *introdução ao Twilio*.
- **TwilioAuthToken** é o token para a conta do Twilio. Esse valor já foi copiado e armazenado em um local temporário em uma etapa anterior.

8. Salve as configurações clicando no botão **Salvar** no menu superior:

## Application settings

Application Settings are encrypted at rest and transmitted over an encrypted connection.

[Hide Values](#) [Show Values](#)

APP SETTING NAME	VALUE
APPINSIGHTS_INSTRUMENTATIONKEY	<i>Hidden value. Click to edit.</i>
AzureWebJobsStorage	<i>Hidden value. Click to edit.</i>
FUNCTIONS_EXTENSION_VERSION	<i>Hidden value. Click to edit.</i>
FUNCTIONS_WORKER_RUNTIME	<i>Hidden value. Click to edit.</i>
WEBSITE_CONTENTAZUREFILECONNECTIO...	<i>Hidden value. Click to edit.</i>
WEBSITE_CONTENTSHARE	<i>Hidden value. Click to edit.</i>
WEBSITE_NODE_DEFAULT_VERSION	<i>Hidden value. Click to edit.</i>
adminPhone	<i>Hidden value. Click to edit.</i>
servicePhone	<i>Hidden value. Click to edit.</i>
serviceEmail	<i>Hidden value. Click to edit.</i>
bccStakeholdersEmail	<i>Hidden value. Click to edit.</i>
toStakeholdersEmail	<i>Hidden value. Click to edit.</i>
SendGridAPIKeyAsAppSetting	<i>Hidden value. Click to edit.</i>
TwilioAccountSid	<i>Hidden value. Click to edit.</i>
TwilioAuthToken	<i>Hidden value. Click to edit.</i>

[+ Add new setting](#)

Figura 11.25: definindo as configurações da aplicação

9. Clique no link **Integrar** no menu à esquerda logo abaixo do nome da função e clique em **+ Nova Saída**. Isso é para adicionar uma saída para o serviço SendGrid:

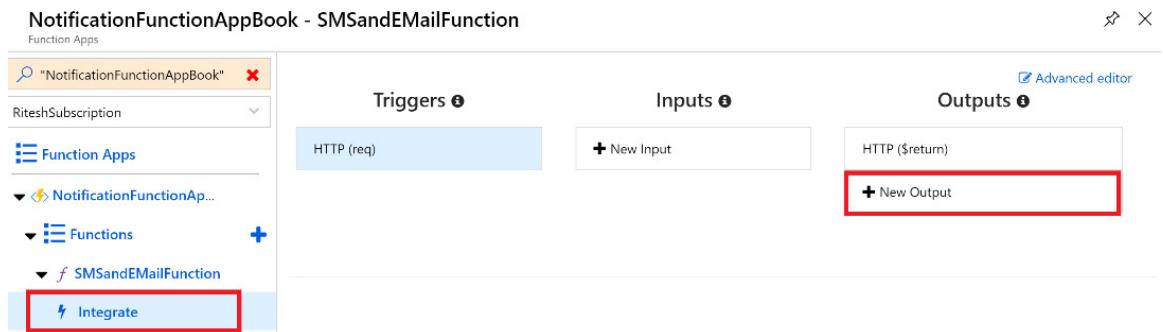


Figura 11.26: adicionando uma saída ao aplicativo de função

10. Em seguida, selecione **SendGrid** – ele pode solicitar a instalação da extensão do SendGrid. Instale a extensão, o que levará alguns minutos:

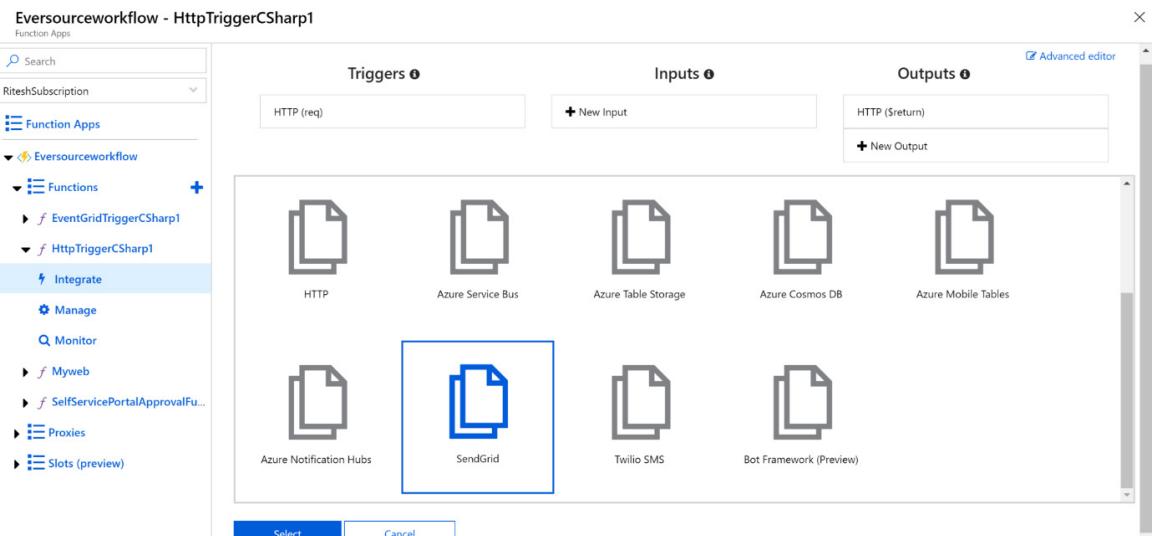


Figura 11.27: configurando um aplicativo de função

11. Depois de instalar a extensão, o formulário de configuração de saída será exibido. Os itens de configuração importantes neste formulário são **Nome do parâmetro de mensagem** e **Configuração de aplicativo de chave da API de SendGrid**. Deixe o valor padrão para **Nome do parâmetro de mensagem** e clique na lista suspensa para selecionar **SendGridAPIKeyAsAppSetting** como a chave de configuração do aplicativo de API. Isso já foi configurado em uma etapa anterior na configuração de definições do aplicativo. O formulário deve ser configurado, como mostrado na Figura 11.28. Em seguida, você precisa clicar no botão **Salvar**:

SendGrid output

Message parameter name ?  
message

SendGrid API Key App Setting ? [show value](#) [new](#)  
SendGridAPIKeyAsAppSetting

From address ?  
From address

To address ?  
To address

Message subject ?  
Message subject

Message Text ?  
Message Text

[Save](#) [Cancel](#)

[+ Documentation](#)

Figura 11.28: configurando o SendGrid

12. Clique em **+ Nova Saída** novamente; isso serve para adicionar uma saída para o serviço Twilio.
13. Em seguida, selecione **SMS do Twilio**. Ele pode solicitar que você instale a extensão de SMS do Twilio. Instale a extensão, o que levará alguns minutos.
14. Depois de instalar a extensão, o formulário de configuração de saída será exibido. Os itens de configuração importantes neste formulário são **Nome do parâmetro de mensagem**, **Configuração de SID da conta** e **Configuração do Token de Autenticação**. Altere o valor padrão para o **Nome do parâmetro de mensagem** para **sms**. Isso é feito porque o parâmetro **mensagem** já é usado para o parâmetro de serviço SendGrid. Certifique-se de que o valor de **Configuração de SID da conta** seja **TwilioAccountSid** e que o valor da **Configuração do Token de Autenticação** seja **TwilioAuthToken**. Esses valores já foram configurados em uma etapa anterior da configuração de definições do aplicativo. O formulário deve ser configurado, como mostrado na Figura 11.29. Em seguida, você deverá clicar em **Salvar**:

Twilio SMS output

Extension Installation Succeeded

Message parameter name

Account SID setting

Use function return value

Auth Token setting

From number

Message text

Save Cancel

Figura 11.29: configurando a saída de SMS do Twilio

Nossas contas do SendGrid e do Twilio estão prontas. Chegou a hora de usar os conectores e adicioná-los ao Aplicativo Lógico. Na próxima parte, criaremos o Aplicativo Lógico e usaremos conectores para trabalhar com os recursos que criamos até agora.

### Etapa 11: criar um Aplicativo Lógico

Nesta etapa, criaremos um novo fluxo de trabalho de Aplicativo Lógico. Nós temos o autor de um runbook da Automação do Azure que consulta todos os segredos em todos os cofres de chaves e publica um evento caso ele encontre algum deles expirando em um mês. O fluxo de trabalho de Aplicativos Lógicos atua como um assinante para estes eventos:

1. A primeira etapa no menu **Aplicativo Lógico** é criar um fluxo de trabalho de Aplicativos Lógicos.
2. Preencha o formulário resultante depois de clicar no botão **Criar**. Estamos provisionando o aplicativo lógico no mesmo grupo de recursos do que os outros recursos para esta solução.
3. Depois que o aplicativo lógico for provisionado, ele abrirá a janela do designer. Selecione **Aplicativo Lógico em Branco** na seção **Modelos**.
4. Na janela resultante, adicione um gatilho que pode assinar a Grade de Evento. Os Aplicativos Lógicos fornecem um gatilho para a Grade de Eventos, e você pode procurar por isso para ver se ele está disponível.

5. Em seguida, selecione o gatilho **Quando ocorrer um evento de recurso (visualização)**:

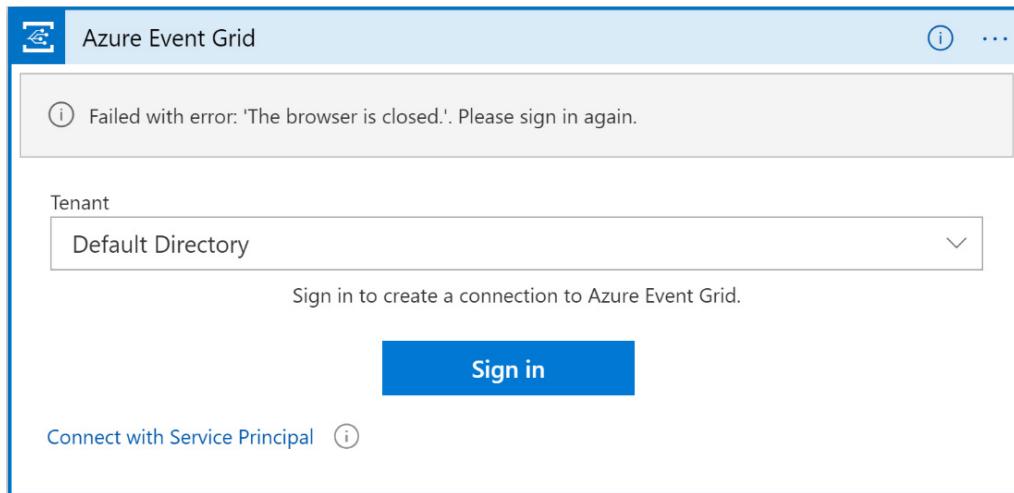


Figura 11.30: selecionando um gatilho da Grade de Eventos

6. Na janela resultante, selecione **Conectar-se com Entidade de Serviço**.

Forneça os detalhes da entidade de serviço, incluindo a ID da aplicação (**ID do Cliente**), a ID do locatário e a senha. Este gatilho não aceita uma entidade de serviço que autentique com o certificado – ele aceita uma entidade de serviço somente com uma senha. Crie uma nova entidade de serviço neste estágio que autentique com uma senha (as etapas para criar uma entidade de serviço com base na autenticação de senha foram abordadas anteriormente na etapa 2) e use os detalhes da entidade do serviço recém-criada para a configuração da Grade de Eventos do Azure, como mostrado na Figura 11.31:

*Connection Name	(Redacted)
Client ID	0d2
Client Secret	.....
Tenant	1e7e

Figura 11.31: fornecendo os detalhes da entidade de serviço para conexão

7. Selecione a assinatura. Com base no escopo da entidade de serviço, isso será preenchido automaticamente. Selecione **Microsoft.EventGrid.Topics** como o valor **Tipo de Recurso** e defina o nome do tópico personalizado como **ExpiredAssetsKeyVaultEvents**:

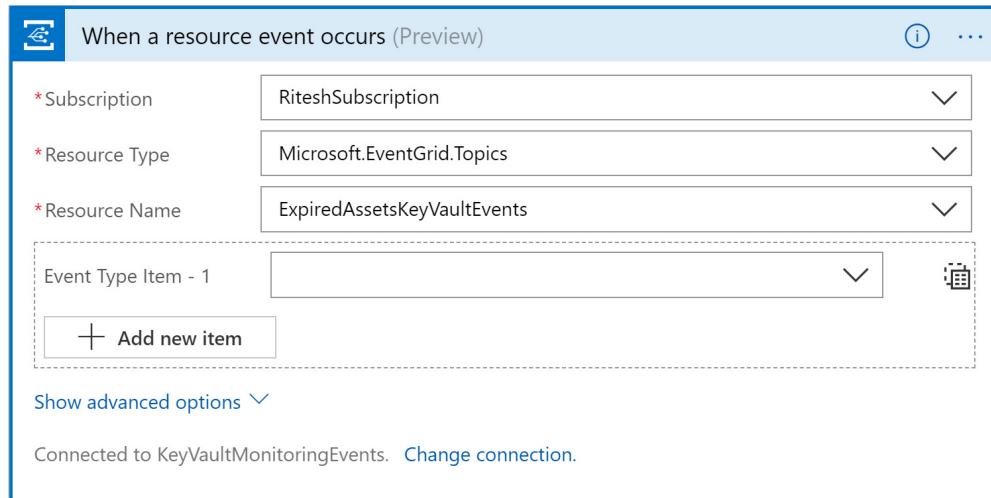


Figura 11.32: fornecendo detalhes do gatilho da Grade de Eventos

8. A etapa anterior criará um conector e as informações de conexão poderão ser alteradas clicando em **Alterar conexão**.
9. A configuração final do gatilho da Grade de Eventos deve ser semelhante à Figura 11.33:

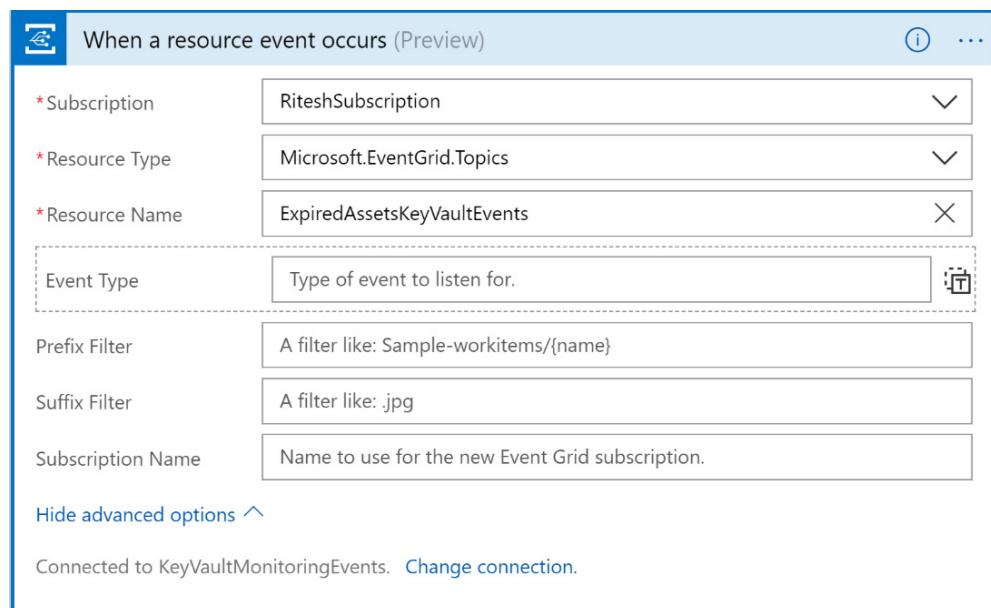


Figura 11.33: visão geral da Grade de Eventos

10. Adicione uma nova atividade **Analisar JSON** depois do gatilho da Grade Eventos – essa atividade precisa do esquema JSON. Geralmente, o esquema não está disponível, mas essa atividade ajudará a gerar o esquema se um JSON válido for fornecido a ele:



Figura 11.34: atividade Analisar JSON

11. Clique em **Usar o conteúdo de amostra para gerar o esquema** e forneça os seguintes dados:

```
{
  "ExpiryDate": "",
  "SecretName": "",
  "VaultName": "",
  "SecretCreationDate": "",
  "IsSecretEnabled": "",
  "SecretId": ""
}
```

Pode surgir aqui uma pergunta a respeito do conteúdo de amostra. Nesta fase, como você calcula a carga gerada pelo editor da Grade de Eventos? A resposta para isso reside no fato de que esse conteúdo de amostra é exatamente o mesmo que é usado no elemento de dados no runbook de automação do Azure. Você pode dar uma olhada no trecho de código novamente:

```
data = @{
  "ExpiryDate" = $certificate.Expires
  "CertificateName" = $certificate.Name.ToString()
  "VaultName" = $certificate.VaultName.ToString()
  "CertificateCreationDate" = $certificate.Created.ToString()
  "IsCertificateEnabled" = $certificate.Enabled.ToString()
  "CertificateId" = $certificate.Id.ToString()
}
```

12. A caixa **Conteúdo** deve conter conteúdo dinâmico vindo do gatilho anterior, conforme demonstrado na Figura 11.35:

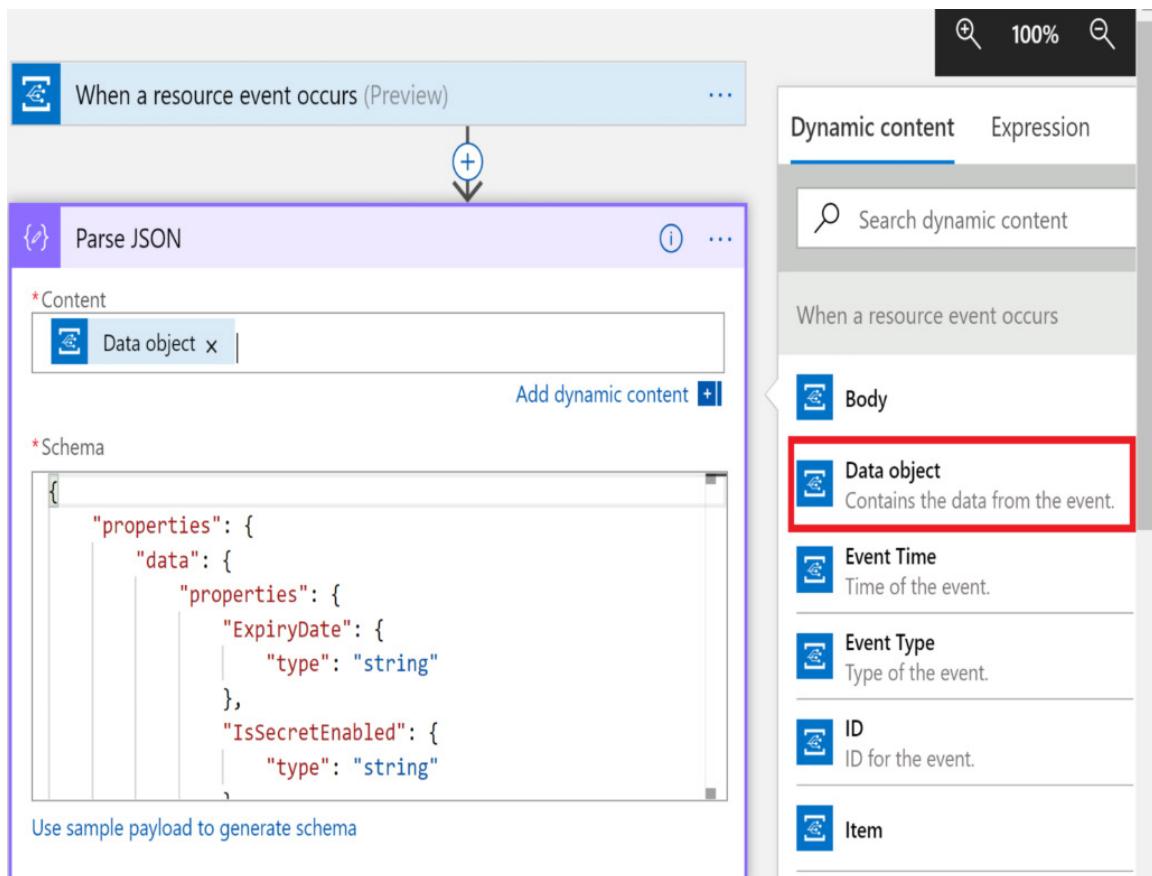


Figura 11.35: fornecendo conteúdo dinâmico para a atividade Analisar JSON

13. Adicione outra ação de **Azure Functions** depois de **Analisar JSON** e então selecione **Escolher uma função do Azure**. Selecione os aplicativos de função do Azure chamados **NotificationFunctionAppBook** e **SMSAndEmailFunction**, que foram criados anteriormente:

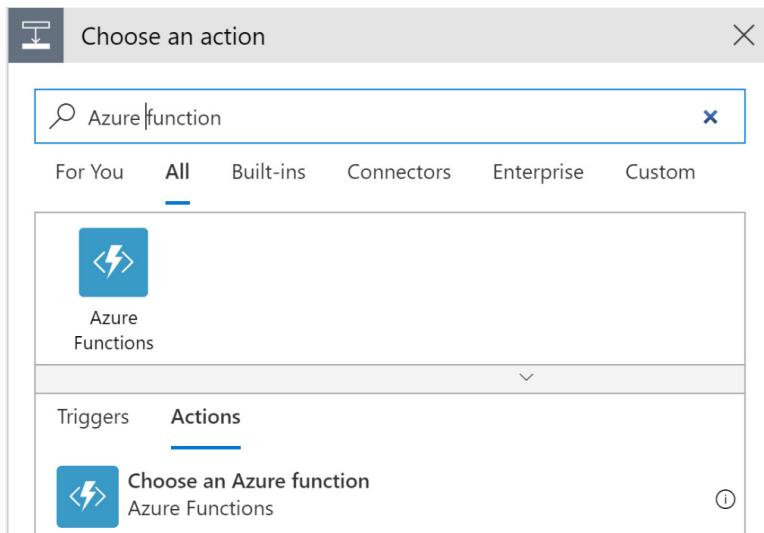


Figura 11.36: adicionando uma ação do Azure Functions

14. Clique na área de texto **Corpo da Solicitação** e preencha-a com o seguinte código. Isso é feito para converter os dados em JSON antes de enviá-los para a função do Azure:

```
{
  "alldata" :
}
```

15. Coloque o cursor depois de ":" no código anterior e clique em **Adicionar conteúdo dinâmico | Corpo** da atividade anterior:

Figura 11.37: convertendo dados para JSON antes de enviá-los para uma função do Azure

16. Salve o aplicativo lógico inteiro; ele deve ter a seguinte aparência:

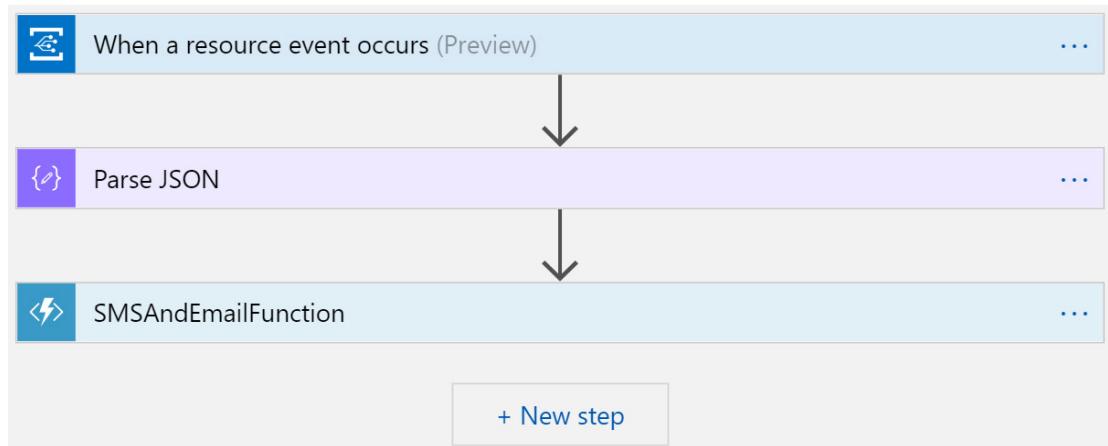


Figura 11.38: fluxo de trabalho de aplicativo lógico

Depois de salvar o aplicativo lógico, sua solução estará pronta para ser testada. Se você não tiver chaves ou segredos, tente adicioná-los com uma data de expiração para que você possa confirmar se sua solução está funcionando.

## Testes

Faça upload de alguns segredos e certificados que tenham datas de expiração para o Azure Key Vault e execute o runbook da Automação do Azure. O runbook está agendado para ser executado por agendamento. Além disso, o runbook publicará eventos na Grade de Eventos. O Aplicativo Lógico deve ser habilitado e ele escolherá o evento e, por fim, invocará a função do Azure para enviar notificações por email e SMS.

O email deve ter a seguinte aparência:

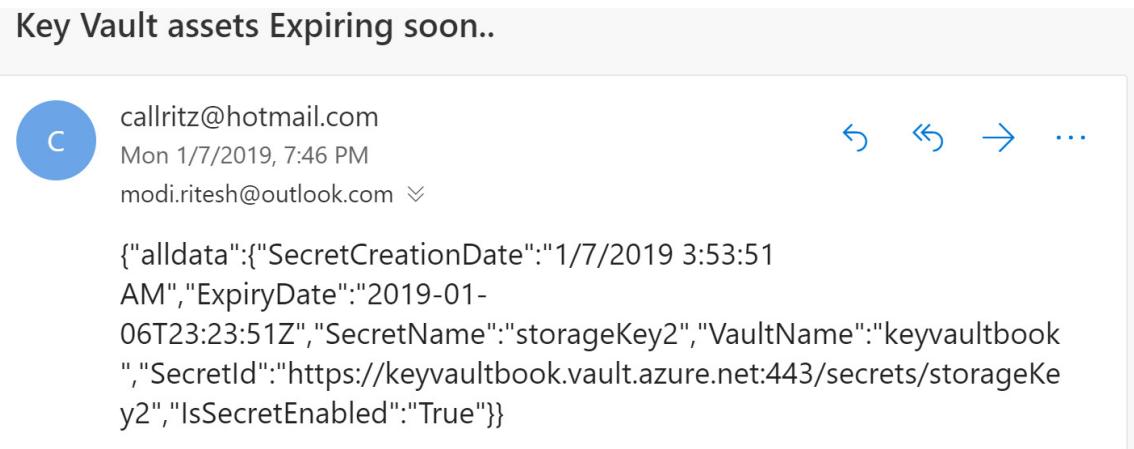


Figura 11.39: email recebido sobre as chaves que estão expirando

Neste exercício, tivemos um problema, arquitetamos uma solução e a implementamos. Isso é exatamente o que acontece na função de um arquiteto. Os clientes terão requisitos específicos e, com base neles, você deverá desenvolver uma solução. Nesse contexto, estamos concluindo este capítulo. Vamos recapitular rapidamente o que discutimos.

## Resumo

Este capítulo apresentou os Aplicativos Lógicos e demonstrou uma solução completa de ponta a ponta usando vários serviços do Azure. O capítulo concentrou-se principalmente na criação de uma arquitetura que integrou vários serviços do Azure para criar uma solução de ponta a ponta. Os serviços usados na solução foram Automação do Azure, Aplicativos Lógicos do Azure, Grade de Eventos do Azure, Azure Functions, SendGrid e Twilio. Esses serviços foram implementados por meio do portal do Azure e do PowerShell usando entidades de serviço como contas de serviço. O capítulo também mostrou uma série de maneiras para criar entidades de serviço com autenticação de senha e certificado.

Uma solução para um problema pode ser encontrada de várias maneiras. Você pode usar um gatilho do Outlook em um Aplicativo Lógico em vez do SendGrid. Haverá muitas soluções para um problema – o que vai depender de qual abordagem você está adotando. Quanto mais familiarizado estiver com os serviços, maior será o número de opções que você terá. No próximo capítulo, você conhecerá a importância dos eventos na arquitetura do Azure e de aplicações do Azure.





# 12

## Soluções de eventos de big data do Azure

Os eventos estão em toda parte. Qualquer atividade ou tarefa que altera o estado do item de trabalho gera um evento. Devido à falta de infraestrutura e à indisponibilidade de dispositivos acessíveis, antes não havia muita tração para a **Internet das Coisas (IoT)**. Historicamente, as organizações usavam ambientes hospedados de **provedores de serviços de Internet (ISPs)** que apenas tinham sistemas de monitoramento acima deles. Esses sistemas de monitoramento geraram eventos que eram poucos e distantes.

No entanto, com o surgimento da nuvem, as coisas estão mudando rapidamente. Com o aumento das implantações na nuvem, especialmente dos serviços **Plataforma como um Serviço (PaaS)**, as organizações não precisam mais de muito controle sobre o hardware e a plataforma, e, agora, toda vez que ocorre uma alteração em um ambiente, é gerado um evento. Com o surgimento de eventos na nuvem, a IoT ganhou muita proeminência, e os eventos começaram a roubar os holofotes.

Outro fenômeno recente durante esse período foi a rápida explosão de crescimento na disponibilidade de dados. A velocidade, a variedade e o volume de dados aumentaram, assim como a necessidade de soluções para armazenamento e processamento de dados. Várias soluções e plataformas surgiram, como o Hadoop, data lakes para armazenamento, data lakes para análise e serviços de aprendizado de máquina.

Além do armazenamento e da análise, há também a necessidade de serviços capazes de ingerir milhões e milhões de eventos e mensagens de várias fontes. Há também a necessidade de serviços que possam trabalhar em dados temporais, em vez de trabalhar em todo o instantâneo de dados. Por exemplo, dados de eventos/IoT são usados em aplicações que tomam decisões com base em dados em tempo real ou quase em tempo real, como sistemas de gerenciamento de tráfego ou sistemas que monitoram a temperatura.

O Azure fornece uma grande variedade de serviços que ajudam na captura e na análise de dados em tempo real de sensores. Neste capítulo, analisaremos alguns serviços de eventos no Azure, conforme listados aqui:

- Hubs de Eventos do Azure
- Azure Stream Analytics

Há outros serviços de eventos, como a Grade de Eventos do Azure, que não são abordados neste capítulo. No entanto, eles são amplamente abordados no Capítulo 10, *Serviços de integração do Azure com funções do Azure (Durable Functions e funções de proxy)*.

## Introdução a eventos

Eventos são componentes importantes no Azure e na arquitetura de aplicação do Azure. Os eventos estão em toda parte no ecossistema de software. Geralmente, qualquer ação realizada resulta em um evento que pode ser retido, e outras ações podem ser tomadas. Para levar adiante esta discussão, é importante primeiro entender os fundamentos dos eventos.

Os eventos ajudam a capturar o novo estado de um recurso de destino. Uma mensagem é uma notificação leve de uma condição ou uma alteração de estado. Os eventos são diferentes das mensagens. As mensagens estão relacionadas à funcionalidade de negócios, como enviar detalhes do pedido para outro sistema. Elas contêm dados brutos e podem ter um tamanho grande. Em comparação, os eventos são diferentes. Por exemplo, uma máquina virtual que está sendo interrompida é um evento. A Figura 12.1 demonstra essa transição do estado atual para o estado de destino:



Figura 12.1: transição de um estado devido a um evento

Os eventos podem ser armazenados no armazenamento durável, pois os dados e eventos históricos também podem ser usados para encontrar padrões que estão surgindo continuamente. Os eventos podem ser considerados dados transmitidos constantemente. Para capturar, ingerir e executar análises em um fluxo de dados, componentes de infraestrutura especiais que podem ler uma pequena janela de dados e fornecer insights conforme necessário, e é aí que o serviço do Stream Analytics entra em ação.

## Streaming de eventos

O processamento de eventos à medida que são ingeridos e transmitidos ao longo de uma janela de tempo fornece insights em tempo real sobre dados. A janela de tempo pode ser de 15 minutos ou uma hora. A janela é definida pelo usuário e depende dos insights que devem ser extraídos dos dados. Compras com cartão de crédito por exemplo: milhões de compras com cartões de crédito acontecem a cada minuto, e a detecção de fraudes pode ser feita em eventos transmitidos por uma janela de tempo de um ou dois minutos.

O streaming de eventos refere-se a serviços que podem aceitar dados como e quando surgem, em vez de aceitá-los periodicamente. Por exemplo, os fluxos de eventos devem ser capazes de aceitar informações de temperatura de dispositivos como e quando enviá-las, em vez de fazer com que os dados esperem em uma fila ou um ambiente de preparo.

O streaming de eventos também tem a capacidade de consultar dados em trânsito. Estes são dados temporais que são armazenados por um tempo, e as consultas ocorrem nos dados em movimento. Portanto, os dados não são estacionários. Esse recurso não está disponível em outras plataformas de dados, que podem consultar apenas dados armazenados, e não dados temporais que acabaram de ser ingeridos.

Os serviços de streaming de eventos devem ser capazes de escalar facilmente para aceitar milhões ou até bilhões de eventos. Eles devem estar altamente disponíveis para que as fontes possam enviar eventos e dados para eles a qualquer momento. A ingestão de dados em tempo real e a capacidade de trabalhar com esses dados, em vez de dados armazenados em um local diferente, é fundamental para o streaming de eventos.

Mas quando já temos tantas plataformas de dados com recursos avançados de execução de consulta, por que precisamos de streaming de eventos? Uma das principais vantagens do streaming de eventos é que ele fornece insights e informações em tempo real cuja utilidade depende do tempo. As mesmas informações encontradas após alguns minutos ou horas podem não ser tão úteis. Vamos considerar alguns cenários em que trabalhar com dados de entrada é muito importante. Esses cenários não podem ser resolvidos de forma eficaz e eficiente por plataformas de dados existentes:

- **Detecção de fraude de cartão de crédito:** deve acontecer quando uma transação fraudulenta acontecer.
- **Informações de telemetria dos sensores:** no caso de dispositivos de IoT que enviam informações essenciais sobre seus ambientes, o usuário deve ser notificado como e quando uma anomalia for detectada.
- **Painéis ativos:** o streaming de eventos é necessário para criar painéis que mostram informações em tempo real.
- **Telemetria do ambiente do datacenter:** permitirá que o usuário saiba sobre invasões, violações de segurança, falhas de componentes, e muito mais.

Há muitas possibilidades de aplicação de streaming de eventos em uma empresa, e é de extrema importância.

## Hubs de Eventos

Hubs de Eventos é uma plataforma de streaming que fornece funcionalidades relacionadas à ingestão e ao armazenamento de eventos relacionados a streaming.

Ela pode ingerir dados de uma variedade de fontes. Elas podem ser sensores de IoT ou qualquer aplicação que use o **Kit de Desenvolvimento de Software (SDK)** dos Hubs de Eventos. Esse serviço é compatível com vários protocolos para ingerir e armazenar dados. Esses protocolos são padrão da indústria e incluem o seguinte:

- **HTTP:** é uma opção sem estado e não requer uma sessão ativa.
- **Advanced Messaging Queuing Protocol (AMQP):** requer uma sessão ativa (ou seja, uma conexão estabelecida usando soquetes) e funciona com os protocolos **Transport Layer Security (TLS)** e **Secure Socket Layer (SSL)**.
- **Apache Kafka:** é uma plataforma de streaming distribuída semelhante ao Stream Analytics. No entanto, o Stream Analytics foi criado para executar análises em tempo real em vários fluxos de dados de várias fontes, como sensores de IoT e sites.

Hubs de Eventos é um serviço de ingestão de eventos. Ele não pode consultar os resultados da consulta de uma solicitação e saída para outro local. Essa é a responsabilidade do Stream Analytics, que é abordado na próxima seção.

Para criar uma instância dos Hubs de Eventos no portal, pesquise Hubs de Eventos no Marketplace e clique em **Criar**. Selecione uma assinatura e um grupo de recursos existente (ou crie um novo). Forneça um nome para o namespace dos Hubs de Eventos, a região de preferência do Azure para hospedá-lo, a camada de preços (básica ou padrão, explicada mais adiante) e o número de unidades de taxa de transferência (explicado posteriormente):

**Create Namespace**

Event Hubs

Basics    Features    Tags    Review + create

**PROJECT DETAILS**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* RiteshSubscription

Resource group \* Create new

**INSTANCE DETAILS**

Enter required settings for this namespace, including a price tier and configuring the number of throughput units.

Namespace name \* eventhub.servicebus.windows.net

Location \* West Central US

Pricing tier (View full pricing details) \* Basic

Throughput Units \* 1

**Review + create**    < Previous    Next: Features >

Figura 12.2: criando um namespace dos Hubs de Eventos

Os Hubs de Eventos, como um serviço PaaS no Azure, é altamente distribuído, disponível e escalável.

Os Hubs de Eventos vêm com as duas SKUs ou as camadas de preço a seguir:

- **Básico:** é fornecido com um grupo de consumidores e pode reter mensagens por 1 dia. Ele pode ter um máximo de 100 conexões intermediadas.
- **Padrão:** é fornecido com um máximo de 20 grupos de consumidores e pode reter mensagens por 1 dia com armazenamento adicional por 7 dias. Ele pode ter um máximo de 1.000 conexões intermediadas. Também é possível definir políticas nesta SKU.

A Figura 12.3 mostra as diferentes SKUs disponíveis ao criar um novo namespace dos Hubs de Eventos. Ele fornece uma opção para escolher uma camada de preços apropriada, juntamente com outros detalhes importantes:

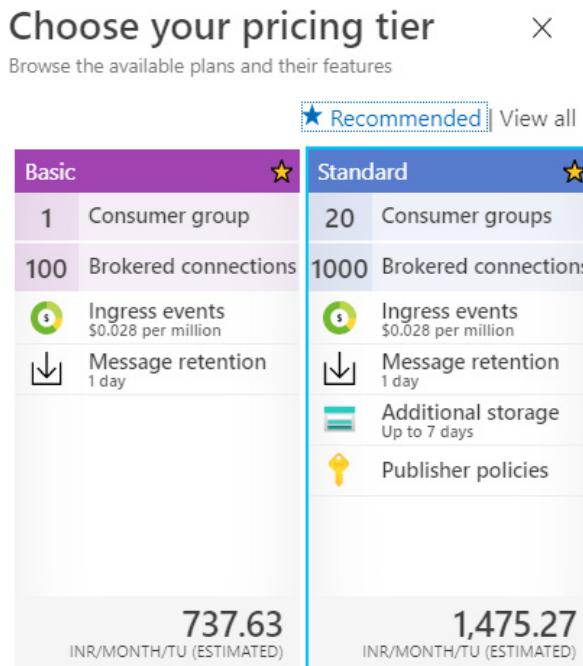


Figura 12.3: SKUs dos Hubs de Eventos

A taxa de transferência também pode ser configurada no nível do namespace. Os namespaces são contêineres que consistem em vários hubs de eventos na mesma assinatura e região. A taxa de transferência é calculada como **unidades de taxa de transferência (TUs)**. Cada TU fornece:

- Até 1 MB por segundo de entrada ou um máximo de 1.000 eventos de entrada e operações de gerenciamento por segundo.
- Até 2 MB por segundo de saída ou um máximo de 4.096 eventos de saída e operações de gerenciamento por segundo.
- Até 84 GB de armazenamento.

As TUs podem variar de 1 a 20 e são cobradas por hora.

É importante observar que a SKU não pode ser alterada após o provisionamento de um namespace dos Hubs de Eventos. A devida consideração e planejamento devem ser adotados antes de selecionar uma SKU. O processo de planejamento deve incluir o planejamento do número de grupos de consumidores necessários e o número de aplicações interessadas em ler o hub de eventos.

Além disso, a SKU padrão não está disponível em todas as regiões. Sua disponibilidade deve ser verificada no momento da criação e da implementação do hub de eventos. A URL para verificar a disponibilidade da região é <https://azure.microsoft.com/global-infrastructure/services/?products=event-hubs>.

## Arquitetura dos Hubs de Eventos

Há três componentes principais da arquitetura dos Hubs de Eventos: **Produtores de Eventos**, **Hub de Eventos** e **Consumidor de Eventos**, conforme mostrado no diagrama a seguir:



Figura 12.4: arquitetura dos Hubs de Eventos

**Os Produtores de Eventos** geram eventos e os enviam ao **Hub de Eventos**. O **Hub de Eventos** armazena os eventos ingeridos e fornece esses dados ao **Consumidor de Eventos**. O **Consumidor de Eventos** é o que está interessado nesses eventos e se conecta ao **Hub de Eventos** para obter os dados.

Não é possível criar hubs de eventos sem um namespace dos Hubs de Eventos. O namespace de Hubs de Eventos funciona como um contêiner e pode hospedar vários hubs de eventos. Cada namespace de Hubs de Eventos fornece um ponto de extremidade exclusivo baseado em REST que é consumido pelos clientes para enviar dados aos Hubs de Eventos. Esse namespace é o mesmo namespace necessário para artefatos do Barramento de Serviço, como tópicos e filas.

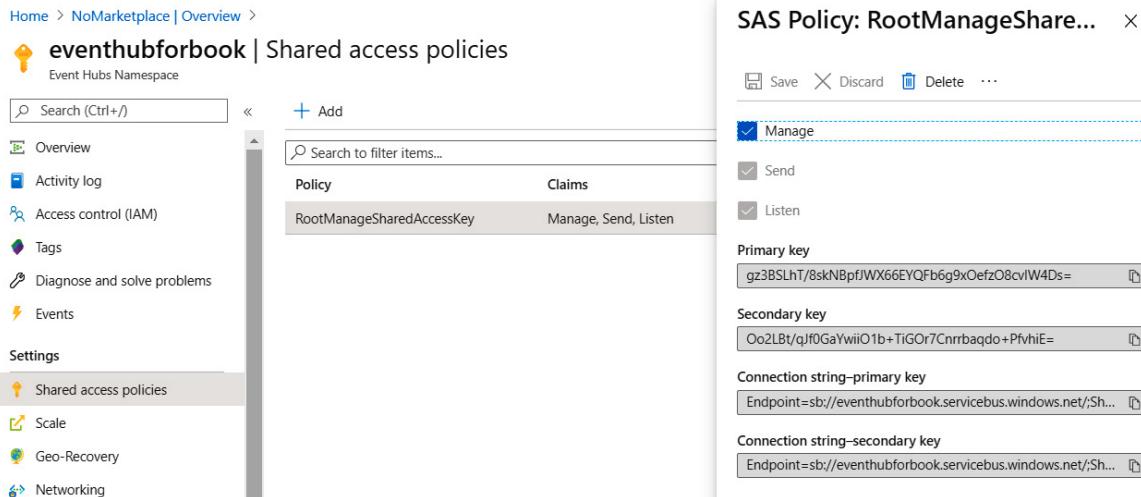
A cadeia de conexão de um namespace dos Hubs de Eventos é composta pela respectiva URL, nome da política e chave. Uma cadeia de conexão de exemplo é mostrada no seguinte bloco de código:

```
Endpoint=sb://demoeventhubsbook.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=M/E4eeBsr7DALXcvw6ziFq1SDNbFX6E49Jfti8CRkbA=
```

Essa cadeia de conexão pode ser encontrada no item de menu **Assinatura de Acesso Compartilhado (SAS)** do namespace. Pode haver várias políticas definidas para um namespace, cada uma com diferentes níveis de acesso ao namespace. Os três níveis de acesso são os seguintes:

- **Gerenciar:** pode gerenciar o hub de eventos de uma perspectiva administrativa. Também tem direitos para enviar e ouvir eventos.
- **Enviar:** pode gravar eventos em Hubs de Eventos.
- **Ouvir:** pode ler eventos de Hubs de Eventos.

Por padrão, a política **RootManageSharedAccessKey** é criada ao criar um hub de eventos, conforme mostrado na Figura 12.5. As políticas ajudam na criação de controle de acesso granular em Hubs de Eventos. A chave associada a cada política é usada pelos consumidores para determinar sua identidade. Políticas adicionais também podem ser criadas com qualquer combinação dos três níveis de acesso mencionados anteriormente:



The screenshot shows the Azure portal interface for managing an Event Hub namespace named "eventhubforbook". The left sidebar includes links for Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings (with Shared access policies selected), Scale, Geo-Recovery, and Networking. The main content area displays the "Shared access policies" section for the "eventhubforbook" hub. It features a search bar, an "Add" button, and a table with columns "Policy" and "Claims". One policy is listed: "RootManageSharedAccessKey" with claims "Manage, Send, Listen". On the right, a detailed view of the "RootManageSharedAccessKey" policy is shown, including sections for Primary key (value: "gz3BSLhT/8skNBpfjWX66EYQFb6g9xOefzO8cvIW4Ds="), Secondary key (value: "Oo2LBt/qJf0GaYwiiO1b+TiGOR7Cnrrbaqdo+PfvhiE="), Connection string-primary key (value: "Endpoint=sb://eventhubforbook.servicebus.windows.net/;Sh..."), and Connection string-secondary key (value: "Endpoint=sb://eventhubforbook.servicebus.windows.net/;Sh..."). Action buttons at the top right include Save, Discard, Delete, and more.

**Figura 12.5: políticas de acesso compartilhado nos Hubs de Eventos**

Os hubs de eventos podem ser criados no serviço de namespace de Hubs de Eventos, realizando as seguintes ações:

1. Clique em **Hubs de Eventos** no menu à esquerda e clique em **+ Hub de Eventos** na tela resultante:

The screenshot shows the Azure portal interface for managing Event Hubs. At the top, the URL is [Home > NoMarketplace | Overview >](#). The main title is **eventhubforbook | Event Hubs**, with a subtitle **Event Hubs Namespace**. On the left sidebar, there are several sections: **Overview**, **Activity log**, **Access control (IAM)**, **Tags**, **Diagnose and solve problems**, and **Events**. Below these are sections for **Settings** (Shared access policies, Scale, Geo-Recovery, Networking, Encryption, Properties, Locks, Export template) and **Entities** (Event Hubs). The **Event Hubs** item is highlighted with a gray background. On the right side, there is a search bar labeled "Search to filter items..." and a "Name" input field containing the text "testeventhub". Above the input field are buttons for "+ Event Hub" and "Refresh".

Figura 12.6: criando um hub de eventos no portal do Azure

2. Em seguida, forneça valores para os campos **Contagem de Partições** e **Retenção de Mensagem**, juntamente com o nome de sua escolha. Em seguida, selecione **Desativada** para **Captura**, conforme demonstrado na Figura 12.7:

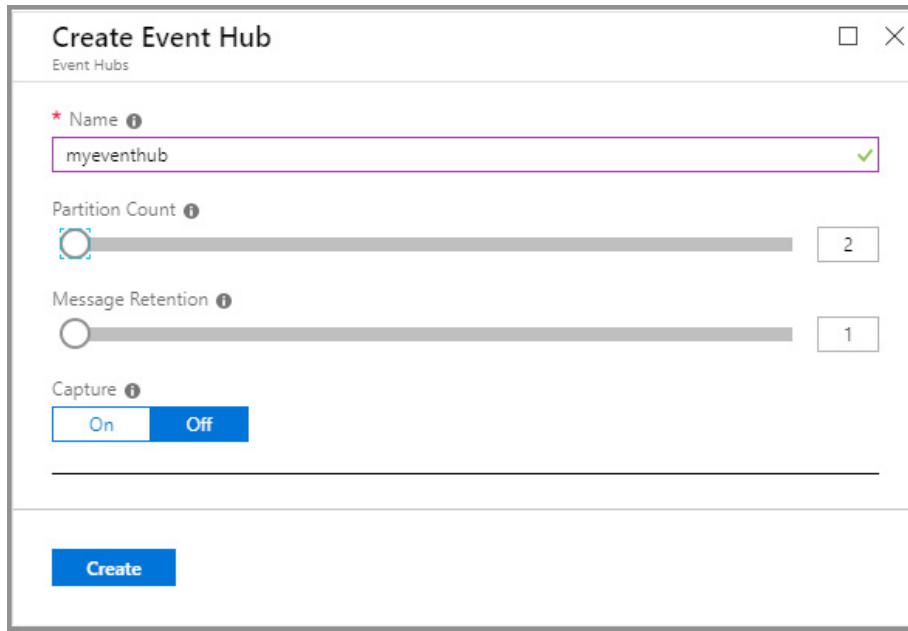


Figura 12.7: criando um novo hub de eventos

Depois que o hub de eventos for criado, você verá na lista de hubs de eventos, conforme mostrado na Figura 12.8:

The screenshot shows the Azure portal interface. In the top left, there's a breadcrumb trail: 'Home > eventrg >'. The main title is 'eventhubforbook | Event Hubs'. On the left, there's a sidebar with icons for Scale, Geo-Recovery, Networking, Encryption, Properties, Locks, and Export template. Below that is a section titled 'Entities' with a 'Event Hubs' icon. The main area has a search bar 'Search (Ctrl+ /)' and a 'Refresh' button. There's also a 'Search to filter items...' bar. A table lists the created hub: myeventhub, Active, 2 days, 2. The 'Event Hubs' tab is highlighted in the navigation bar.

Name	Status	Message Retention	Partition Count
myeventhub	Active	2 days	2

Figura 12.8: lista de hubs de eventos criados

Os Hubs de Eventos também permitem o armazenamento de eventos em uma conta de armazenamento ou data lake diretamente usando um recurso conhecido como Captura.

A Captura ajuda no armazenamento automático de dados ingeridos para uma conta de armazenamento do Azure ou um data lake do Azure. Esse recurso garante que a ingestão e o armazenamento de eventos ocorra em uma única etapa, em vez de transferir dados para o armazenamento como uma atividade separada:

Capture ⓘ

On  Off

Note: Enabling Capture will result in additional charges to this account. Learn more about our pricing here.

Time window (minutes)

Size window (MB)

Do not emit empty files when no events occur during the Capture time window

Capture Provider

Azure Storage Account

Azure Storage Container \*

Select Container

Storage Account

Sample Capture file name formats

{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

Capture file name format ⓘ

{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

e.g. eventhubforbook/undefined/0/2020/4/23/7/48/11

Figura 12.9: opções do recurso Captura

Políticas separadas podem ser atribuídas a cada hub de eventos adicionando uma nova política no nível do hub de eventos.

Depois de criar a política, a cadeia de conexão está disponível no item de menu esquerdo **Assinatura de Acesso Seguro** no portal do Azure.

Como um namespace pode consistir em vários hubs de eventos, a cadeia de conexão para um hub de eventos individual será semelhante ao bloco de códigos a seguir. A diferença aqui é o valor da chave e a adição de **EntityPath** com o nome do hub de eventos:

```
Endpoint=sb://azurertwittereventdata.servicebus.windows  
=rxEu5K4Y2qsi5wEe0Ku0vRnhtgW8xW35UBex4V1IKqg=;EntityPath=myeventhub
```

Tivemos que manter a opção **Captura** definida como **Desativada** enquanto criamos o hub de eventos, e ela pode ser ativada após a criação do hub de eventos. Isso ajuda a salvar eventos em uma conta do Azure Data Lake Storage ou do Armazenamento de Blobs do Azure automaticamente. A configuração para o intervalo de tamanho e tempo é mostrada na Figura 12.10:

**myeventhub (eventhubforbook/myeventhub) | Capture**

Event Hubs Instance

Search (Ctrl+ /) Save changes Discard

**Capture**

On Off

Note: Enabling Capture will result in additional charges to this account. Learn more about our pricing [here](#).

**Time window (minutes)** 5

**Size window (MB)** 300

Do not emit empty files when no events occur during the Capture time window

**Capture Provider** Azure Storage Account

**Azure Storage Container \*** Select Container

**Storage Account**

**Sample Capture file name formats** {Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

**Capture file name format** {Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}

e.g. eventhubforbook/myeventhub/0/2020/05/12/7/36/41

Figura 12.10: selecionando o tamanho e o intervalo de tempo para capturar eventos

Nós não abordamos os conceitos de partições e opções de retenção de mensagens ao criar hubs de eventos.

A partição é um conceito importante relacionado à escalabilidade de qualquer armazenamento de dados. Os eventos são retidos nos hubs de eventos por um período específico. Se todos os eventos forem armazenados no mesmo armazenamento de dados, será extremamente difícil escalar esse armazenamento de dados. Todo produtor de evento se conectará ao mesmo armazenamento de dados e enviará seus eventos a ele. Compare isso com um armazenamento de dados que pode particionar os mesmos dados em vários armazenamentos de dados menores, cada um identificado exclusivamente com um valor.

O armazenamento de dados menor é chamado de **partição**, e o valor que define a partição é conhecido como **chave de partição**. Essa chave de partição é parte dos dados do evento.

Agora, os produtores de eventos podem se conectar ao hub de eventos e, com base no valor da chave de partição, o hub de eventos armazenará os dados em uma partição apropriada. Isso permitirá que o hub de eventos ingira vários eventos ao mesmo tempo em paralelo.

Decidir sobre o número de partições é um aspecto crucial da escalabilidade de um hub de eventos. A Figura 12.11 mostra que os dados ingeridos são armazenados na partição apropriada internamente pelos Hubs de Eventos usando a chave de partição:

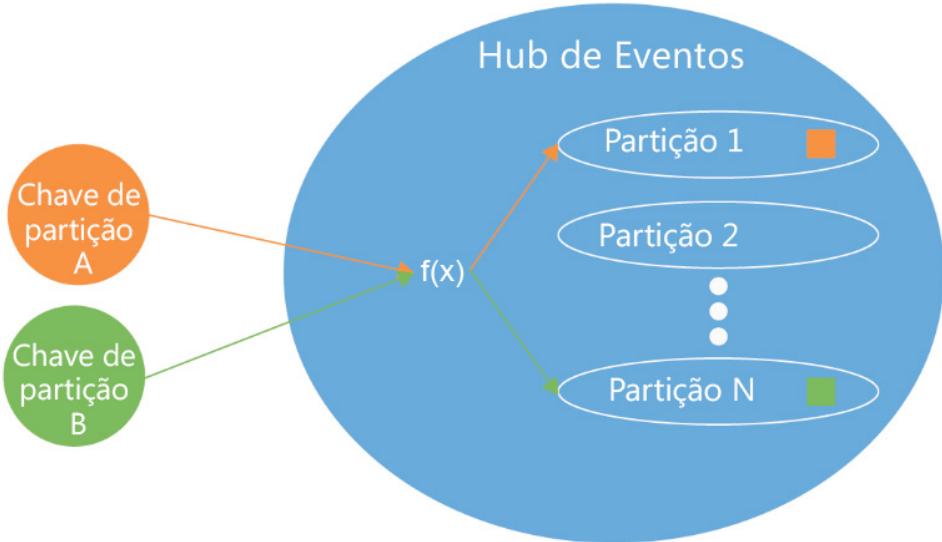


Figura 12.11: particionando em um hub de eventos

É importante entender que uma partição pode ter várias chaves. O usuário decide quantas partições são necessárias, e o hub de eventos decide internamente a melhor maneira de alocar as chaves de partição entre eles. Cada partição armazena dados de forma ordenada usando um carimbo de data/hora e eventos mais recentes são anexados ao final da partição.

É importante observar que não é possível alterar o número de partições depois que o hub de eventos é criado.

Também é importante lembrar que as partições ajudam a trazer paralelismo e simultaneidade para aplicações que leem os eventos. Por exemplo, se houver 10 partições, 10 leitores paralelos poderão ler os eventos sem qualquer degradação na performance.

Retenção de mensagens refere-se ao período pelo qual os eventos devem ser armazenados. Após o término do período de retenção, os eventos são descartados.

## Grupos de consumidores

Os consumidores são aplicações que leem eventos de um hub de eventos. Os grupos de consumidores são criados para que os consumidores se conectem a ele a fim de ler os eventos. Pode haver vários grupos de consumidores para um hub de eventos, e cada um deles tem acesso a todas as partições de um hub de eventos. Cada grupo de consumidores faz uma consulta sobre os eventos nos hubs de eventos. As aplicações podem usar grupos de consumidores e cada aplicação receberá uma exibição diferente dos eventos dos hubs de eventos. Um grupo de consumidores **\$default** padrão é criado ao criar um hub de eventos. É recomendável para um consumidor estar associado a um grupo de consumidores para obter excelente performance. No entanto, é possível ter cinco leitores em cada partição em um grupo de consumidores:

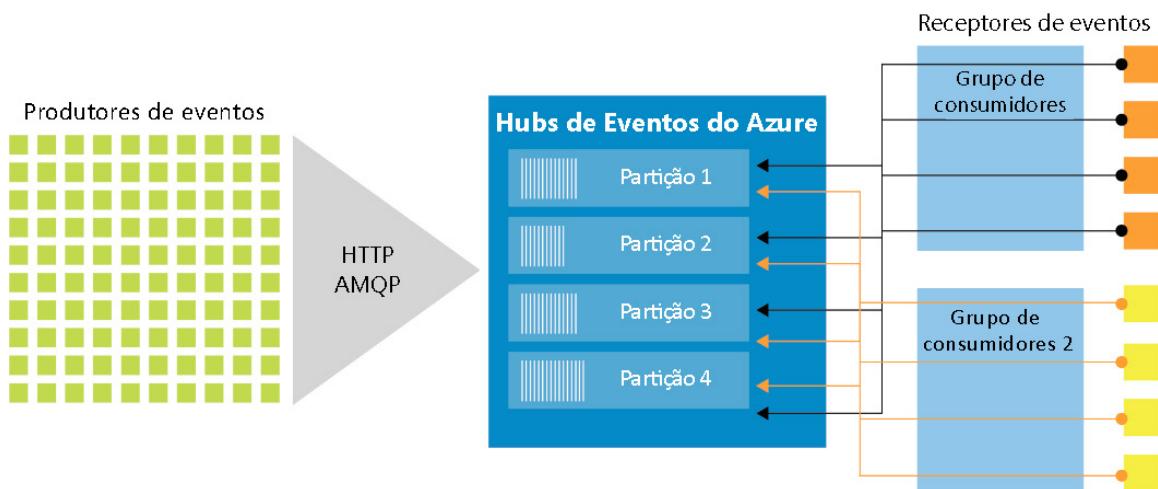


Figura 12.12: receptores de eventos em um grupo de consumidores

Agora que você compreende os grupos de consumidores, é hora de nos aprofundarmos no conceito de taxa de transferência dos Hubs de Eventos.

## Taxa de transferência

As partições ajudam com a escalabilidade, enquanto a taxa de transferência ajuda com a capacidade por segundo. Então, qual é a capacidade em termos de Hubs de Eventos? É a quantidade de dados que podem ser manipulados por segundo.

Nos Hubs de Eventos, uma única TU permite o seguinte:

- 1 MB de dados de ingestão por segundo ou 1.000 eventos por segundo (o que acontecer primeiro)
- 2 MB de dados de saída por segundo ou 4.096 eventos por segundo (o que acontecer primeiro)

A opção de inflação automática ajuda a aumentar a taxa de transferência automaticamente se o número de eventos de entrada/saída ou o tamanho total de entrada/saída ultrapassar um limite. Em vez de limitar, a taxa de transferência aumentará e diminuirá. A configuração da taxa de transferência no momento da criação do namespace é mostrada na Figura 12.13. Novamente, deve-se considerar com cuidado ao decidir sobre as TUs:

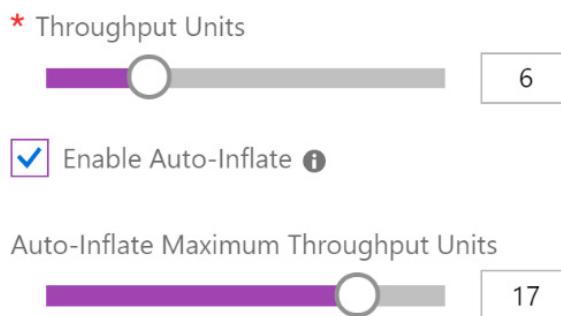


Figura 12.13: selecionando as TUs juntamente com a inflação automática

## Uma cartilha sobre o Stream Analytics

Os Hubs de Eventos é uma plataforma de fluxo de dados altamente escalável. Portanto, precisamos de outro serviço que possa processar esses eventos como um fluxo, e não apenas como dados armazenados. O Stream Analytics ajuda no processamento e no exame de um fluxo do big data, e os trabalhos do Stream Analytics ajudam a executar o processamento de eventos.

O Stream Analytics pode processar milhões de eventos por segundo e é muito fácil começar com ele. O Azure Stream Analytics é uma PaaS totalmente gerenciada pelo Azure. Os clientes do Stream Analytics não precisam gerenciar o hardware e a plataforma subjacentes.

Cada trabalho consiste em várias entradas, saídas e uma consulta, que transforma os dados recebidos em novos resultados. Toda a arquitetura do Stream Analytics é mostrada na Figura 12.14:

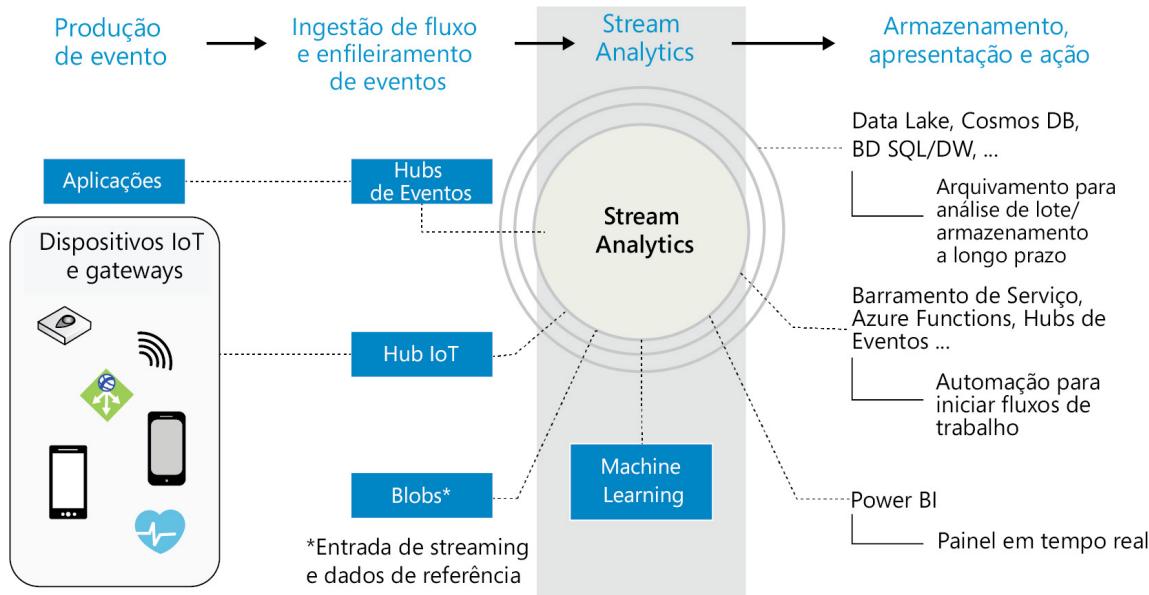


Figura 12.14: arquitetura do Azure Stream Analytics

Na Figura 12.14, as fontes de eventos são exibidas à extrema esquerda. Estas são as fontes que produzem os eventos. Podem ser dispositivos IoT, aplicações personalizadas escritas em qualquer linguagem de programação ou eventos provenientes de outras plataformas do Azure, como Log Analytics ou Application Insights.

Esses eventos devem primeiro ser ingeridos no sistema, e existem vários serviços do Azure que podem ajudar a ingerir esses dados. Já vimos os Hubs de Eventos e como eles ajudam na ingestão de dados. Há outros serviços, como Hub IoT, que também ajudam na ingestão de dados específicos do dispositivo e do sensor. O Hub IoT e a ingestão são abordados detalhadamente no Capítulo 11, Projetar soluções IoT. Esses dados ingeridos passam por processamento à medida que chegam em um fluxo, e esse processamento é feito usando o Stream Analytics. A saída do Stream Analytics pode ser alimentada em uma plataforma de apresentação, como o Power BI, para mostrar dados em tempo real para os stakeholders, ou uma plataforma de armazenamento, como o Cosmos DB, o Data Lake Storage ou o Armazenamento do Azure, nos quais os dados podem ser lidos e executados posteriormente pelas filas do Azure Functions e do Barramento de Serviço.

O Stream Analytics ajuda a coletar insights de dados ingeridos em tempo real em uma janela de tempo e ajuda na identificação de padrões.

Ele faz isso por meio de três tarefas diferentes:

- **Entrada:** os dados devem ser ingeridos no processo de análise. Os dados podem ser originados dos Hubs de Eventos, do Hub IoT ou do Armazenamento de Blobs do Azure. Várias entradas de referência separadas usando uma conta de armazenamento e Banco de Dados SQL podem ser usadas para pesquisa de dados em consultas.
- **Consulta:** é aqui que o Stream Analytics faz o trabalho principal de analisar os dados ingeridos e extrair insights e padrões significativos. Ele faz isso com a ajuda de funções JavaScript definidas pelo usuário, agregados JavaScript definidos pelo usuário, o Azure Machine Learning e o Azure Machine Learning Studio.
- **Saída:** o resultado das consultas pode ser enviado para vários tipos diferentes de destinos, e os mais importantes entre eles são o Cosmos DB, o Power BI, o Synapse Analytics, o Data Lake Storage e o Functions:

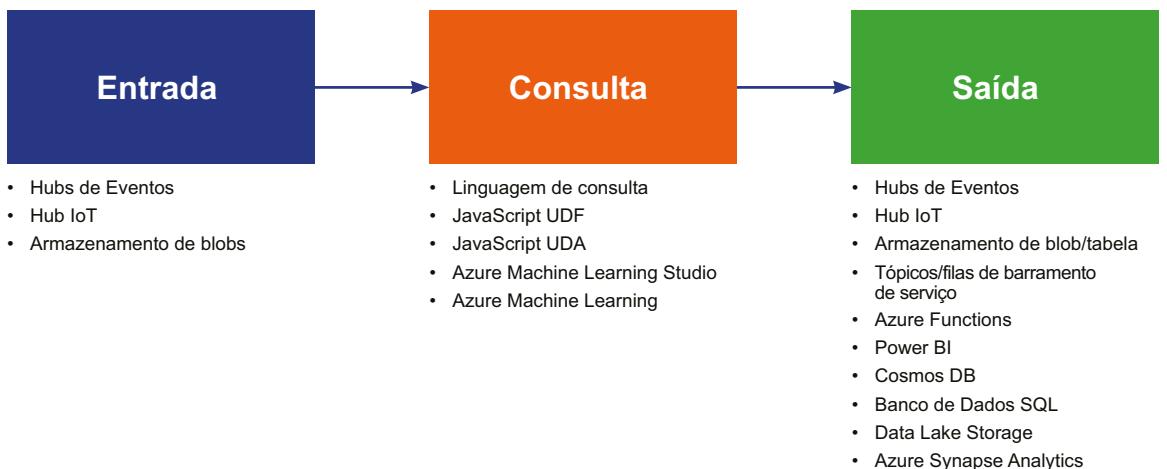


Figura 12.15: processo do Stream Analytics

O Stream Analytics é capaz de ingerir milhões de eventos por segundo e pode executar consultas sobre eles.

Os dados de entrada são compatíveis com qualquer um dos três formatos a seguir:

- **JavaScript Object Notation (JSON):** um formato leve e baseado em texto simples, legível para humanos. Consiste em pares nome-valor. Um exemplo de um evento JSON é o seguinte:

```
{
  "SensorId" : 2,
  "humidity" : 60,
  "temperature" : 26C
}
```

- **Valores separados por vírgulas (CSV)**: também são valores de texto simples, separados por vírgulas. Um exemplo de CSV é mostrado na Figura 12.16. A primeira linha é o cabeçalho contendo três campos seguidos por duas linhas de dados:

```
SensorID, humidity, temperature
2,60,26C
3,65,31C
```

Figura 12.16: Valores de texto simples

- **Avro**: formato semelhante ao JSON. No entanto, ele é armazenado em um formato binário, em vez de um formato de texto.

```
{
    "firstname": "Ritesh",
    "lastname": "Modi",
    "email": "ritesh.modi@outlook.com"
}
```

No entanto, isso não significa que o Stream Analytics só pode ingerir dados usando esses três formatos. Ele também pode criar desserializadores baseados em .NET personalizados, usando o qual qualquer formato de dados pode ser ingerido, dependendo da implementação dos desserializadores. As etapas que você pode seguir para criar um desserializador personalizado estão disponíveis em <https://docs.microsoft.com/azure/stream-analytics/custom-deserializer-examples>.

O Stream Analytics não só pode receber eventos, como também fornece recursos avançados de consulta para os dados que recebe. As consultas podem extrair insights importantes dos fluxos de dados temporais e gerá-los.

Conforme mostrado na Figura 12.17, há uma conjunto de dados de entrada e um de saída. A consulta move os eventos da entrada para a saída. A cláusula **INTO** refere-se ao local de saída, e a cláusula **FROM** refere-se ao local de entrada. As consultas são muito semelhantes às consultas SQL. Portanto, a curva de aprendizado não é muito alta para programadores de SQL:

```
1 SELECT
2   *
3 INTO
4   [TwitterData]
5 FROM
6   [TwitterIncomingEvents]
```

Figura 12.17: consulta do Stream Analytics para receber dados do Twitter

Os Hubs de Eventos fornece mecanismos para enviar saídas de consultas para destinos. No momento da redação, o Stream Analytics oferece suporte a vários destinos para eventos e saídas de consulta, conforme mostrado anteriormente.

Também é possível definir funções personalizadas que podem ser reutilizadas nas consultas. Há quatro opções fornecidas para definir funções personalizadas.

- Azure Machine Learning
- Funções JavaScript definidas pelo usuário
- Agregados JavaScript definidos pelo usuário
- Azure Machine Learning Studio

## Ambiente de hospedagem

Os trabalhos do Stream Analytics podem ser executados em hosts que estão em execução na nuvem ou podem ser executados em dispositivos do IoT Edge. Os dispositivos do IoT Edge são dispositivos que estão perto de sensores IoT, em vez de na nuvem. A Figura 12.18 mostra o painel **Novo trabalho do Stream Analytics**:

The screenshot shows the 'New Stream Analytics job' configuration page. It includes the following fields:

- Job name:** A text input field labeled 'Enter job name'.
- Subscription:** A dropdown menu set to 'RiteshSubscription'.
- Resource group:** A dropdown menu labeled 'Select existing...' with a 'Create new' link below it.
- Location:** A dropdown menu set to 'East US'.
- Hosting environment:** A button group with 'Cloud' selected and 'Edge' as an option.
- Streaming units (1 to 120):** A slider control showing a value of 3.

Figura 12.18: criando um novo trabalho do Stream Analytics

Vamos conferir unidades de streaming em detalhes.

## Unidades de streaming

Na Figura 12.18, você pode ver que a única configuração exclusiva para o Stream Analytics é unidades de streaming. Unidades de streaming referem-se aos recursos (isto é, CPU e memória) que são atribuídos para executar um trabalho do Stream Analytics. As unidades de streaming mínimas e máximas são 1 e 120, respectivamente.

As unidades de streaming devem ser pré-alocadas de acordo com a quantidade de dados e do número de consultas executadas nesses dados. Caso contrário, o trabalho falhará.

É possível aumentar e diminuir as unidades de streaming no portal do Azure.

## Uma aplicação de exemplo usando os Hubs de Eventos e o Stream Analytics

Nesta seção, estaremos criando uma aplicação de exemplo que consiste em vários serviços do Azure, incluindo os Aplicativos Lógicos do Azure, os Hubs de Eventos do Azure, o Armazenamento do Azure e o Azure Stream Analytics.

Nesta aplicação de exemplo, estaremos lendo todos os tweets que contenham a palavra "Azure" e armazenando-os em uma conta de armazenamento do Azure.

Para criar essa solução, primeiro precisamos provisionar todos os recursos necessários.

## Provisionar um novo grupo de recursos

Navegue até o portal do Azure, faça logon usando credenciais válidas, clique em **+ Criar um recurso** e procure **Grupo de recursos**. Selecione **Grupo de recursos** nos resultados da pesquisa e crie um novo grupo de recursos. Em seguida, forneça um nome e escolha um local apropriado. Observe que todos os recursos devem ser hospedados no mesmo grupo de recursos e local para que seja fácil excluí-los:

The screenshot shows the 'Create a resource group' wizard in the Azure portal. The top navigation bar includes 'Home > New > Marketplace > Everything > Resource group > Create a resource group'. The main title is 'Create a resource group'. Below it, there are three tabs: 'Basics' (selected), 'Tags', and 'Review + Create'. The 'PROJECT DETAILS' section contains two required fields: 'Subscription' (set to 'RiteshSubscription') and 'Resource group' (set to 'TwitterAnalysis'). The 'RESOURCE DETAILS' section contains one required field: 'Region' (set to 'West Europe').

Figura 12.19: provisionando um novo grupo de recursos no portal do Azure

Em seguida, criaremos um namespace dos Hubs de Eventos.

## Criando um namespace dos Hubs de Eventos

Clique em **+ Criar um recurso** e procure **Hubs de Eventos**. Selecione **Hubs de Eventos** nos resultados da pesquisa e crie um novo hub de eventos. Em seguida, forneça um nome e local e selecione uma assinatura com base no grupo de recursos que foi criado anteriormente. Selecione **Padrão** como a camada de preços e também selecione **Habilitar Inflação Automática**, conforme mostrado na Figura 12.20:

The screenshot shows the 'Create Namespace' wizard for Azure Event Hubs. The steps are as follows:

- Name:** AzureTwitterEventData (highlighted with a red border). A message below it says: "Namespace already exists. Enter a different name."
- Pricing tier:** Standard (20 Consumer groups, 1000 B...)
- Enable Kafka:** Unchecked
- Make this namespace zone redundant:** Unchecked
- Subscription:** Visual Studio Enterprise
- Resource group:** TwitterAnalysis (with a 'Create new' link)
- Location:** West Europe
- Throughput Units:** A slider set to 1, with a text input field next to it showing the value 1.
- Enable Auto-Inflate:** Checked (indicated by a checked checkbox)
- Auto-Inflate Maximum Throughput Units:** A slider set to 1, with a text input field next to it showing the value 1.
- Create** button at the bottom.

Figura 12.20: criando um namespace dos Hubs de Eventos

Neste momento, um namespace dos Hubs de Eventos deve ter sido criado. É um pré-requisito ter um namespace antes que um hub de eventos possa ser criado. A próxima etapa é provisionar um hub de eventos.

## Criar um hub de eventos

No serviço de namespace dos Hubs de Eventos, clique em **Hubs de Eventos** no menu à esquerda e, em seguida, clique em **+ Hubs de eventos** para criar um novo hub de eventos. Nomeie-o como **azuretwitterdata** e forneça um número ideal de partções e um valor de **Retenção de Mensagem**:

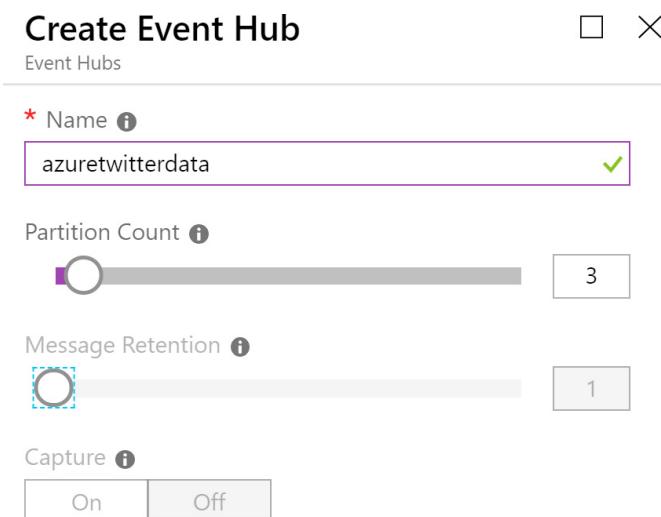


Figura 12.21: criando o hub de eventos azuretwitterdata

Após essa etapa, você terá um hub de eventos que pode ser usado para enviar dados de eventos, que são armazenados em um armazenamento durável, como um data lake ou uma conta de armazenamento do Azure, para serem usados pelos serviços downstream.

## Provisionar um aplicativo lógico

Depois que o grupo de recursos for provisionado, clique em **+ Criar um recurso** e procure **Aplicativos Lógicos**. Selecione **Aplicativos Lógicos** nos resultados da pesquisa e crie um novo aplicativo lógico. Em seguida, forneça um nome e local e selecione uma assinatura com base no grupo de recursos criado anteriormente. É recomendável habilitar **Análise do Log**. Os Aplicativos Lógicos são abordados em mais detalhes no Capítulo 11, Soluções do Azure usando Aplicativos Lógicos do Azure, Grade de Eventos e Functions. O aplicativo lógico é responsável pela conexão ao Twitter usando uma conta e a obtenção de todos os tweets com **Azure** neles:

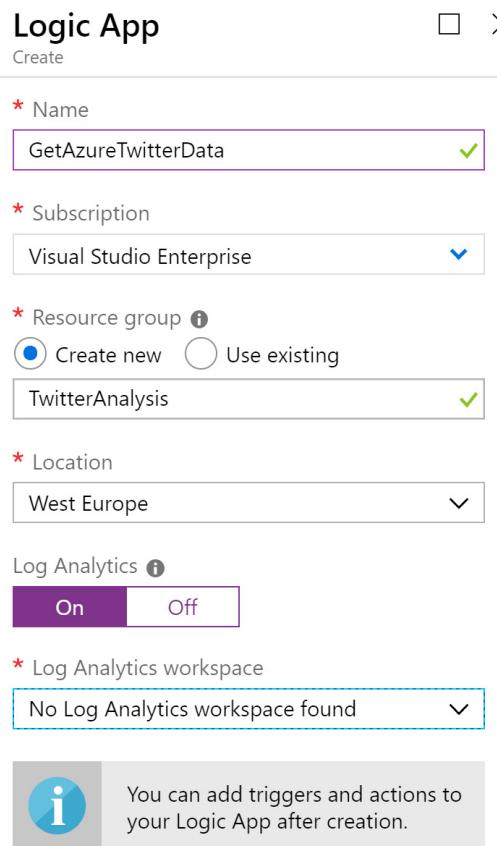
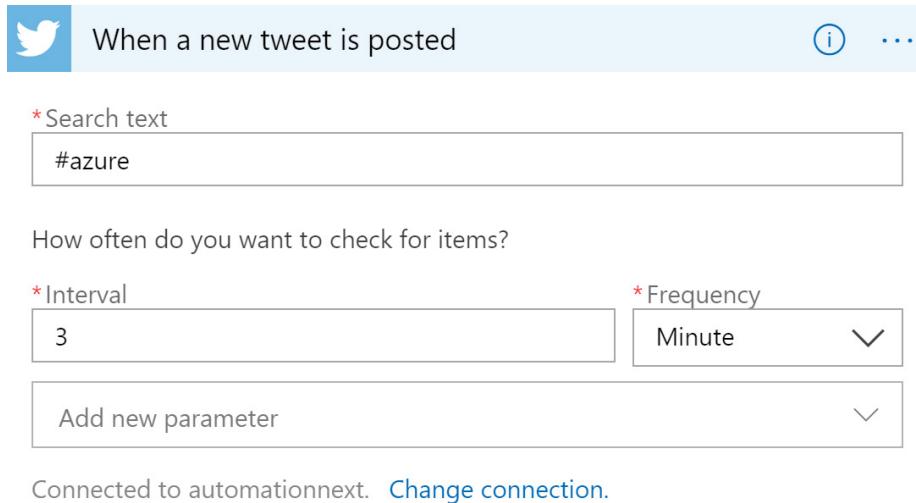


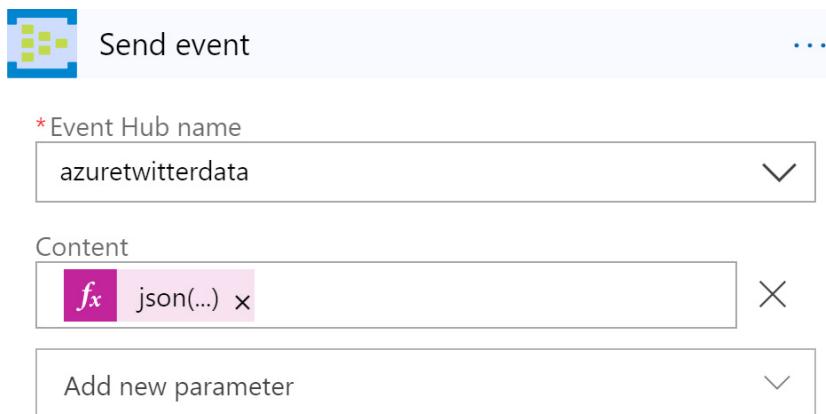
Figura 12.22: criando um aplicativo lógico

Depois que o aplicativo lógico for criado, selecione **Quando um novo tweet for postado**, faça login e configure-o como mostrado na Figura 12.23. Você precisará de uma conta válida no Twitter antes de configurar esse gatilho:



**Figura 12.23:** configurando a frequência dos tweets de entrada

Em seguida, solte uma ação de **Enviar evento** na superfície do designer. Essa ação é responsável pelo envio de tweets para o hub de eventos:



**Figura 12.24:** adicionando uma ação para enviar tweets para o hub de eventos

Selecione o nome do hub de eventos que foi criado em uma etapa anterior.

O valor especificado na caixa de texto do conteúdo é uma expressão que foi redigida dinamicamente usando funções fornecidas pelo Aplicativo Lógico e dados do Twitter. Ao clicar em **Adicionar conteúdo dinâmico**, uma caixa de diálogo é fornecida, por meio da qual é possível redigir a expressão:

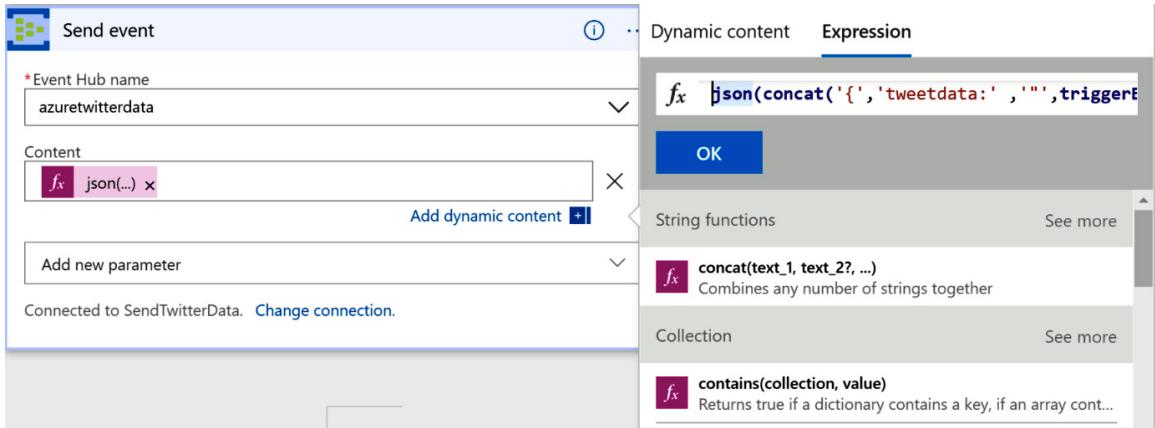


Figura 12.25: configurando a atividade de Aplicativos Lógicos usando expressões dinâmicas

O valor da expressão é o seguinte:

```
json(concat('{', 'tweetdata:', '''', triggerBody()?['TweetText'], '''', '}'))
```

Na próxima seção, provisionaremos a conta de armazenamento.

## Provisionar a conta de armazenamento

Clique em **+ Criar um recurso** e procure **Conta de armazenamento**. Selecione **Conta de armazenamento** nos resultados da pesquisa e crie uma nova conta de armazenamento. Em seguida, forneça um nome e local e selecione uma assinatura com base no grupo de recursos que foi criado anteriormente. Por fim, selecione **StorageV2** para **Tipo de Conta, Padrão** para **Performance** e **Armazenamento com redundância local (LRS)** para o campo **Replicação**.

Em seguida, criaremos um contêiner de Armazenamento de Blobs para armazenar os dados provenientes do Stream Analytics.

## Criar um contêiner de armazenamento

O Stream Analytics gerará os dados como arquivos, que serão armazenados em um contêiner de Armazenamento de Blobs. Um contêiner chamado **twitter** será criado no Armazenamento de Blobs, conforme mostrado na Figura 12.26:

The screenshot shows the Azure Storage account interface for 'twittereventdata'. On the left, a sidebar lists various options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, and Storage Explorer (preview). The 'Access control (IAM)' option is currently selected. The main area displays a table of containers. One container, 'twitter', is highlighted with a blue border. The table columns are NAME, LAST MODIFIED, PUBLIC ACCESS L..., and LEASE STATE. The 'twitter' row shows the last modified date as 1/25/2019, 8:57:50 AM, and the lease state as Available.

Figura 12.26: criando um contêiner de armazenamento

Vamos criar um novo trabalho do Stream Analytics com um ambiente de hospedagem na nuvem e definir as unidades de streaming para as configurações padrão.

## Criar trabalhos do Stream Analytics

A entrada para esse trabalho do Stream Analytics vem do hub de eventos e, por isso, precisamos configurá-la no menu **Entradas**:

The screenshot shows the Azure Stream Analytics 'Inputs' configuration page. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Locks), Job topology (Inputs selected), Functions, Query, Outputs, Configure (Scale, Locale, Event ordering, Error policy, Compatibility level), and a 'Save' button. The main area has a table with columns NAME, SOURCE TYPE, and SOURCE. There is one entry: 'Empty'. To the right, there is a detailed configuration panel for an Event Hub input:

- Name:** TwitterIncomingEvents
- Source Type:** Event Hub
- Subscription:** RiteshSubscription
- Event Hub namespace:** AzureTwitterEventData
- Event Hub name:** azurewitterdata
- Event Hub policy name:** RootManageSharedAccessKey
- Event Hub policy key:** (redacted)
- Event Hub consumer group:** (redacted)
- Event serialization format:** JSON
- Encoding:** UTF-8

Figura 12.27: criando um trabalho de entrada do Stream Analytics

A saída para o trabalho do Stream Analytics é uma conta de Armazenamento de Blobs, portanto, você precisa configurar a saída de acordo. Forneça um padrão de caminho adequado para este exercício. Por exemplo, **{datetime:ss}** é o padrão de caminho que estamos usando para este exercício:

The screenshot shows the 'Outputs' blade for the 'StreamAzureTwitterJob'. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Locks), Job topology (Inputs, Functions, Query), and Outputs. The 'Outputs' link is highlighted. The main area shows a table with one row for 'TwitterData' using 'Blob storage' as the sink. To the right, there's a detailed configuration panel for 'TwitterData'. It includes fields for Output alias (set to 'TwitterData'), Blob storage settings (Subscription: RiteshSubscription, Storage account: twittereventdata, Storage account key: masked, Container: twitter, Path pattern: '{datetimestss}'), Date format (YYYY/MM/DD), and Time format (HH). A 'Save' button is at the bottom.

Figura 12.28: criando uma conta de Armazenamento de Blobs como saída

A consulta é bem simples; você está apenas copiando os dados da entrada para a saída:

The screenshot shows the 'Query' blade for the 'StreamAzureTwitterJob'. The sidebar has the same structure as the previous screenshot. The main area shows the query editor with the following code:

```

1  SELECT
2    *
3  INTO
4    [TwitterData]
5  FROM
6    [TwitterIncomingEvents]
  
```

The 'Inputs' section shows 'TwitterIncomingEvents' and the 'Outputs' section shows 'TwitterData'.

Figura 12.29: consulta para copiar feeds do Twitter

Embora este exemplo envolva apenas a cópia de dados, pode haver consultas mais complexas para realizar a transformação antes de carregar dados em um destino.

Isso conclui todas as etapas para a aplicação. Agora, você deve conseguir executá-la.

## Executar a aplicação

O aplicativo lógico deve estar habilitado, e o Stream Analytics deve estar em execução. Agora, execute o aplicativo lógico. Isso criará um trabalho para executar todas as atividades nele, conforme mostrado na Figura 12.30:

The screenshot shows the Azure Logic App interface for the 'GetAzureTwitterData' application. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Development Tools (Logic app designer, Logic app code view, Versions, API connections, Quick start guides, Release notes), and Settings (Workflow settings, Access keys, Identity, Properties, Locks). The main content area displays the following details:

- Resource group (change) :** TwitterAnalysis
- Location :** West Europe
- Subscription (change) :** RiteshSubscription
- Subscription ID :** 9755ffce-e94b-4332-9be8-1ade15e78909
- Status :** Enabled
- Runs last 24 hours :** 217 successful, 1275 failed
- Integration Account :** ---

**Summary**

Trigger	Actions
<b>TWITTER</b> When a new tweet is posted	<b>COUNT</b> 1 action <a href="#">View in Logic Apps designer</a>
<b>FREQUENCY</b> Runs every 3 minutes	
<b>EVALUATION</b> Evaluated 2357 times, fired 1494 times in the last 24 hours <a href="#">See trigger history</a>	

**Runs history**

STATUS	START TIME	IDENTIFIER	DURATION
Succeeded	1/26/2019, 10:16 AM	08586530910864161272447961552CU24	299 Milliseconds
Succeeded	1/26/2019, 10:15 AM	08586530911478685688296257377CU29	447 Milliseconds

Figura 12.30: visão geral da aplicação GetAzureTwitterData

O contêiner da **Conta de armazenamento** deve obter dados, conforme mostrado na Figura 12.31:

Property	Value
URL	<a href="https://twittereventdata.blob.core.windows.net/">https://twittereventdata.blob.core.windows.net/</a>
LAST MODIFIED	1/25/2019, 9:06:25 AM
CREATION TIME	1/25/2019, 9:06:24 AM
TYPE	Block blob
SIZE	10.8 KiB
SERVER ENCRYPTED	true
ETAG	0x8D682CE4AF8642B
CONTENT-MD5	-
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-
COPY STATUS	-
COPY COMPLETION TIME	-

Figura 12.31: verificando os dados do contêiner da conta de armazenamento

Como exercício, você pode estender essa solução de exemplo e avaliar o sentimento dos tweets a cada três minutos. O fluxo de trabalho de Aplicativos Lógicos para esse exercício será o seguinte:

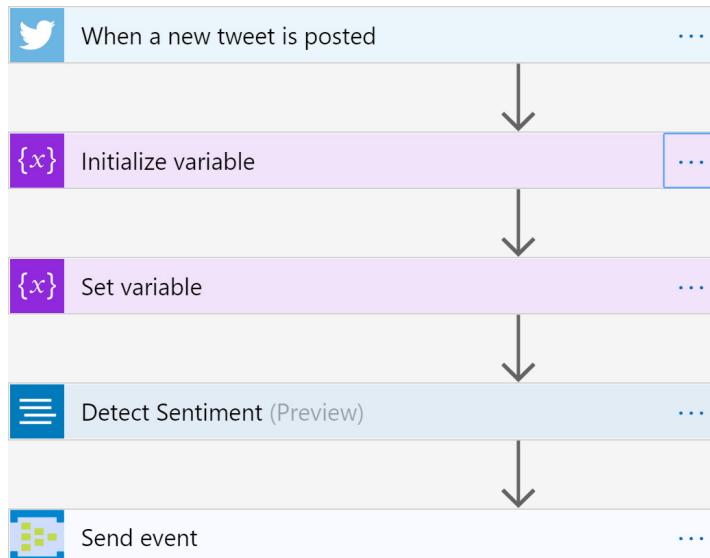


Figura 12.32: fluxograma para analisar o sentimento do tweet

Para detectar o sentimento, você precisará usar a API de Análise de Texto, que deve ser configurada antes de ser usada em Aplicativos Lógicos.

## Resumo

Este capítulo abordou tópicos relacionados ao streaming e o armazenamento de eventos. Os eventos tornaram-se uma consideração importante na arquitetura geral de soluções. Abordamos recursos importantes, como os Hubs de Eventos e o Stream Analytics, e conceitos fundamentais, como grupos de consumidores e taxas de transferência, além de criar uma solução completa usando-os juntamente com Aplicativos Lógicos. Você aprendeu que os eventos são gerados de várias fontes e, para obter insights em tempo real sobre atividades e seus eventos relacionados, serviços como os Hubs de Eventos e o Stream Analytics desempenham uma função significativa. No próximo capítulo, aprenderemos a integrar o Azure DevOps e o Jenkins e implementar algumas das práticas recomendadas da indústria ao desenvolver soluções.





# 13

## Integrar o DevOps do Azure

No capítulo anterior, você aprendeu sobre eventos de big data e seu relacionamento com os Hubs de Eventos do Azure e os serviços do Stream Analytics. O desenvolvimento de software é uma tarefa complexa composta de vários processos e ferramentas, envolvendo pessoas de diferentes departamentos. Todos precisam se unir e trabalhar de maneira coesa. Com tantas variáveis, os riscos são altos quando você entrega aos clientes finais. Uma pequena omissão ou configuração incorreta pode arruinar a aplicação. Este capítulo é sobre a adoção e a implementação de práticas que reduzem esse risco consideravelmente e garantem que um software de alta qualidade possa ser entregue ao cliente constantemente.

Antes de analisar os detalhes do DevOps, veja a seguir uma lista dos problemas enfrentados pelas empresas de software que são resolvidos pelo DevOps:

- Organizações rígidas que não aceitam mudanças
- Processos demorados
- Equipes isoladas trabalhando em silos
- Design monolítico e implantações repentinhas
- Execução manual
- Falta de inovação

Neste capítulo, abordaremos os seguintes tópicos:

- DevOps
- Práticas de DevOps
- Azure DevOps
- Preparação de DevOps
- DevOps para soluções de PaaS
- DevOps para soluções baseadas em máquinas virtuais (IaaS)
- DevOps para soluções baseadas em contêineres (IaaS)
- Azure DevOps e Jenkins
- Automação do Azure
- Ferramentas do Azure para DevOps

## DevOps

No momento, não há um consenso na indústria quanto à definição de DevOps. As organizações têm formulado sua própria definição de DevOps e tentado implementá-la. Elas têm sua própria perspectiva e acreditam que terão implementado o DevOps quando implementarem o gerenciamento de automação e configuração e usarem processos ágeis.

Com base na minha experiência de trabalho em projetos de DevOps na indústria, defino o DevOps da seguinte forma: DevOps consiste no mecanismo de entrega de sistemas de software. Ele aproxima as pessoas, fazendo com que colaborem e se comuniquem, trabalhando juntas para alcançar um objetivo e uma visão comuns. Ele incentiva que as pessoas assumam responsabilidade e propriedade em conjunto. Ele permite a implementação de processos que promovem a colaboração e uma mentalidade de serviço. Ele habilita os mecanismos de entrega que levam agilidade e flexibilidade à organização. Ao contrário da crença popular, o DevOps não é sobre ferramentas, tecnologia, automação. Esses são facilitadores que ajudam na colaboração, na implementação de processos ágeis e na entrega melhor e mais rápida ao cliente.

Há várias definições disponíveis na Internet para o DevOps e elas não estão erradas. O DevOps não fornece uma estrutura nem uma metodologia. É um conjunto de princípios e práticas que, quando utilizado dentro de uma organização, engajamento ou projeto, atinge o objetivo e a visão do DevOps e da organização. Esses princípios e práticas não exigem processos, ferramentas, tecnologias ou ambientes específicos. O DevOps fornece as orientações que podem ser implementadas por meio de qualquer ferramenta, tecnologia ou processo, embora parte da tecnologia e dos processos possa ser mais aplicável do que outras para alcançar a visão de princípios e práticas de DevOps.

Embora as práticas de DevOps possam ser implementadas em qualquer organização que forneça serviços e produtos para clientes, neste livro, examinaremos o DevOps da perspectiva do desenvolvimento de software e do departamento de operações de qualquer organização.

Então, o que é o DevOps? O DevOps é definido como um conjunto de princípios e práticas que reúne todas as equipes, incluindo desenvolvedores e operações desde o início do projeto para obter uma entrega completa mais rápida e eficiente desse sistema ao cliente final, de forma consistente e previsível, reduzindo o tempo de chegada ao mercado e, consequentemente, obtendo uma vantagem competitiva.

A definição anterior do DevOps não indica nem se refere a processos, ferramentas ou tecnologia específicos. Ela não prescreve qualquer metodologia ou ambiente.

O objetivo da implementação de princípios e práticas de DevOps em qualquer organização é garantir que as demandas dos stakeholders (incluindo os clientes) e as expectativas sejam atendidas com eficiência e eficácia.

As demandas e expectativas do cliente são atendidas quando o seguinte acontece:

- O cliente obtém os recursos desejados
- O cliente obtém os recursos quando desejados
- O cliente recebe atualizações mais rápidas sobre os recursos
- A qualidade da entrega é alta

Quando uma organização consegue atender a essas expectativas, os clientes ficam satisfeitos e permanecem fiéis a ela. Por sua vez, isso aumenta a competitividade de mercado da organização, que resulta em um maior valor de marca e de mercado.

Isso tem um impacto direto sobre a receita e o lucro líquido da organização. Ela pode investir mais em inovação e comentários dos clientes, provocando mudanças contínuas em seus sistemas e serviços para permanecer relevante.

A implementação de princípios e práticas de DevOps em qualquer organização é orientada pelo ecossistema ao seu redor. Esse ecossistema é composto pela indústria e pelos domínios aos quais a organização pertence.

O DevOps se baseia em um conjunto de princípios e práticas. Vamos ver os detalhes desses princípios e práticas mais adiante neste capítulo. Os princípios fundamentais do DevOps são:

- **Agilidade:** ser ágil aumenta a flexibilidade geral para mudanças e garante que a adaptabilidade aumente para cada ambiente em constante mudança e que seja produtivo. Os processos ágeis têm uma duração de trabalho mais curta, e é fácil encontrar problemas no início do ciclo de vida do desenvolvimento, em vez de muito mais tarde, reduzindo assim a dívida técnica.
- **Automação:** a adoção de ferramentas e da automação aumenta a eficiência geral e a previsibilidade do processo e do produto final. Ela ajuda a fazer as coisas de forma mais rápida, mais fácil e menos dispendiosa.
- **Colaboração:** refere-se a um repositório comum, à rotação de responsabilidades de trabalho, ao compartilhamento de dados e informações e a outros aspectos que melhoraram a produtividade de cada membro da equipe, apoiando assim a entrega eficaz do produto em geral.
- **Comentários:** refere-se a loops de comentários rápidos e antecipados entre várias equipes sobre coisas que funcionam e não funcionam. Eles ajudam as equipes a priorizar problemas e corrigi-los em versões subsequentes.

As principais práticas de DevOps são:

- **Integração contínua:** refere-se ao processo de validação e verificação da qualidade e da exatidão do código enviado no repositório pelos desenvolvedores. Ela pode ser agendada, manual ou contínua. "Contínua" significa que o processo verificará vários atributos de qualidade cada vez que um desenvolvedor enviar o código, enquanto "agendada" significa que, em um período agendado, as verificações serão conduzidas. "Manual" refere-se à execução manual de um administrador ou desenvolvedor.
- **Gerenciamento de configuração:** é uma faceta importante do DevOps e fornece orientações para configurar a infraestrutura e as aplicações, extraíndo configurações de servidores de gerenciamento de configuração ou enviando essas configurações de acordo com um agendamento. O gerenciamento de configuração deverá trazer o ambiente novamente para o estado desejado esperado toda vez que for executado.
- **Entrega contínua:** refere-se ao estado de preparação de uma aplicação para poder ser implantada em qualquer ambiente existente, bem como um novo. Ela geralmente é executada por meio de uma definição de versão em ambientes mais baixos, como desenvolvimento e teste.
- **Implantação contínua:** refere-se à capacidade de implantar o ambiente e a aplicação em produção automaticamente. Ela geralmente é executada por meio de uma definição de versão no ambiente de produção.
- **Aprendizado contínuo:** refere-se ao processo de compreensão dos problemas enfrentados pelas operações e pelos clientes e garantir que eles sejam comunicados às equipes de desenvolvimento e testes de modo que possam corrigir esses problemas em versões subsequentes para melhorar a integridade geral e a usabilidade da aplicação.

## A essência do DevOps

O DevOps não é um paradigma novo. No entanto, tem ganhado muita popularidade e tracção. Sua adoção está no auge, e cada vez mais empresas estão iniciando essa jornada. Eu me referi intencionalmente ao DevOps como uma jornada porque há diferentes níveis de maturidade dentro dele. Enquanto a implementação bem-sucedida de implantação e entrega contínuas é considerada como o nível mais alto de maturidade nessa jornada, a adoção do desenvolvimento de software Agile e controle de código-fonte é considerada como a primeira etapa na jornada de DevOps.

Um dos primeiros tópicos abordados pelo DevOps é a remoção de barreiras entre as equipes de desenvolvimento e de operações. Isso resulta na estreita colaboração entre várias equipes. Ela muda a mentalidade de que o desenvolvedor é responsável apenas por escrever o código e passá-lo adiante às operações para que seja implantado após passar por testes. Ela também muda a mentalidade de que as operações não desempenham nenhum papel nas atividades de desenvolvimento. As operações devem influenciar o planejamento do produto e estar cientes dos recursos que serão lançados em breve. Elas também devem continuamente fornecer comentários aos desenvolvedores sobre os problemas operacionais, de modo que eles possam ser corrigidos em versões posteriores. Elas devem influenciar o design do sistema para melhorar o funcionamento operacional desse sistema. Da mesma forma, os desenvolvedores devem ajudar a equipe de operações a implantar o sistema e resolver os incidentes quando surgirem.

A definição de DevOps fala sobre uma entrega de ponta a ponta dos sistemas às stakeholders de forma mais rápida e eficiente. Ela não fala sobre o nível de rapidez ou eficiência que a entrega deve ter. Seu nível de rapidez depende do domínio da organização, da indústria, da segmentação de clientes e necessidades. Para algumas organizações, as versões trimestrais são suficientes, enquanto para outras elas poderiam ser semanais. Ambos são válidos de um ponto de vista de DevOps, e essas organizações podem implantar processos e tecnologias relevantes para atingir seus prazos de lançamento de destino. O DevOps não requer nenhum período específico para a **integração contínua/implantação contínua (CI/CD)**. As organizações devem identificar a melhor implementação de princípios e práticas de DevOps com base no seu projeto como um todo, no engajamento e na visão organizacional.

A definição também fala sobre entrega de ponta a ponta. Isso significa que tudo, desde o planejamento e a entrega do sistema aos serviços até as operações, devem fazer parte da adoção de DevOps. Os processos permitem maior flexibilidade, modularidade e agilidade no ciclo de vida de desenvolvimento da aplicação. Embora as organizações sejam livres para usar o processo mais adequado, como Waterfall, Agile, Scrum e muito mais, elas normalmente costumam preferir processos ágeis com entrega baseada em iterações. Isso permite uma entrega mais rápida em unidades menores que são muito mais testáveis e gerenciáveis em comparação a uma grande entrega.

O DevOps fala repetidamente sobre clientes finais de forma consistente e previsível. Isso significa que as organizações devem continuamente entregar aos clientes recursos novos e atualizados usando a automação. Não podemos alcançar consistência e previsibilidade sem o uso da automação. O trabalho manual não deve existir para garantir um alto nível de consistência e previsibilidade. A automação também deve ser de ponta a ponta para evitar falhas. Isso também indica que o design do sistema deve ser modular, permitindo entregas mais rápidas em sistemas confiáveis, disponíveis e escaláveis. Os testes desempenham um papel importante em entregas consistentes e previsíveis.

O resultado final da implementação dessas práticas e princípios mencionados anteriormente é que a organização é capaz de atender às expectativas e demandas dos clientes. Ela é capaz de crescer mais rápido do que a concorrência e aumentar ainda mais a qualidade e a capacidade de seus produtos e serviços por meio de inovação e melhoria contínua.

Agora que você entendeu a ideia por trás do DevOps, é hora de investigar as práticas essenciais de DevOps.

## Práticas de DevOps

O DevOps consiste em várias práticas, cada uma fornecendo funcionalidades distintas ao processo como um todo. O diagrama a seguir mostra a relação entre elas. O gerenciamento de configuração, a integração contínua e a implantação contínua formam as práticas fundamentais que habilitam o DevOps. Quando fornecemos serviços de software que combinam esses três serviços, alcançamos a entrega contínua. A entrega contínua é a capacidade e o nível de maturidade de uma organização dependente da maturidade de gerenciamento de configuração, integração contínua e implantação contínua. Comentários contínuos em todos os estágios constituem o loop de comentários que ajuda a fornecer serviços melhores aos clientes. Ele é executado em todas as práticas de DevOps. Vamos nos aprofundar em cada um desses recursos e práticas de DevOps:

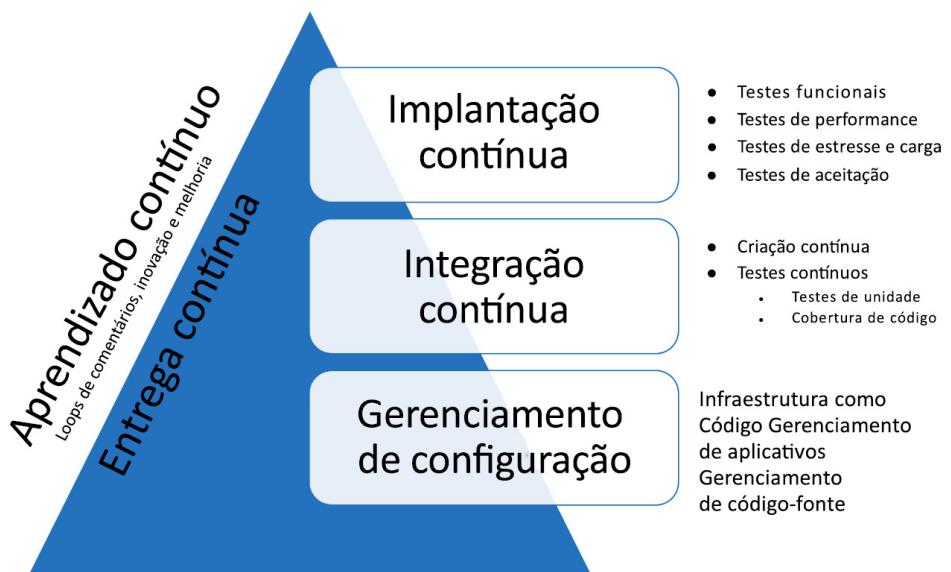


Figura 13.1: práticas de DevOps

## Gerenciamento de configuração

Serviços e aplicações de negócios precisam de um ambiente em que possam ser implantados. Normalmente, o ambiente é uma infraestrutura composta de vários servidores, computadores, rede, armazenamento, contêineres e muitos outros serviços funcionando em conjunto, de modo que as aplicações de negócios possam ser implantadas sobre eles. Aplicações de negócios são decompostas em vários serviços executados em vários servidores, na infraestrutura local ou na nuvem, e cada serviço tem sua própria configuração, juntamente com requisitos relacionados à configuração da infraestrutura. Em suma, a aplicação e a infraestrutura são necessárias para fornecer sistemas aos clientes e cada um tem sua própria configuração. Se houver desvio na configuração, a aplicação pode não funcionar conforme o esperado, resultando em tempo de inatividade e falhas. Além disso, como os processos de **Gerenciamento do ciclo de vida da aplicação (ALM)** determinam o uso de vários estágios e ambientes, um aplicação será implantada em vários ambientes com diferentes configurações. A aplicação será implantada no ambiente de desenvolvimento para que os desenvolvedores vejam o resultado do seu trabalho. Ela será implantada em vários ambientes de teste com diferentes configurações para testes funcionais, de carga e estresse, de performance, de integração e muito mais. Também será implantada no ambiente de pré-produção para realizar testes de aceitação do usuário e, por fim, no ambiente de produção. É importante que uma aplicação possa ser implantada em vários ambientes sem alterações manuais na sua configuração.

O gerenciamento de configuração fornece um conjunto de processos e ferramentas para garantir que cada ambiente e aplicação tenha sua própria configuração. Ele acompanha itens de configuração, e qualquer coisa que mude de um ambiente para outro deve ser tratada como um item de configuração. Ele também define as relações entre os itens de configuração e como as alterações em um desses itens afetará o outro.

### Uso do gerenciamento de configuração

O gerenciamento de configuração ajuda nos seguintes pontos:

- **Infraestrutura como Código:** quando o processo de provisionamento da infraestrutura e sua configuração são representados por meio de código, e o mesmo código passa pelo processo de ciclo de vida da aplicação, isso é conhecido como **Infraestrutura como Código (IaC)**. O IaC ajuda a automatizar o provisionamento e a configuração de infraestrutura. Ela também representa toda a infraestrutura em código que pode ser armazenado em um repositório e submetido ao controle de versão. Isso permite que os usuários empreguem as configurações do ambiente anterior quando necessário. Também permite o provisionamento de um ambiente várias vezes de forma consistente e previsível. Todos os ambientes provisionados dessa forma são consistentes e iguais em todos os estágios de ALM. Há muitas ferramentas que ajudam a alcançar a IaC, incluindo modelos do ARM, Ansible e Terraform.

- **Implantar e configurar a aplicação:** a implantação de uma aplicação e sua configuração são a etapa seguinte ao provisionamento da infraestrutura. A implantação de um pacote **webdeploy** em um servidor, a implantação de um esquema e dados de um SQL Server (**bacpac**) em outro servidor, e a alteração da cadeia de conexão do SQL no servidor Web para representar o SQL Server apropriado são exemplos de implantação e configuração da aplicação. O gerenciamento de configuração armazena valores da configuração da aplicação para cada ambiente no qual é implantada.

A configuração aplicada também deve ser monitorada. A configuração esperada e desejada deve ser mantida consistentemente. Qualquer desvio dessa configuração esperada e desejada tornará a aplicação indisponível. O gerenciamento de configuração também é capaz de encontrar o desvio e reconfigurar a aplicação e o ambiente para o seu estado desejado.

Com o gerenciamento de configuração automatizado em vigor, ninguém na equipe precisa implantar e configurar os ambientes e aplicações em produção. A equipe de operações não depende da equipe de desenvolvimento ou de uma longa documentação de implantação.

Outro aspecto do gerenciamento de configuração é o controle do código-fonte. Serviços e aplicações de negócios englobam o código e outros artefatos. Vários membros da equipe trabalham nos mesmos arquivos. O código-fonte deve estar sempre atualizado e ser acessado somente por membros autenticados da equipe. O código e outros artefatos isolados são itens de configuração. O controle do código-fonte ajuda na colaboração e na comunicação dentro da equipe, pois todos estão cientes do que todos estão fazendo, e os conflitos são resolvidos em um estágio inicial.

O gerenciamento de configuração pode ser amplamente dividido em duas categorias:

- Dentro da máquina virtual
- Fora da máquina virtual

## Ferramentas de gerenciamento de configuração

As ferramentas disponíveis para o gerenciamento de configuração dentro da máquina virtual são abordadas a seguir.

### Desired State Configuration

**Desired State Configuration (DSC)** é uma nova plataforma de gerenciamento de configuração da Microsoft criada como uma extensão do PowerShell. O DSC foi originalmente lançado como parte do **Windows Management Framework (WMF) 4.0**. Ele está disponível como parte do WMF 4.0 e 5.0 para todos os sistemas operacionais Windows Server anteriores ao Windows 2008 R2. O WMF 5.1 está disponível e pronto para uso no Windows Server 2016/2019 e no Windows 10.

## Chef, Puppet e Ansible

Além do DSC, há uma série de ferramentas de gerenciamento de configuração, como Chef, Puppet e Ansible com suporte do Azure. Detalhes sobre essas ferramentas não são abordados neste livro. Leia mais sobre elas aqui: <https://docs.microsoft.com/azure/virtual-machines/windows/infrastructure-automation>.

As ferramentas disponíveis para gerenciamento de configuração fora de uma máquina virtual são:

## Modelos do ARM

Os modelos do ARM são os principais meios de provisionamento de recursos no ARM. Eles fornecem um modelo declarativo por meio do qual recursos e suas configurações, scripts e extensões são especificados. Os modelos do ARM se baseiam no formato **JavaScript Object Notation (JSON)**. Eles usam as convenções e a sintaxe JSON para declarar e configurar recursos. Os arquivos JSON são baseados em texto, fáceis de usar e de ler. Eles podem ser armazenados em um repositório de código-fonte e submetidos ao controle de versão. Eles também servem para representar a infraestrutura como código que pode ser usada para provisionar recursos em grupos de recursos do Azure de forma constante, previsível, consistente e uniforme.

Os modelos fornecem a flexibilidade de serem genéricos e modulares no seu design e na sua implementação. Os modelos fornecem a capacidade de aceitar parâmetros de usuários, declarar variáveis internas, ajudar na definição de dependências entre recursos, vincular recursos dentro de grupos de recursos iguais ou diferentes, além de executar outros modelos. Eles também fornecem expressões e funções do tipo de linguagem de script que os tornam dinâmicos e personalizáveis no tempo de execução. Há dois capítulos dedicados a modelos ARM neste livro: *Capítulo 15, Implantações entre assinaturas usando modelos do ARM* e *Capítulo 16, Design modular e implementação de modelos do ARM*.

Agora, é hora de se concentrar no próximo importante princípio de DevOps: a integração contínua.

## Integração contínua

Vários desenvolvedores escrevem código que eventualmente é armazenado em um repositório comum. O código normalmente é verificado ou enviado ao repositório quando os desenvolvedores terminam de desenvolver seus recursos. Isso pode ocorrer em um dia ou pode levar dias ou semanas. Alguns dos desenvolvedores talvez estejam trabalhando no mesmo recurso e também seguindo as mesmas práticas de envio/verificação de código em dias ou semanas. Isso pode criar problemas para a qualidade do código. Um dos princípios do DevOps é falhar rápido. Os desenvolvedores devem verificar/enviar o código ao repositório frequentemente e compilá-lo para verificar se não criou bugs e se ele é compatível com o código escrito por seus colegas. Se um desenvolvedor não seguir essa prática, o código em sua máquina ficará muito grande, e será difícil integrá-lo ao código de outro colega. Além disso, se a compilação falhar, será difícil e demorado corrigir os problemas decorrentes disso.

## Integração de código

A integração contínua resolve esses tipos de desafios. Ela ajuda na compilação e na validação do código enviado/verificado por um desenvolvedor por meio de uma série de etapas de validação. A integração contínua cria um fluxo de processo que consiste em várias etapas. A integração contínua é composta por compilações automatizadas contínuas e testes automatizados contínuos. Normalmente, a primeira etapa é o build do código. Após o build bem-sucedido, cada etapa é responsável por validar o código de uma perspectiva específica. Por exemplo, os testes de unidade podem ser executados no código compilado e a cobertura de código pode ser executada para verificar quais caminhos de código são executados pelos testes de unidade. Eles podem revelar se testes de unidade abrangentes estão escritos ou se há espaço para adicionar mais testes de unidade. O resultado final da integração contínua são pacotes de implantação que podem ser usados pela implantação contínua para implantá-los em vários ambientes.

## Envio de código frequente

Os desenvolvedores são incentivados a verificar seu código várias vezes por dia, em vez de fazer isso após dias ou semanas. A integração contínua inicia a execução do pipeline inteiro logo após a verificação ou o envio do código. Se a compilação for bem-sucedida, os testes de código e outras atividades que fazem parte do pipeline serão executados sem erros, o código será implantado em um ambiente de teste, e testes de integração serão executados nele.

## Maior produtividade

A integração contínua aumenta a produtividade dos desenvolvedores. Eles não precisam compilar seu código manualmente, executar vários tipos de testes, um após o outro, e criar pacotes a partir deles. Ela também reduz o risco de introdução de bugs no código, o qual não fica obsoleto. Ela fornece comentários precoces aos desenvolvedores sobre a qualidade do seu código. Em geral, a qualidade dos produtos finais é alta, e eles são entregues com mais rapidez por meio da adoção de práticas de integração contínua. Um pipeline de integração contínua de exemplo é mostrado aqui:

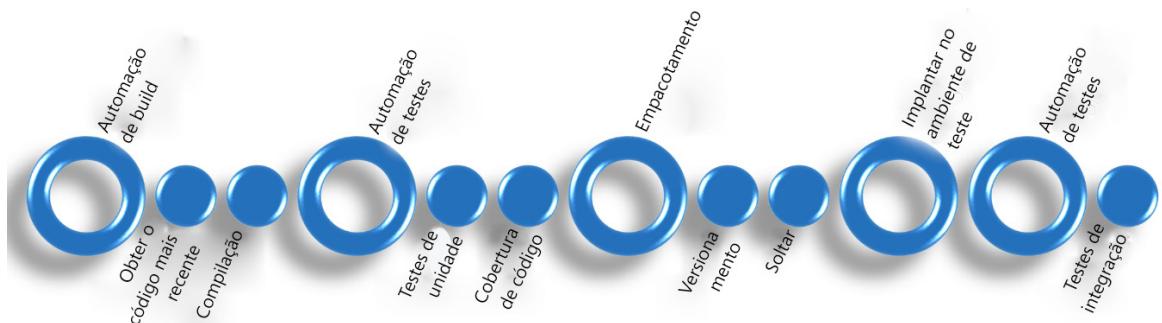


Figura 13.2: pipeline de integração contínua

## Automação de build

A automação de build consiste em várias tarefas executadas em sequência. Geralmente, a primeira tarefa é responsável pela obtenção do código-fonte mais recente do repositório. O código-fonte pode incluir vários projetos e arquivos. Eles são compilados para gerar artefatos, como executáveis, bibliotecas de vínculo dinâmico e assemblies. A automação de build bem-sucedida revela que não há erros de tempo de compilação no código.

Pode haver mais etapas na automação de build dependendo da natureza e do tipo de projeto.

## Automação de testes

A automação de testes consiste em tarefas que são responsáveis por validar diferentes aspectos do código. Essas tarefas têm como objetivo testar o código de uma perspectiva diferente e são executadas em sequência. Geralmente, a primeira etapa é a execução de uma série de testes de unidade no código. Os testes de unidade se referem ao processo de testes da menor denominação de um recurso, validando seu comportamento isolado de outros recursos. Eles podem ser manuais ou automatizados. No entanto, a preferência é por testes de unidade automatizados.

A cobertura de código é outro tipo de teste automatizado que pode ser realizado no código para descobrir qual parte dele é executada durante os testes de unidade. Ela geralmente é representada como uma porcentagem e se refere à parte de código testável por meio de testes de unidade. Se a cobertura de código não for próxima a 100%, é porque o desenvolvedor não criou testes de unidade para esse comportamento ou o código não coberto não é necessário.

A execução bem-sucedida da automação de testes que não resultar em nenhuma falha de código significativa deverá iniciar a execução das tarefas de empacotamento. Pode haver mais etapas para a automação de testes dependendo da natureza e do tipo de projeto.

## Empacotamento

O empacotamento se refere ao processo de geração de artefatos implantáveis, como pacotes **MSI**, **NuGet** e **webdeploy**, pacotes de bancos de dados, seu versionamento e armazenamento em um local onde possam ser consumidos por outros pipelines e processos.

Depois que o processo de integração contínua for concluído, o processo de implantação contínua será iniciado, e esse será o foco da próxima seção.

## Implantação contínua

Quando o processo chegar na implantação contínua, a integração contínua terá garantido que tenhamos bits totalmente funcionais de uma aplicação que agora poderá passar por diferentes atividades de implantação contínua. A implantação contínua se refere à capacidade de implantação de serviços e aplicações de negócios em ambientes de pré-produção e produção por meio da automação. Por exemplo, a implantação contínua pode provisionar e configurar o ambiente de pré-produção, implantar aplicações nele e configurá-los. Após a realização de várias validações, como testes funcionais e de performance no ambiente de pré-produção, o ambiente de produção é provisionado, configurado e a aplicação é implantada em ambientes de produção por meio da automação. Não há etapas manuais no processo de implantação. Todas as tarefas de implantação são automatizadas. A implantação contínua pode provisionar o ambiente e implantar a aplicação do zero, enquanto poderá simplesmente implantar as alterações delta no ambiente existente se o ambiente já existir.

Todos os ambientes são provisionados por meio de automação usando a IaC. Isso garante que todos os ambientes, sejam eles de desenvolvimento, teste, pré-produção ou produção, sejam iguais. Da mesma forma, a aplicação é implantada por meio da automação, garantindo que também seja implantado uniformemente em todos os ambientes. A configuração nesses ambientes pode ser diferente para a aplicação.

A implantação contínua geralmente está assimilada à integração contínua. Quando a integração contínua conclui seu trabalho gerando os pacotes implantáveis finais, a implantação contínua entra em ação e inicia seu próprio pipeline. Esse pipeline é chamado de pipeline de lançamento. O pipeline de lançamento consiste em vários ambientes, e cada um deles consiste em tarefas responsáveis por provisionar e configurar o ambiente, implantar e configurar aplicações, executar validação operacional em ambientes e testar a aplicação em vários ambientes.

Empregar a implantação contínua fornece benefícios imensos. Há um alto nível de confiança no processo de implantação como um todo que ajuda em lançamentos mais rápidos e livres de riscos na produção. As chances de algo dar errado são drasticamente reduzidas. A equipe terá níveis de estresse mais baixos, e a reversão para o ambiente de trabalho anterior será possível se houver problemas na versão atual:

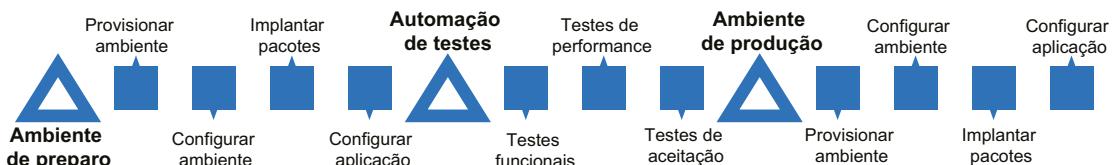


Figura 13.3: pipeline de implantação contínua

Embora cada sistema exija sua própria configuração de pipeline de lançamento, um exemplo da implantação contínua é mostrado no diagrama anterior. É importante observar que geralmente o provisionamento e a configuração de vários ambientes fazem parte do pipeline de lançamento e aprovações devem ser obtidas antes de passar para o próximo ambiente. O processo de aprovação pode ser manual ou automatizado dependendo da maturidade da organização.

Em seguida, investigaremos aspectos relacionados ao ambiente de teste.

### **Implantação do ambiente de teste**

O pipeline de lançamento é iniciado quando o depósito da integração contínua fica disponível, e a primeira etapa é obter todos os artefatos do depósito. Depois disso, o pipeline de lançamento pode criar um ambiente de teste bare-metal completamente novo ou reutilizar um já existente. Novamente, isso depende do tipo de projeto e da natureza dos testes planejados nesse ambiente. O ambiente é provisionado e configurado. Os artefatos da aplicação são implantados e configurados.

### **Automação de testes**

Após a implantação de uma aplicação, uma série de testes pode ser realizada no ambiente. Um dos testes realizados aqui é um teste funcional. Os testes funcionais são destinados principalmente a validar a plenitude do recurso e a funcionalidade da aplicação. Esses testes são criados a partir de requisitos do cliente. Outro conjunto de testes que pode ser realizado está relacionado a escalabilidade e disponibilidade da aplicação. Isso normalmente inclui testes de carga, de estresse e de performance. Também deve incluir uma validação operacional do ambiente de infraestrutura.

### **Implantação do ambiente de preparo**

Ela é muito semelhante à implantação do ambiente de teste, sendo que a única diferença é que os valores de configuração para o ambiente e a aplicação são diferentes.

### **Testes de aceitação**

Os testes de aceitação geralmente são conduzidos pelos stakeholders da aplicação e podem ser manuais ou automatizados. Essa etapa é uma validação do ponto de vista do cliente sobre a exatidão e a integralidade da funcionalidade da aplicação.

### **Implantação na produção**

Depois que o cliente dá sua aprovação, as mesmas etapas de implantação dos ambientes de teste e preparo são executadas, sendo que a única diferença é que os valores de configuração para o ambiente e a aplicação são específicos para o ambiente de produção. Uma validação é realizada após a implantação para garantir que a aplicação esteja funcionando conforme o esperado.

A entrega contínua é um importante princípio de DevOps e é semelhante à implantação contínua. No entanto, há algumas diferenças. Na próxima seção, analisaremos a entrega contínua.

## Entrega contínua

A entrega contínua e a implantação contínua podem parecer semelhantes para muitos leitores. No entanto, não são a mesma coisa. Enquanto a implantação contínua envolve a implantação em vários ambientes e, por fim, no ambiente de produção por meio da automação, as práticas de entrega contínua consistem na capacidade de gerar pacotes de aplicações de forma que sejam prontamente implantáveis em qualquer ambiente. Para gerar artefatos prontamente implantáveis, a integração contínua deve ser usada para gerar os artefatos da aplicação e um ambiente novo ou existente deve ser usado para implantar esses artefatos e conduzir testes funcionais, testes de performance e testes de aceitação do usuário por meio da automação. Depois que essas atividades forem executadas com êxito e sem erros, o pacote da aplicação será considerado prontamente implantável. A entrega contínua inclui a integração e implantação contínua em um ambiente para validações finais. Ele ajuda a obter feedback mais rapidamente de ambas as operações e o usuário final. Esses comentários podem ser usados para implementar iterações subsequentes.

Na próxima seção, analisaremos o aprendizado contínuo.

## Aprendizado contínuo

Com todas as práticas de DevOps mencionadas anteriormente, é possível criar ótimas aplicações de negócios e implantá-las automaticamente no ambiente de produção. No entanto, os benefícios de DevOps não serão duradouros se os princípios de comentários e melhoria contínuos não estiverem em vigor. É muito importante que comentários em tempo real sobre o comportamento da aplicação, provenientes dos usuários finais e da equipe de operações, sejam repassados à equipe de desenvolvimento.

Os comentários devem ser repassados às equipes, fornecendo informações relevantes sobre o que está dando certo e, mais importante, o que não está dando certo.

A arquitetura e o design de uma aplicação devem ser criados com o monitoramento, a auditoria e a telemetria em mente. A equipe de operações deve coletar as informações de telemetria do ambiente de produção, capturando os bugs e problemas, e repassá-los à equipe de desenvolvimento para que possam ser corrigidos em versões subsequentes.

O aprendizado contínuo ajuda a tornar a aplicação robusta e resistente a falha. Ele ajuda a garantir que a aplicação atenda aos requisitos dos consumidores. A Figura 13.4 mostra o loop de comentários que deve ser implementado entre diferentes equipes:

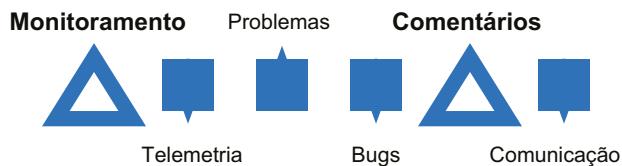


Figura 13.4: loop de comentários

Depois de explicar as práticas importantes relacionadas ao DevOps, agora é hora de abordar as ferramentas e os serviços as viabilizam.

## Azure DevOps

Vamos ver outro serviço online de alta qualidade, o Visual Studio Team Services (VSTS), que permite a integração, a implantação e a entrega contínuas facilmente: Azure DevOps. Na verdade, seria mais apropriado chamá-lo de um conjunto de serviços disponíveis sob um único nome. O Azure DevOps é uma PaaS fornecida pela Microsoft e hospedada na nuvem. O mesmo serviço está disponível como **Team Foundation Server (TFS)** na infraestrutura local. Todos os exemplos mostrados neste livro usam o Azure DevOps.

De acordo com a Microsoft, o Azure DevOps é uma plataforma de colaboração baseada na nuvem que ajuda as equipes a compartilhar código, acompanhar o trabalho e enviar software. Azure DevOps é um novo nome; anteriormente, ele era conhecido como **Visual Studio Team Services (VSTS)**. O Azure DevOps é uma ferramenta e um serviço de desenvolvimento de software corporativo que permite que as organizações forneçam instalações de automação ao processo de gerenciamento do ciclo de vida da aplicação de ponta a ponta, desde planejamento até a implantação de aplicações e a obtenção de comentários em tempo real em sistemas de software. Isso aumenta a maturidade e a capacidade de uma organização fornecer sistemas de software de alta qualidade aos seus clientes.

O fornecimento bem-sucedido de software envolve a união eficiente de muitos processos e atividades. Entre eles estão a execução e a implementação de vários processos Agile, o aumento da colaboração entre as equipes, a transição perfeita e automática de artefatos de uma fase de ALM para outra, e implantações em vários ambientes. É importante acompanhar e relatar essas atividades para medir, agir e aprimorar os processos de entrega. O Azure DevOps torna isso simples e fácil. Ele fornece todo um conjunto de serviços que possibilita o seguinte:

- Colaboração entre todos os membros da equipe, fornecendo uma única interface para o gerenciamento de todo o ciclo de vida da aplicação.
- Colaboração entre equipes de desenvolvimento usando serviços de gerenciamento de código-fonte
- Colaboração entre equipes de teste usando serviços de gerenciamento de testes
- Validação automática de código e empacotamento por meio da integração contínua, usando serviços de gerenciamento de builds
- Validação automática de funcionalidade da aplicação, implantação e configuração de vários ambientes por meio de implantação e entrega contínuas usando serviços de gerenciamento de versões
- Acompanhamento e gerenciamento de itens de trabalho usando serviços de gerenciamento de trabalhos

A tabela a seguir mostra todos os serviços disponíveis para um projeto típico na barra de navegação à esquerda do **Azure DevOps**:

Serviço	Descrição
Boards	O Boards ajuda no planejamento do projeto exibindo o progresso atual de tarefas, pendências e histórias de usuários ao lado de informações de sprint. Ele também fornece um processo de Kanban e ajuda a representar as tarefas atuais em andamento e concluídas.
Repos	O Repos ajuda no gerenciamento de repositórios. Ele fornece suporte para criar ramificações adicionais, mesclá-las, resolver conflitos de código e também gerenciar permissões. Pode haver vários repositórios dentro de um projeto.
Pipelines	Os pipelines de compilação e lançamento são criados e gerenciados a partir de pipelines. Eles ajudam a automatizar o processo de compilação e lançamento. Pode haver vários pipelines de compilação e lançamento em um projeto.
Planos de teste	Todos os artefatos relacionados a testes, junto com seu gerenciamento, estão disponíveis nos planos de teste.
Artefatos	Os pacotes NuGet e outros artefatos são armazenados e gerenciados aqui.

Tabela 13.1: uma lista de serviços do Azure DevOps

Uma organização no Azure DevOps funciona como um limite de segurança e um contêiner lógico que fornece todos os serviços necessários para implementar uma estratégia de DevOps. O Azure DevOps permite a criação de vários projetos em uma única organização. Por padrão, um repositório é criado juntamente com o projeto. No entanto, o Azure DevOps permite a criação de repositórios adicionais dentro de um mesmo projeto. A relação entre uma **Organização do Azure DevOps**, **Projetos** e um **Repositório** é mostrada na Figura 13.5:

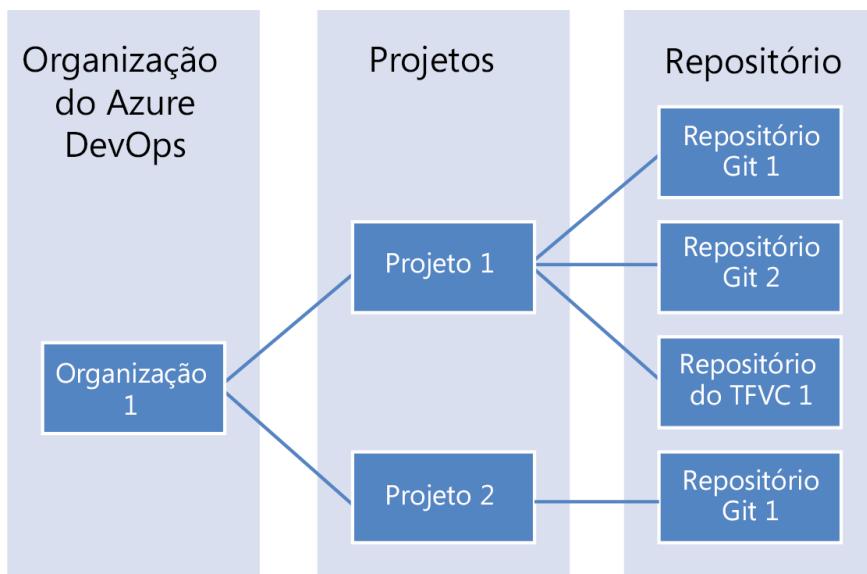


Figura 13.5: relação entre uma Organização do Azure DevOps, Projetos e um Repositório

O Azure DevOps fornece dois tipos de repositório:

- Git
- Controle de Versão do Team Foundation (TFVC)

Ele também fornece a flexibilidade de escolher o repositório de controle de código-fonte do Git ou do TFVC. Pode haver uma combinação de repositórios do TFS e do TFVC disponíveis em um único projeto.

## TFVC

O TFVC é a maneira tradicional e centralizada de implementar o controle de versão na qual há um repositório central e os desenvolvedores trabalham nele diretamente no modo conectado para verificar suas alterações. Se o repositório central estiver offline ou não disponível, os desenvolvedores não poderão verificar seu código e terão que esperar que ele fique online e disponível. Outros desenvolvedores poderão ver somente o código verificado. Os desenvolvedores podem agrupar várias alterações em um único conjunto a fim de verificar alterações de código que são logicamente agrupadas para formar uma única alteração. O TFVC bloqueia os arquivos de código que estão passando por edições. Outros desenvolvedores poderão ler um arquivo bloqueado, mas não poderão editá-lo. Eles deverão aguardar a conclusão da edição anterior e liberar o bloqueio para poder editá-lo. O histórico de verificações e alterações é mantido no repositório central, enquanto os desenvolvedores possuem a cópia funcional dos arquivos, mas não o histórico.

O TFVC funciona muito bem com equipes grandes que estão trabalhando nos mesmos projetos. Isso permite o controle do código-fonte em um local central. Ele também funciona melhor quando projetos têm uma longa duração, pois o histórico é gerenciado em um local central. O TFVC não tem problemas para trabalhar com arquivos grandes e binários.

## Git

Por outro lado, o Git é uma maneira moderna e distribuída de implementar o controle de versão na qual os desenvolvedores podem trabalhar em suas próprias cópias locais de código e histórico no modo offline. Os desenvolvedores podem trabalhar offline no seu clone de código local. Cada desenvolvedor tem uma cópia local do código e todo o seu histórico e trabalha nas suas alterações com esse repositório local. Ele pode confirmar o código no repositório local. Pode se conectar ao repositório central para a sincronização do repositório local, conforme necessário. Isso permite que todos os desenvolvedores trabalhem em qualquer arquivo, pois eles trabalharão na sua cópia local. A ramificação no Git não cria outra cópia do código original, e sua criação é extremamente rápida.

O Git funciona bem com equipes pequenas e grandes. A ramificação e a mesclagem são tranquilas com opções avançadas do Git.

O Git é a maneira recomendada de usar o controle de código-fonte devido à funcionalidade avançada que oferece. Usaremos o Git como o repositório para nossa aplicação de exemplo neste livro. Na próxima seção, vamos ter uma visão geral detalhada da implementação da automação por meio do DevOps.

## Preparar para o DevOps

Seguindo adiante, nosso foco será a automação de implantações e processos usando diferentes padrões no Azure. Entre eles estão:

- DevOps para soluções de IaaS
- DevOps para soluções de PaaS
- DevOps para soluções baseadas em contêiner

Geralmente, há serviços comuns compartilhados que não são exclusivos a nenhuma aplicação. Esses serviços são consumidos por várias aplicações de diferentes ambientes, como desenvolvimento, teste e produção. O ciclo de vida desses serviços comuns compartilhados é diferente para cada aplicação. Portanto, eles têm diferentes repositórios de controle de versão, uma base de código diferente e gerenciamento de lançamento e build. Eles têm seu próprio ciclo de planejamento, design, criação, teste e lançamento.

Os recursos que fazem parte desse grupo são provisionados usando modelos do ARM, o PowerShell e configurações de DSC.

O fluxo geral para a criação desses componentes comuns é mostrado aqui:

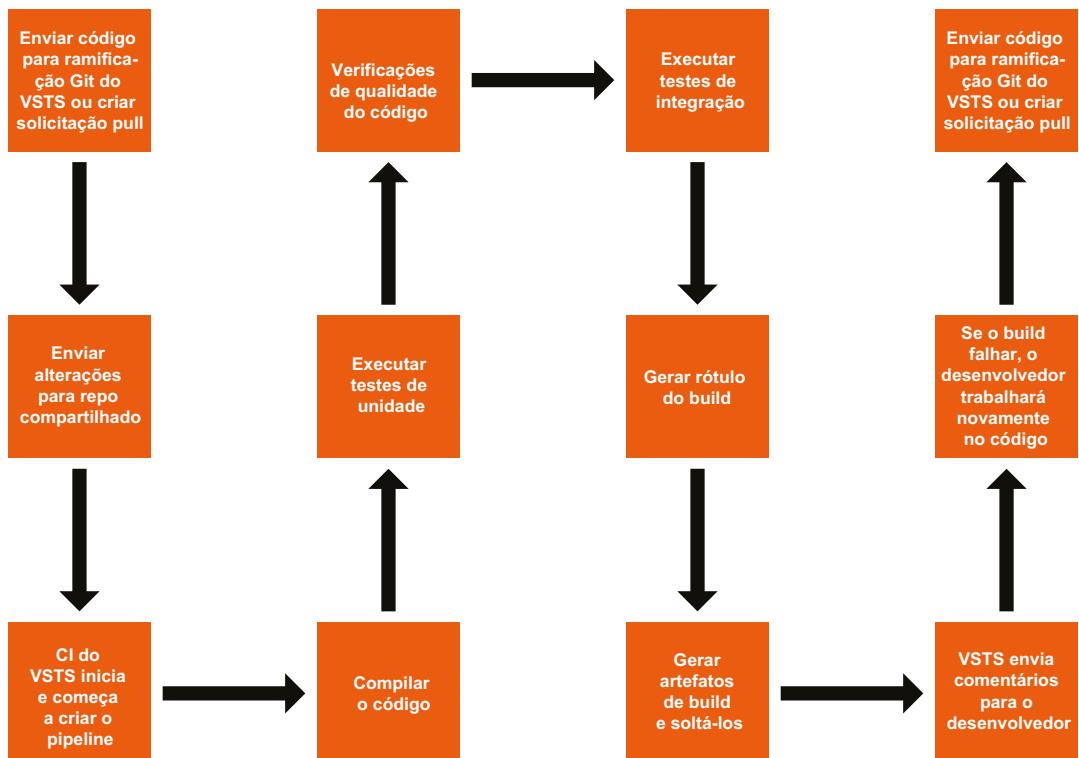


Figura 13.6: fluxo geral para a criação de componentes comuns

O processo de lançamento é mostrado na Figura 13.7:

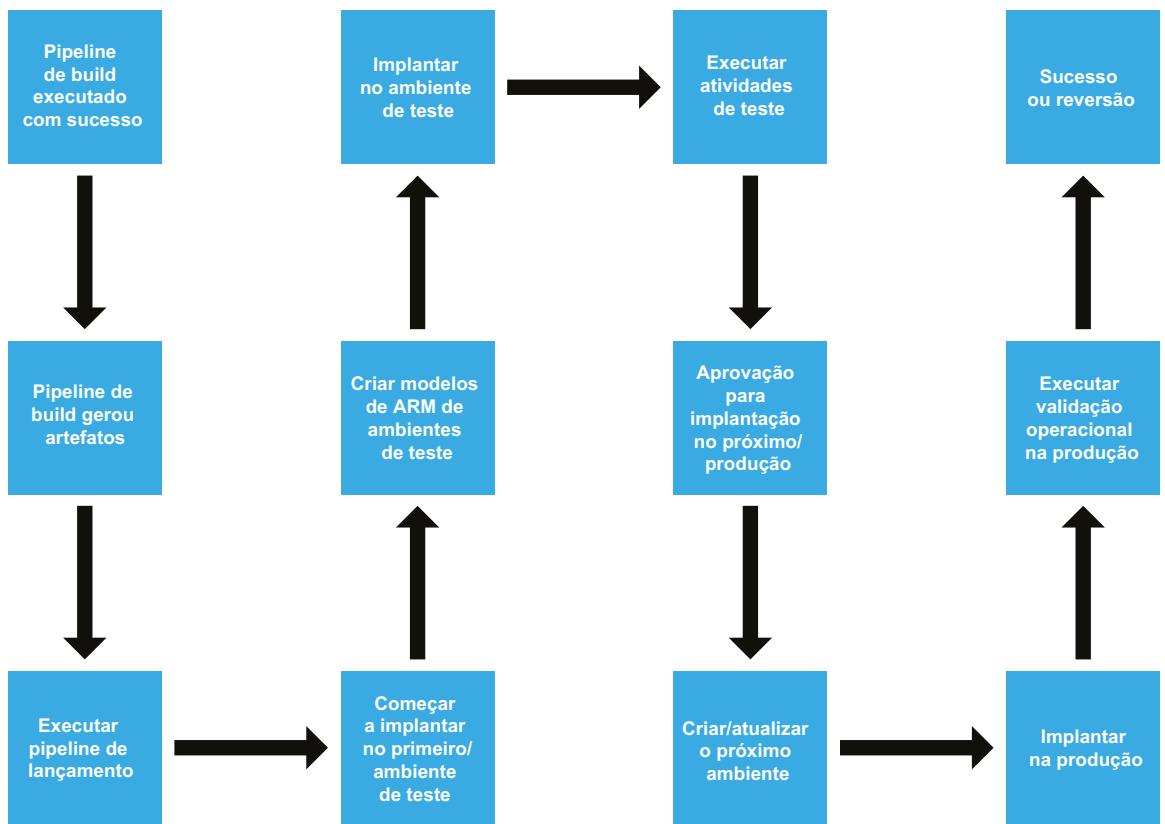


Figura 13.7: processo de lançamento

Na jornada de DevOps, é importante compreender e provisionar os serviços e componentes comuns antes de iniciar qualquer engajamento de software, produto ou serviço.

A primeira etapa para começar a usar o Azure DevOps é provisionar uma organização.

## Organizações do Azure DevOps

Um sistema de controle de versão é necessário para colaborar no nível do código. O Azure DevOps fornece versões centralizadas e descentralizadas de sistemas de controle. O Azure DevOps também fornece serviços de orquestração para a criação e a execução de pipelines de build e de lançamento. É uma plataforma madura que organiza todo controle de versão relacionado ao DevOps e cria e lança artefatos relacionados a itens de trabalho. Depois que uma organização for provisionada no Azure DevOps, um projeto do Azure DevOps deverá ser criado para armazenar todos os artefatos relacionados ao projeto.

Uma organização do Azure DevOps pode ser provisionada acessando <https://dev.azure.com>.

Uma organização do Azure DevOps é o limite administrativo e de gerenciamento de nível superior que fornece segurança, acesso e colaboração entre membros da equipe pertencentes a uma organização. Pode haver vários projetos em uma organização, e cada projeto é composto por várias equipes.

## Provisionar o Azure Key Vault

Não é recomendável armazenar segredos, certificados, credenciais ou outras informações confidenciais em arquivos de configuração de código, bancos de dados, ou qualquer outro sistema de armazenamento geral. É recomendável armazenar esses dados importantes em um cofre criado especificamente para armazenar segredos e credenciais. O Azure Key Vault fornece esse serviço. O Azure Key Vault está disponível como um recurso e serviço do Azure. Agora, vamos avançar para explorar as opções de armazenamento para configurações.

## Provisionar um servidor/serviço de gerenciamento de configuração

Um servidor/serviço de gerenciamento de configuração que fornece armazenamento a ela e aplica essas configurações a diferentes ambientes é sempre uma boa estratégia para automatizar implantações. DSC em máquinas virtuais personalizadas e DSC da Automação do Azure, Chef, Puppet e Ansible são algumas das opções e podem ser usadas no Azure facilmente para ambientes Windows e Linux. Este livro usa o DSC como uma ferramenta de gerenciamento de configuração para todos os finalidades e fornece um servidor de pull que mantém todos os documentos de configuração (arquivos MOF) para a aplicação de exemplo. Ele também mantém o banco de dados de todos os contêineres e máquinas virtuais que estão configurados e registrados no servidor de pull para obter documentos de configuração dele. O gerenciador de configurações local nessas máquinas virtuais de destino e os contêineres verificam periodicamente a disponibilidade de novas configurações, bem como desvios na configuração atual, e os relata de volta ao servidor de pull. Ele também possui recursos internos de relatório que fornecem informações sobre nós que estão em conformidade, bem como aqueles que não estão em conformidade em uma máquina virtual. Um servidor de pull é uma aplicação Web geral que hospeda o ponto de extremidade do servidor de pull do DSC. No próximo tópico, abordaremos uma técnica para monitorar processos em tempo real com o Log Analytics.

## Log Analytics

O Log Analytics é um serviço de auditoria e monitoramento fornecido pelo Azure para obter informações em tempo real sobre todas as alterações, os desvios e os eventos que ocorrem dentro de máquinas virtuais e contêineres. Ele fornece um painel e espaço de trabalho centralizado para que administradores de TI possam exibir, pesquisar e realizar pesquisas detalhadas em todas as alterações, os desvios e eventos que ocorrerem nessas máquinas virtuais. Ele também fornece agentes que são implantados em máquinas virtuais e contêineres de destino. Depois de implantados, esses agentes começam a enviar todas as alterações, os eventos e desvios para o espaço de trabalho centralizado. Vamos conferir as opções de armazenamento para implantar várias aplicações.

## Contas de armazenamento do Azure

O Armazenamento do Azure é um serviço fornecido pelo Azure para armazenar arquivos como blobs. Todos os scripts e códigos para automatizar o provisionamento, a implantação e a configuração da infraestrutura e da aplicação de exemplo são armazenados no repositório Git do Azure DevOps e são empacotados e implantados em uma conta de armazenamento do Azure. O Azure fornece recursos de extensão de scripts do PowerShell que podem automaticamente fazer o download de scripts do DSC e do PowerShell e executá-los em máquinas virtuais durante a execução de modelos do ARM. Esse armazenamento funciona como um armazenamento comum em todas as implantações para várias aplicações. Armazenar scripts e modelos em uma conta de armazenamento garante que eles possam ser usados em projetos, independentemente de projetos no Azure DevOps. Vamos avançar para explorar a importância das imagens na próxima seção.

## Imagens do Docker e do sistema operacional

As imagens de máquinas virtuais e de contêineres devem ser criadas como parte do pipeline de lançamento e de build de serviços comuns. Ferramentas como Packer e Build do Docker podem ser usadas para gerar essas imagens.

## Ferramentas de gerenciamento

Todas as ferramentas de gerenciamento, como Kubernetes, DC/OS, Docker Swarm e ferramentas ITIL devem ser provisionadas antes de criar e implantar a solução.

Concluiremos esta seção sobre como preparar o DevOps com ferramentas de gerenciamento. Há várias opções para cada atividade em um ecossistema de DevOps, e devemos habilitá-las como parte da jornada de DevOps. Não deve ser uma consideração posterior, mas sim parte do planejamento de DevOps.

## DevOps para soluções de PaaS

A arquitetura típica para serviços de aplicativos de PaaS do Azure se baseia na Figura 13.8:

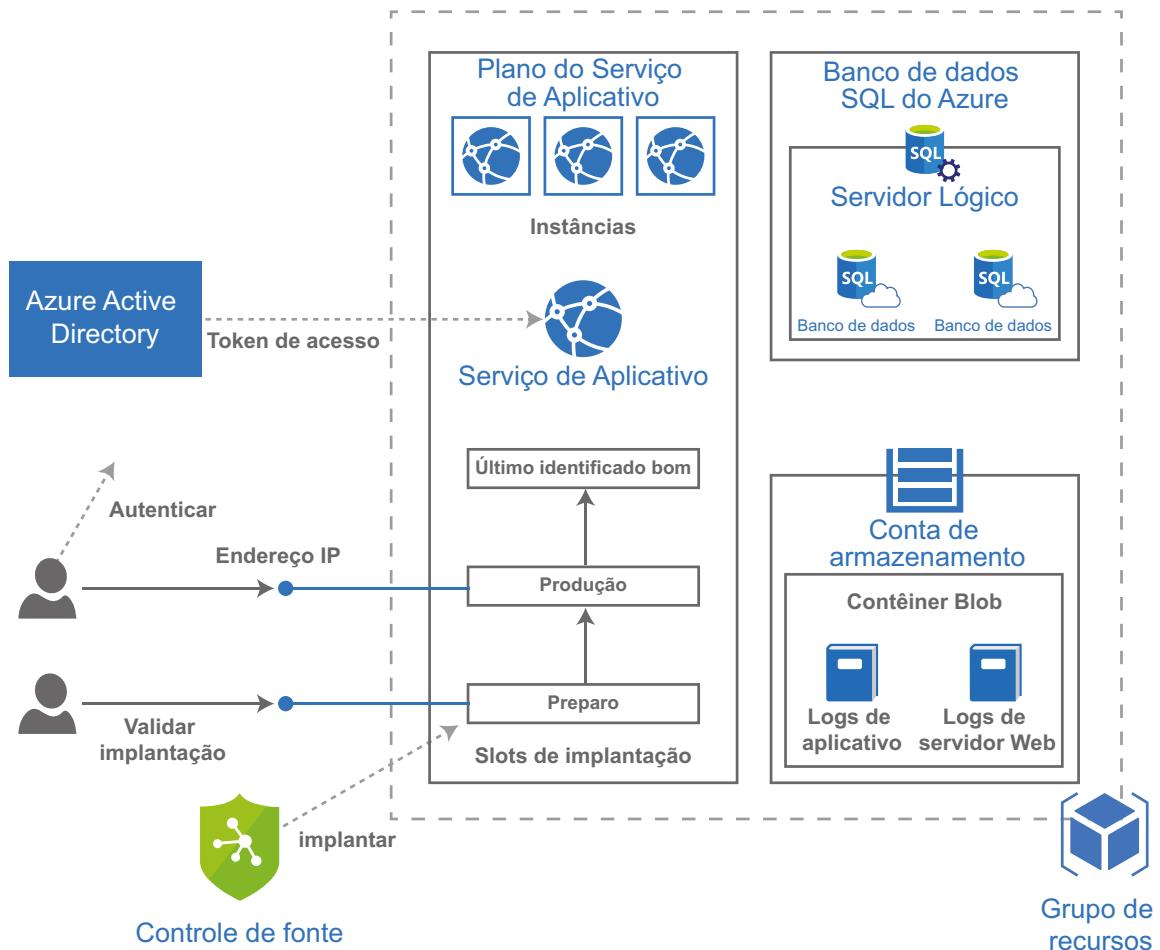


Figura 13.8: uma arquitetura típica do Serviço de Aplicativo de PaaS do Azure

A arquitetura mostra alguns dos componentes importantes, como SQL do Azure, contas de armazenamento e o sistema de controle de versão, que participam da arquitetura de solução de nuvem baseada no Serviço de Aplicativo do Azure. Esses artefatos devem ser criados usando modelos do ARM. Esses modelos do ARM devem fazer parte da estratégia global de gerenciamento de configuração. Ele pode ter seus próprios pipelines de gerenciamento de lançamento e build muito semelhantes àquele mostrado na Figura 13.9:

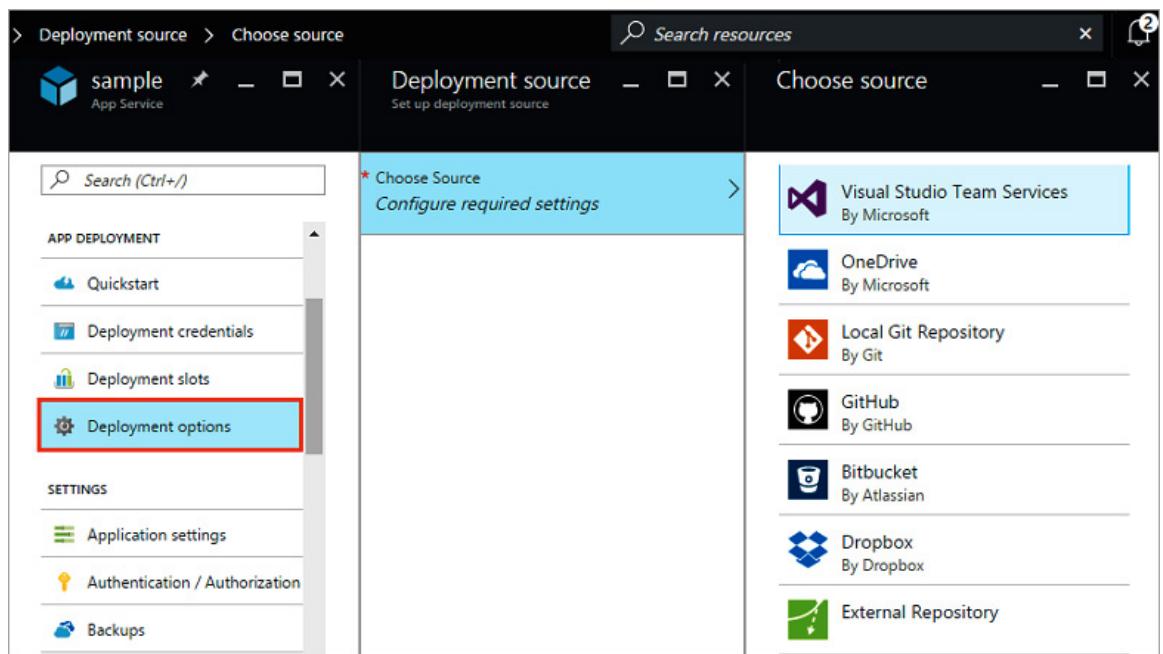


Figura 13.9: escolhendo opções de implantação para o serviço de aplicativo

Agora que exploramos as várias opções de origem de implantação, vamos avançar e nos aprofundar na compreensão de como implantar soluções de nuvem no Azure.

## Serviço de Aplicativo do Azure

O Serviço de Aplicativo do Azure fornece serviços de hospedagem gerenciados para soluções de nuvem. É uma plataforma totalmente gerenciada que provisão e implanta soluções de nuvem. O Serviço de Aplicativo do Azure elimina o fardo de criar e gerenciar a infraestrutura e fornecem **contratos de nível de serviço (SLAs)** mínimos para hospedar suas soluções de nuvem.

## Slots de implantação

O Serviço de Aplicativo do Azure fornece slots de implantação que torna a implantação neles tranquila e fácil. Existem vários slots e a troca entre slots é feita em um nível de DNS. Isso significa que qualquer coisa no slot de produção pode ser trocado com um slot de preparação simplesmente trocando as entradas de DNS. Isso ajuda na implantação inicial da solução de nuvem personalizada na preparação e, após todas as verificações e testes, eles podem ser trocados para produção, se considerados satisfatórios. No entanto, caso ocorram problemas na produção após a troca, os bons valores anteriores do ambiente de produção podem ser restabelecidos por meio de uma nova troca. Vamos avançar para compreender a oferta de banco de dados do Azure e alguns de seus principais recursos.

## SQL do Azure

O SQL do Azure é um serviço PaaS do SQL fornecido pelo Azure para hospedar bancos de dados. O Azure fornece uma plataforma segura para hospedar bancos de dados e assume total propriedade para gerenciar a disponibilidade, a confiabilidade e a escalabilidade do serviço. Com o SQL do Azure, não é necessário provisionar máquinas virtuais personalizadas, implantar um SQL Server e configurá-lo. Em vez disso, a equipe do Azure faz isso nos bastidores e as gerencia em seu nome. Ela também fornece um serviço de firewall que habilita a segurança, e somente um endereço IP permitido pelo firewall pode se conectar ao servidor e acessar o banco de dados. As máquinas virtuais provisionadas para hospedar aplicativos Web têm endereços IP públicos distintos atribuídos a elas e são adicionadas dinamicamente às regras de firewall do SQL do Azure. O SQL Server Azure e seu banco de dados são criados ao executar o modelo do ARM. Em seguida, abordaremos os pipelines de build e lançamento.

## Os pipelines de build e lançamento

Nesta seção, um novo pipeline de build é criado que compila e valida uma aplicação MVC ASP.NET e, em seguida, gera pacotes para implantação. Após a geração do pacote, uma definição de versão garante que a implantação no primeiro ambiente ocorra em um Serviço de Aplicativo e no SQL do Azure como parte da implantação contínua.

Há duas maneiras de criar pipelines de build e lançamento:

1. Usando o editor clássico
2. Usando arquivos YAML

Os arquivos YAML fornecem mais flexibilidade para a criação de pipelines de build e lançamento.

A estrutura do projeto da aplicação de exemplo é mostrada na Figura 13.10:

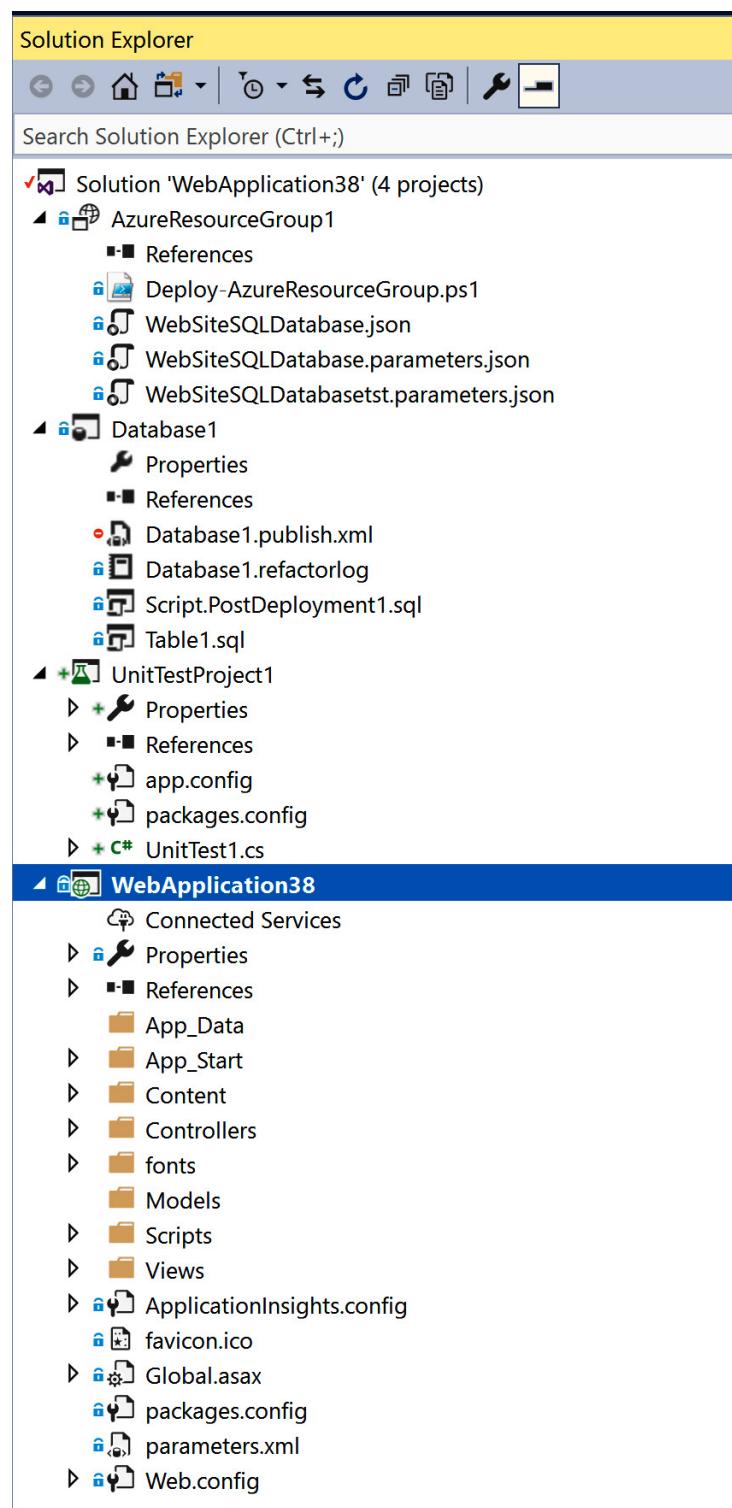
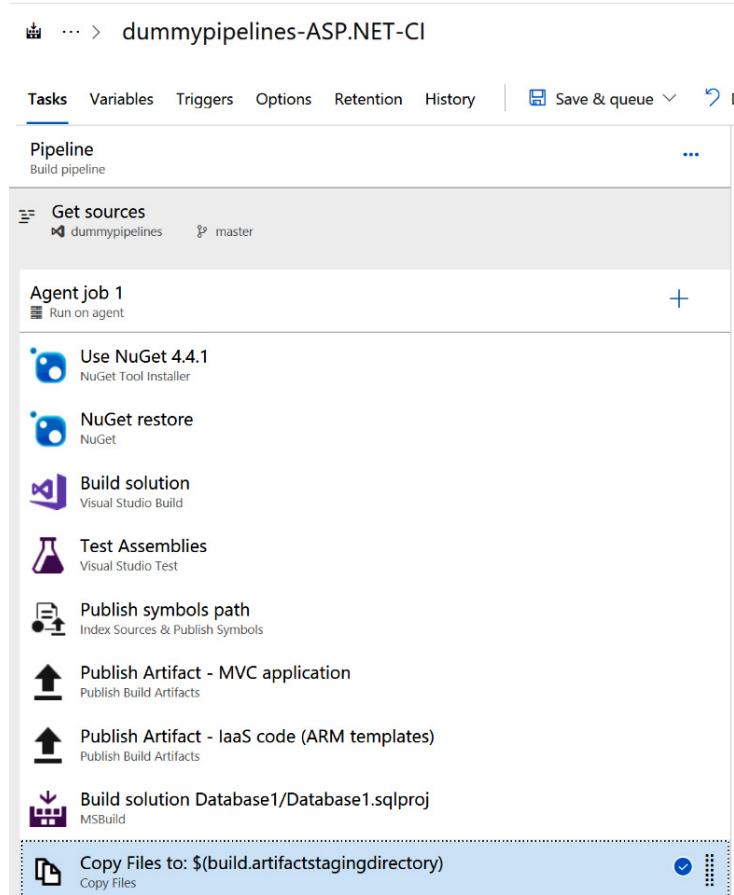


Figura 13.10: estrutura de projeto de uma aplicação de exemplo

Neste projeto, há uma aplicação MVC ASP.NET, a aplicação principal, e consiste em páginas de aplicação. Os pacotes Web Deploy serão gerados fora deste projeto de pipelines de build e eles eventualmente estarão em aplicativos Web do Azure. Há outros projetos que também fazem parte da solução, conforme mencionado a seguir:

- **Projeto de teste de unidade:** código para teste de unidade da aplicação MVC ASP.NET. Os assemblies deste projeto serão gerados e executados na execução de build.
- **Projeto do Banco de Dados SQL:** código relacionado ao esquema, à estrutura e aos dados mestre do banco de dados SQL. Os arquivos **DacPac** serão gerados fora deste projeto usando a definição de build.
- **Projeto de grupo de recursos do Azure:** modelos do ARM e código de parâmetro para provisionar todo o ambiente do Azure no qual a aplicação MVC ASP.NET e as tabelas SQL são criadas.

O pipeline de build é mostrado na *Figura 13.11*:



**Figura 13.11:** pipeline de build da aplicação MVC ASP.NET

A configuração de cada tarefa é mostrada na Tabela 13.2:

Nome da tarefa	Configuração da tarefa
<b>Usar o NuGet 4.4.1</b>	<p>NuGet Tool Installer ⓘ</p> <p>Version <input type="text" value="0.*"/> ⏺</p> <p>Display name * <input type="text" value="Use NuGet 4.4.1"/></p> <p>Version of NuGet.exe to install * ⓘ <input type="text" value="4.4.1"/></p> <p><input type="checkbox"/> Always download the latest matching version ⓘ</p> <p>Control Options ⏺</p> <p>Output Variables ⏺</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>
<b>Restauração do NuGet</b>	<p>NuGet ⓘ</p> <p>Version <input type="text" value="2.*"/> ⏺</p> <p>Display name * <input type="text" value="NuGet restore"/></p> <p>Command * ⓘ <input type="text" value="restore"/></p> <p>Path to solution, packages.config, or project.json * <input type="text" value="**\*.sln"/></p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>
<b>Solução de build</b>	<p>Visual Studio Build ⓘ</p> <p>Version <input type="text" value="1.*"/> ⏺</p> <p>Display name * <input type="text" value="Build solution"/></p> <p>Solution * ⓘ <input type="text" value="WebApplication38/WebApplication38.csproj"/> ...</p> <p>Visual Studio Version ⓘ <input type="text" value="Latest"/></p> <p>MSBuild Arguments ⓘ <input \$(build.artifactstagingdirectory)\""="" type="text" value="/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:PackageLocation="/></p> <p>Platform ⓘ <input type="text" value="\$(BuildPlatform)"/></p> <p>Configuration ⓘ <input type="text" value="\$(BuildConfiguration)"/></p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>

Nome da tarefa	Configuração da tarefa
<b>Assemblies de teste</b>	<p>Visual Studio Test ⓘ</p> <p>Version <input type="text" value="2.*"/> ⏺</p> <p>Display name * Test Assemblies</p> <p>Test selection ⏺</p> <p>Select tests using * ⓘ</p> <p>Test assemblies</p> <p>Test files * ⓘ **\\$(BuildConfiguration)\*test*.dll !**\obj\**</p> <p>Search folder * ⓘ \$(System.DefaultWorkingDirectory)</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>
<b>Publicar caminho de símbolos</b>	<p>Index Sources &amp; Publish Symbols ⓘ</p> <p>Version <input type="text" value="2.*"/> ⏺</p> <p>Display name * Publish symbols path</p> <p>Path to symbols folder ⓘ \$(Build.SourcesDirectory)</p> <p>Search pattern * ⓘ **\bin\**\*.pdb</p> <p><input checked="" type="checkbox"/> Index sources ⓘ <input type="checkbox"/> Publish symbols ⓘ</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>
<b>Publicar Artefato - Aplicação MVC</b>	<p>Publish Build Artifacts ⓘ</p> <p>Version <input type="text" value="1.*"/> ⏺</p> <p>Display name * Publish Artifact - MVC application</p> <p>Path to publish * ⓘ \$(build.artifactstagingdirectory) ⏺</p> <p>Artifact name * drop</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>

Nome da tarefa	Configuração da tarefa
<b>Publicar Artefato - código IaaS (modelos ARM)</b>	<p>Publish Build Artifacts ⓘ</p> <p>Version <input type="text" value="1.*"/> ⏺</p> <p>Display name * Publish Artifact - IaaS code (ARM templates)</p> <p>Path to publish * ⓘ</p> <p>AzureResourceGroup1</p> <p>Artifact name * ⓘ</p> <p>drop1</p> <p>Artifact publish location * ⓘ</p> <p>Azure Pipelines/TFS</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>
<b>Solução de build Database1/Database1.sqlproj</b>	<p>MSBuild ⓘ</p> <p>Version <input type="text" value="1.*"/> ⏺</p> <p>Display name * Build solution Database1/Database1.sqlproj</p> <p>Project * ⓘ</p> <p>Database1/Database1.sqlproj</p> <p>MSBuild ⓘ</p> <p><input checked="" type="radio"/> Version <input type="radio"/> Specify Location</p> <p>MSBuild Version ⓘ</p> <p>Latest</p> <p>MSBuild Architecture ⓘ</p> <p>MSBuild x64</p> <p>Platform ⓘ</p> <p>Configuration ⓘ</p> <p>MSBuild Arguments ⓘ</p> <p>/t:build /p:CmdLineInMemoryStorage=True</p> <p><a href="#">Link settings</a> <a href="#">View YAML</a> <a href="#">Remove</a></p>

Nome da tarefa	Configuração da tarefa
Copiar Arquivos para: \$ (build.artifactstagingdirectory)	<p>Copy Files ⓘ</p> <p>Version 2.*</p> <p>Display name *</p> <p>Copy Files to: \$(build.artifactstagingdirectory)</p> <p>Source Folder ⓘ</p> <p>Contents *</p> <p>**\*.dacpac</p> <p>Target Folder *</p> <p>\$(build.artifactstagingdirectory)</p>

Tabela 13.2: configuração das tarefas do pipeline de build

O pipeline de build é configurado para ser executado automaticamente como parte da integração contínua, conforme mostrado na Figura 13.12:

... > dummypipelines-ASP.NET-CI

Tasks Variables Triggers Options Retention History | Save & queue Discard Summary Queue ...

Continuous integration

dummypipelines Enabled

Scheduled + Add

No builds scheduled

Build completion + Add

Build when another build completes

Enable continuous integration

Batch changes while a build is in progress

Branch filters

Type Branch specification

Include master

+ Add

Path filters

+ Add

Figura 13.12: habilitando a integração contínua no pipeline de build

A definição de versão consiste em vários ambientes, como desenvolvimento, teste, **Teste de integração de sistema (SIT)**, **Teste de aceitação do usuário (UAT)**, pré-produção e produção. As tarefas são bastante semelhantes em cada ambiente, com a adição de tarefas específicas para esse ambiente. Por exemplo, um ambiente de teste tem tarefas adicionais relacionadas à interface do usuário e testes funcionais e de integração, em comparação a um ambiente de desenvolvimento.

A definição de versão para uma aplicação assim é mostrada na Figura 13.13:

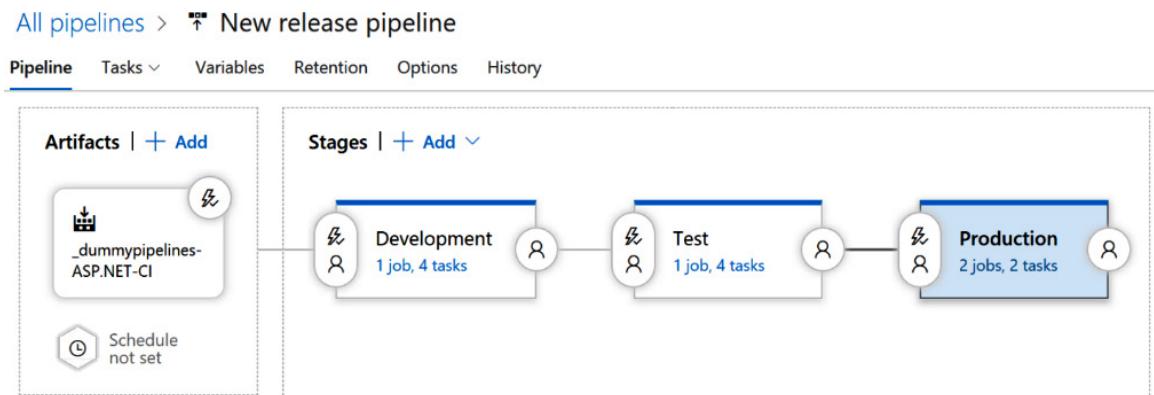


Figura 13.13: definição de lançamento

As tarefas de lançamento para um único ambiente são mostradas na Figura 13.14:

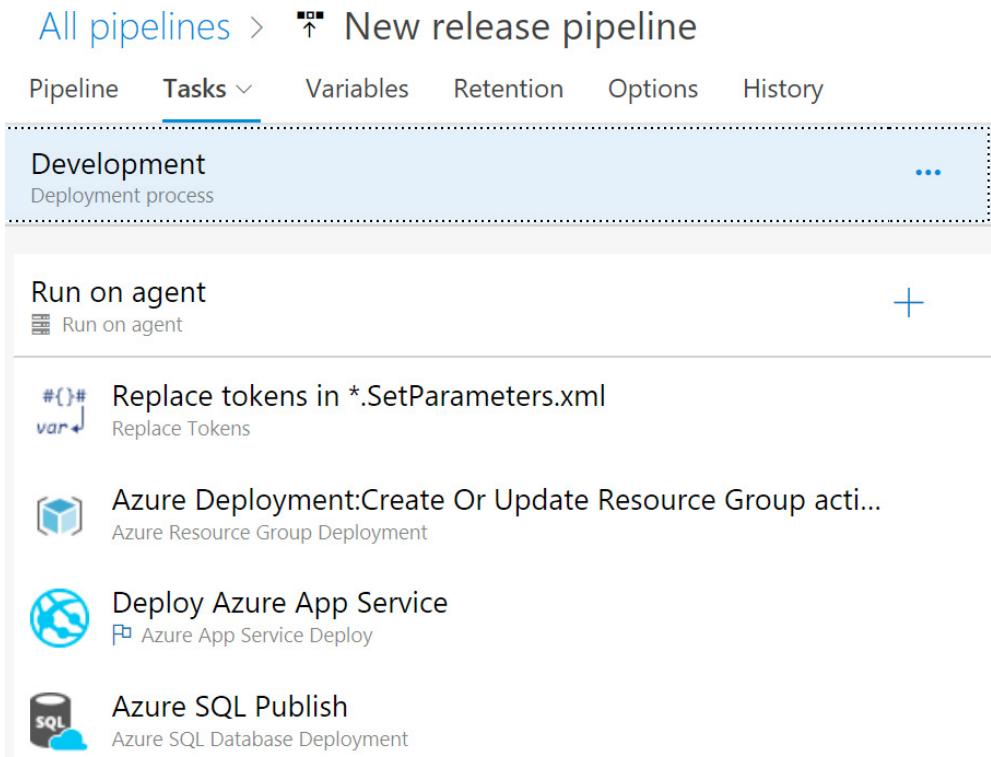


Figura 13.14: tarefas de lançamento para um único ambiente

A configuração para cada uma das tarefas é listada aqui:

Nome da tarefa	Configuração da tarefa
<p><b>Substitua tokens em *.SetParameters.xml</b> (Essa é uma tarefa instalada do MarketPlace).</p>	<p>Version 3.*</p> <p>Display name *</p> <p>Replace tokens in *.SetParameters.xml</p> <p>Root directory ⓘ</p> <p>\$(System.DefaultWorkingDirectory)/_dummypipelines-ASP.NET-CI/drop</p> <p>Target files *</p> <p>*.SetParameters.xml</p> <p>Files encoding *</p> <p>auto</p> <p><input checked="" type="checkbox"/> Write unicode BOM ⓘ</p> <p>Missing variables ^</p> <p>Action *</p> <p>log warning</p> <p><input checked="" type="checkbox"/> Keep token ⓘ</p> <p>Advanced ^</p> <p>Token prefix *</p> <p>—</p> <p>Token suffix *</p> <p>—</p> <p>Empty value ⓘ</p> <p>(empty)</p> <p>Escape values type ⓘ</p> <p>no escaping</p>

Nome da tarefa	Configuração da tarefa
<b>Implantação do Azure: criar ou atualizar ação do Grupo de Recursos no devRG</b>	<p>Azure Resource Group Deployment ⓘ <span style="float: right;">X Remove</span></p> <p>Version <input type="text" value="2.*"/> <span style="margin-left: 10px;">▼</span></p> <p>Display name * <input type="text" value="Azure Deployment:Create Or Update Resource Group action on devRG"/></p> <p>Azure Details ▾</p> <p>Azure subscription * ⓘ   Manage <span style="color: blue;">Manage</span> <span style="margin-left: 10px;">↻</span></p> <p><input type="text" value="myconnection"/> <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p>Scoped to subscription 'Visual Studio Enterprise'</p> <p>Action * ⓘ <span style="margin-left: 10px;">▼</span></p> <p><input type="text" value="Create or update resource group"/> <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p>Resource group * ⓘ <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p><input type="text" value="devRG"/> <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p>Location * ⓘ <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p><input type="text" value="West Europe"/> <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">↻</span></p> <p>Template ▾</p> <p>Template location * <span style="margin-left: 10px;">▼</span></p> <p><input type="text" value="Linked artifact"/> <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">...</span></p> <p>Template * ⓘ <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">...</span></p> <p><input type="text" value="\$(System.DefaultWorkingDirectory)/_dummypipelines-ASP.NET-CI/drop1/WebSiteSQLDatabase.json"/> <span style="margin-left: 10px;">...</span></p> <p>Template parameters <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">...</span></p> <p><input type="text" value="\$(System.DefaultWorkingDirectory)/_dummypipelines-ASP.NET-CI/drop1/WebSiteSQLDatabase.parameters.json"/> <span style="margin-left: 10px;">...</span></p> <p>Override template parameters <span style="margin-left: 10px;">▼</span> <span style="margin-left: 10px;">...</span></p> <p><input type="text" value="-sqlserverName \$(SQLServerName) -hostingPlanName \$(AppServiceName)"/> <span style="margin-left: 10px;">...</span></p> <p>Deployment mode * ⓘ <span style="margin-left: 10px;">▼</span></p> <p><input type="text" value="Incremental"/> <span style="margin-left: 10px;">▼</span></p>

Nome da tarefa	Configuração da tarefa
<b>Implantar o Serviço de Aplicativo do Azure</b>	<p>Azure App Service Deploy ⓘ</p> <p>Version 3.*</p> <p>Display name *</p> <p>Deploy Azure App Service</p> <p>Azure subscription * ⓘ   Manage 🔍</p> <p>myconnection</p> <p>Scoped to subscription 'Visual Studio Enterprise'</p> <p>App type * ⓘ</p> <p>webApp</p> <p>App Service name *</p> <p>\$(AppServiceName)</p> <p><input type="checkbox"/> Deploy to slot ⓘ</p> <p>Virtual application ⓘ</p> <p>Package or folder *</p> <p>\$(System.DefaultWorkingDirectory)/_dummypipelines-ASP.NET-CI/drop/WebApplication38.zip</p>

Nome da tarefa	Configuração da tarefa
<b>Publicação do SQL do Azure</b>	<p>Azure SQL Database Deployment ⓘ <span style="float: right;">X Remove</span></p> <p>Version <input type="text" value="1.*"/> ▾</p> <p>Display name * <input type="text" value="Azure SQL Publish"/></p> <p>Azure Service Connection Type <input type="text" value="Azure Resource Manager"/> ▾</p> <p>Azure Subscription * ⓘ   Manage ↗ <input type="text" value="myconnection"/> ▾ <span style="float: right;">↻</span></p> <p><small>Scoped to subscription 'Visual Studio Enterprise'</small></p> <p>SQL DB Details ▾</p> <p>Azure SQL Server Name * ⓘ <input type="text" value="mdemoclasssql.database.windows.net"/></p> <p>Database Name * ⓘ <input type="text" value="myecommerce"/></p> <p>Server Admin Login * ⓘ <input type="text" value="citynextadmin"/></p> <p>Password * ⓘ <input type="text" value="citynext!1234"/></p> <p>Deployment Package ▾</p> <p>Action * ⓘ <input type="text" value="Publish"/> ▾</p> <p>Type <input type="text" value="SQL DACPAC File"/> ▾</p> <p>DACPAC File * ⓘ <input type="text" value="\$System.DefaultWorkingDirectory/_dummypipelines-ASP.NET-Ci/drop2/Database1/bin/Debug/Database1.dacpac"/> ...</p>

Tabela 13.3: configuração das tarefas do pipeline de lançamento

Nesta seção, você viu maneiras de configurar pipelines de build e lançamento no Azure DevOps. Na próxima seção, o foco será diferentes arquiteturas, como IaaS, contêineres e diferentes cenários.

## DevOps para IaaS

A IaaS envolve o gerenciamento e a administração de infraestrutura e aplicações de base de forma conjunta, e há vários recursos e componentes que precisam ser provisionados, configurados e implantados em vários ambientes. É importante entender a arquitetura antes de prosseguir.

A arquitetura típica para uma solução de IaaS baseada em máquinas virtuais é mostrada aqui:

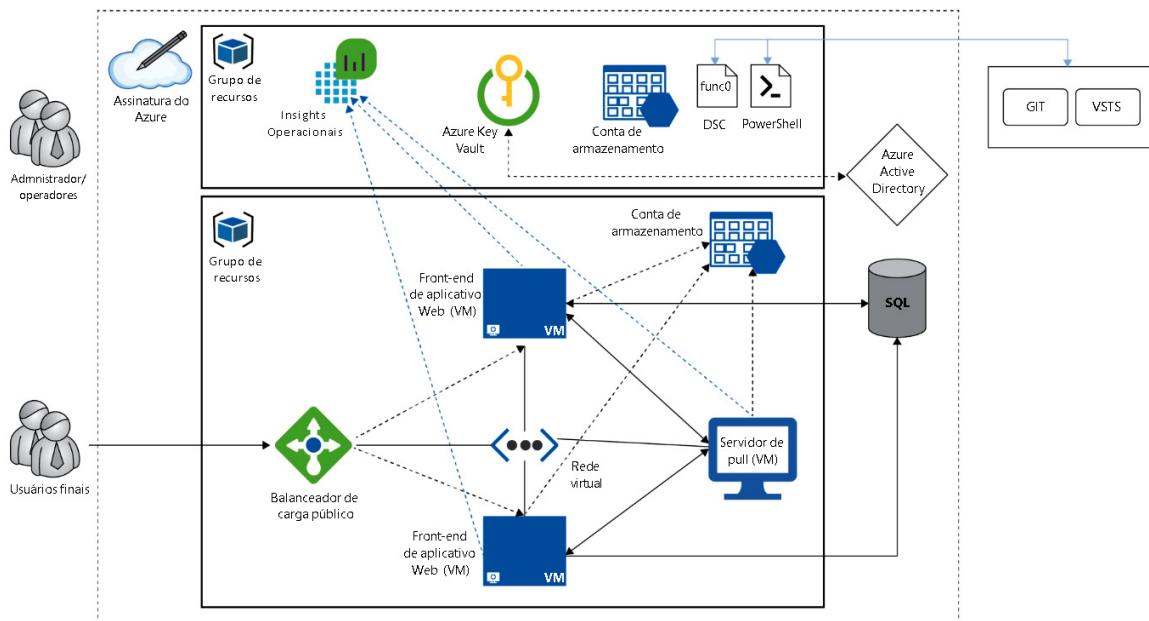


Figura 13.15: arquitetura típica para uma solução de IaaS baseada em máquinas virtuais

Cada um dos componentes listados na arquitetura é abordado a partir da próxima seção.

### Máquinas virtuais do Azure

Máquinas virtuais do Azure que hospedam aplicações Web, servidores de aplicações, bancos de dados e outros serviços são provisionadas usando modelos do ARM. Elas estão conectadas a uma rede virtual e têm um endereço IP privado da mesma rede. O IP público para máquinas virtuais é opcional, pois elas estão conectadas a um balanceador de carga público. Os agentes operacionais do Insights são instalados em máquinas virtuais para monitorá-las. Scripts do PowerShell também são executados nessas máquinas virtuais cujo download foi feito em uma conta de armazenamento disponível em outro grupo de recursos para abrir portas de firewall relevantes, fazer o download de pacotes apropriados e instalar certificados locais para garantir o acesso por meio do PowerShell. O aplicativo Web está configurado para ser executado na porta fornecida nessas máquinas virtuais. O número da porta para o aplicativo Web e toda a sua configuração é obtido do servidor de pull do DSC e atribuído dinamicamente.

## Balanceadores de carga públicos do Azure

Um balanceador de carga público é conectado a algumas das máquinas virtuais para enviar solicitações a elas de maneira alternada. Isso geralmente é necessário para APIs e aplicações Web de front-end. Um endereço IP público e um nome DNS podem ser atribuídos a um balanceador de carga de modo que ele possa atender a solicitações da Internet. Ele aceita solicitações HTTP da Web em portas diferentes e as encaminha para as máquinas virtuais. Ele também explora determinadas portas em protocolos HTTP com alguns caminhos de aplicação fornecidos. As regras de **Conversão de Endereço de Rede (NAT)** também podem ser aplicadas de modo que possam ser usadas para fazer logon nas máquinas virtuais usando áreas de trabalho remotas.

Um recurso alternativo ao balanceador de carga público do Azure é o Gateway de Aplicativo do Azure. Os gateways de aplicativo são平衡adores de carga de camada 7 e fornecem recursos como terminação SSL, afinidade de sessão e roteamento baseado em URL. Vamos abordar o pipeline de build na próxima seção.

## O pipeline de build

Um pipeline de build típico para uma solução de IaaS baseada em máquinas virtuais é mostrado a seguir. Um pipeline de lançamento é iniciado quando um desenvolvedor envia seu código ao repositório. O pipeline de build é iniciado automaticamente como parte da integração contínua. Ele compila e cria o código, realiza testes de unidade e verificações de qualidade nele, verifica a qualidade do código e gera a documentação desde os comentários do código. Ele implanta os novos binários no ambiente de desenvolvimento (observe que esse ambiente não é recém-criado), altera configurações, realiza testes de integração e gera rótulos de build para fácil identificação. Em seguida, ele deposita os artefatos gerados em um local acessível pelo pipeline de lançamento. Caso haja algum problema durante a execução de qualquer etapa desse pipeline, isso será comunicado ao desenvolvedor como parte dos comentários do pipeline de build para que possa retrabalhar e reenviar suas alterações. O pipeline de build falhará ou será aprovado com base na gravidade dos problemas encontrados, e isso varia de acordo com a organização. Um pipeline de build típico é mostrado na Figura 13.16:

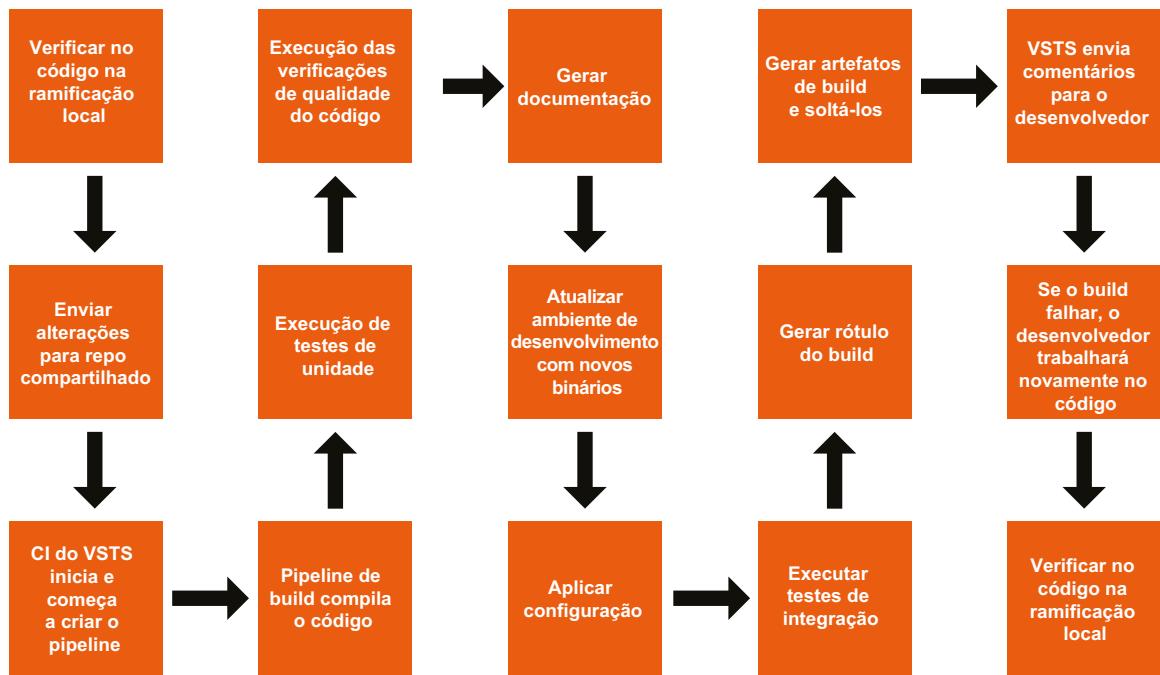


Figura 13.16: um pipeline de build típico para IaaS

Semelhante ao pipeline de build, vamos aprender sobre a implementação de um pipeline de lançamento.

## O pipeline de lançamento

Um pipeline de lançamento típico para uma implantação de IaaS baseada em máquinas virtuais é mostrado a seguir. Um pipeline de lançamento é iniciado após a conclusão do pipeline de build. A primeira etapa no pipeline de lançamento é a coleta dos artefatos gerados pelo pipeline de build. Eles geralmente são documentos de configuração, binários e assemblies implantáveis. O pipeline de lançamento executa e cria ou atualiza o primeiro ambiente que geralmente é um ambiente de teste. Ele usa modelos do ARM para provisionar todos os recursos e serviços IaaS e PaaS no Azure, além de configurá-los. Ele também ajuda na execução de scripts e na configuração do DSC após a criação de máquinas virtuais como etapas de pós-criação. Isso ajuda a configurar o ambiente dentro da máquina virtual e do sistema operacional. Nesse estágio, os binários de aplicações do pipeline de build são implantados e configurados. Diferentes testes automatizados são realizados para verificar o funcionamento na solução, e, se considerado satisfatório, o pipeline passa para a implantação no próximo ambiente após obter as aprovações necessárias. As mesmas etapas são executadas novamente no próximo ambiente, incluindo o ambiente de produção. Por fim, os testes de validação operacional são realizados na produção para garantir que a aplicação esteja funcionando conforme o esperado e que não haja desvios.

Nesse estágio, se houver problemas ou bugs, eles deverão ser corrigidos e o ciclo inteiro deverá ser repetido. No entanto, se isso não acontecer dentro de um período estipulado, o último instantâneo conhecido deverá ser restaurado no ambiente de produção para minimizar o tempo de inatividade. Um pipeline de lançamento típico é mostrado na Figura 13.17:

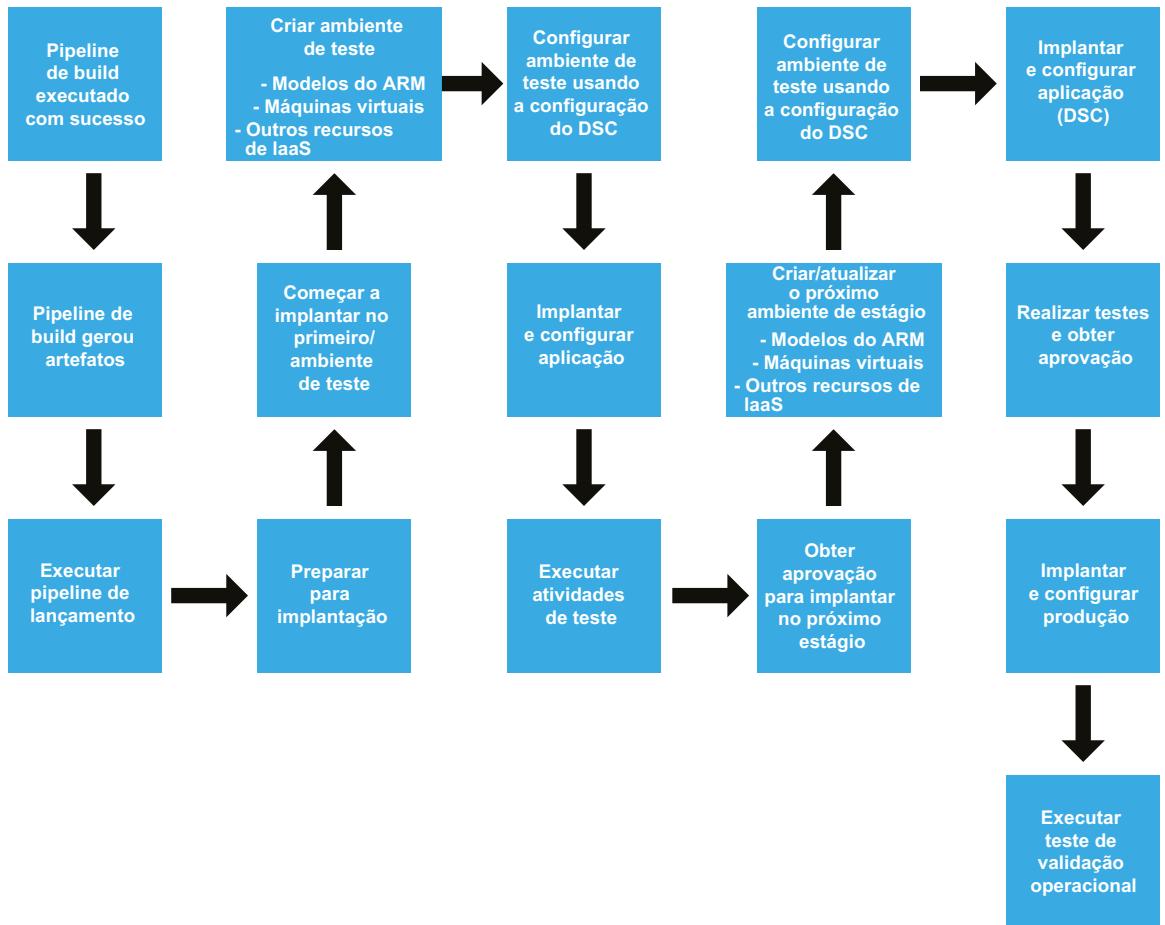


Figura 13.17: um pipeline de lançamento típico para IaaS

Esta seção conclui o processo de DevOps para soluções de IaaS, e o próximo capítulo se concentrará em contêineres em máquinas virtuais. Observe que os contêineres também podem ser executados em PaaS, como o Serviço de Aplicativo e o Azure Functions.

## DevOps com contêineres

Na arquitetura típica, os tempos de execução do contêiner são implantados em máquinas virtuais e os contêineres são executados dentro deles. A arquitetura típica para soluções baseadas em contêineres de IaaS é mostrada aqui:

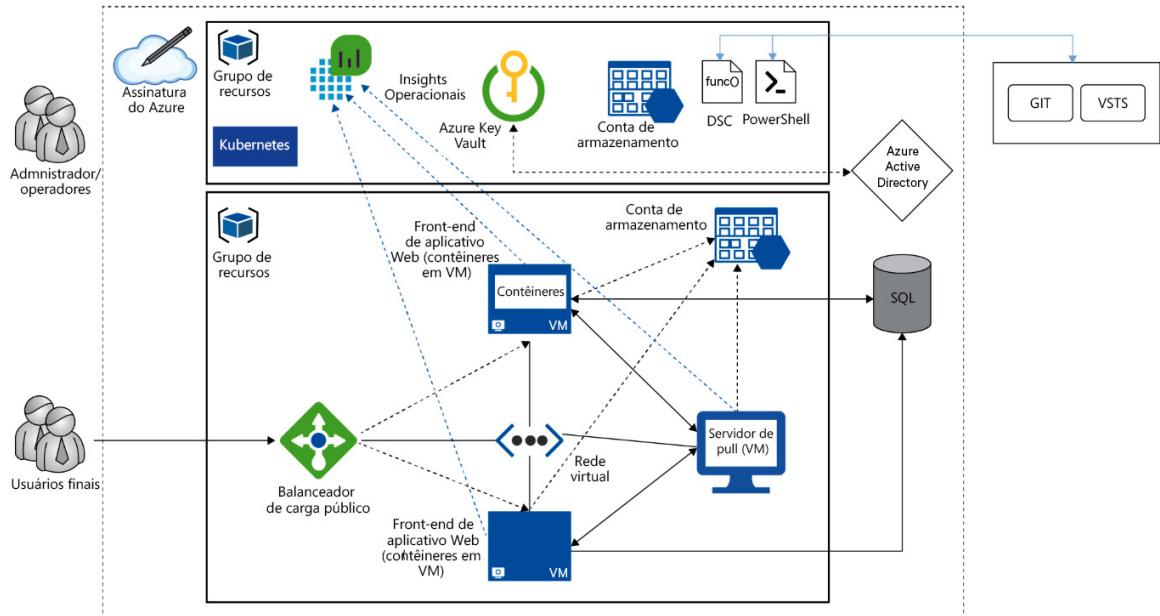


Figura 13.18: arquitetura para soluções baseadas em contêineres de IaaS

Esses contêineres são gerenciados por orquestradores de contêiner, como Kubernetes. Os serviços de monitoramento são fornecidos pelo Log Analytics e todos os segredos e chaves são armazenados no Azure Key Vault. Há também um servidor de pull, que pode estar em uma máquina virtual ou na Automação do Azure, fornecendo informações de configuração para as máquinas virtuais.

### Contêineres

Os contêineres são uma tecnologia de virtualização. No entanto, eles não virtualizam servidores físicos. Em vez disso, contêineres são uma virtualização no nível do sistema operacional. Isso significa que os contêineres compartilham o kernel do sistema operacional fornecido pelo host entre eles e com o host. A execução de vários contêineres em um host (físico ou virtual) compartilha o kernel do sistema operacional do host. Há um único kernel de sistema operacional fornecido pelo host e usado por todos os contêineres em execução sobre ele.

Os contêineres também são completamente isolados de seu host e de outros contêineres, assim como uma máquina virtual. Os contêineres usam namespaces do sistema operacional, grupos de controle no Linux, para fornecer a percepção de um novo ambiente de sistema operacional e usar técnicas específicas de virtualização do sistema operacional no Windows. Cada contêiner tem sua própria cópia dos recursos do sistema operacional.

## Docker

O Docker fornece recursos de gerenciamento a contêineres. Ele é composto por dois executáveis:

- Daemon do Docker
- Cliente do Docker

O daemon do Docker é útil para o gerenciamento de contêineres. É um serviço de gerenciamento responsável por gerenciar todas as atividades no host relacionadas a contêineres. O cliente do Docker interage com o daemon do Docker e é responsável pela captura de entradas e pelo seu envio ao daemon do Docker. O daemon do Docker fornece tempo de execução, bibliotecas, drivers de gráficos, os mecanismos para criar, gerenciar e monitorar contêineres e imagens no servidor host. Ele também pode criar imagens personalizadas que são usadas para criar e enviar aplicações para vários ambientes.

## O Dockerfile

O **Dockerfile** é o bloco de construção fundamental na criação de imagens de contêiner do Windows. É um arquivo simples baseado em texto e legível por humanos sem nenhuma extensão, e é até chamado de **Dockerfile**. Embora haja um mecanismo para chamá-lo de forma diferente, geralmente é chamado de **Dockerfile**. O Dockerfile contém instruções para criar uma imagem personalizada usando uma imagem base. Essas instruções são executadas sequencialmente de cima para baixo pelo daemon do Docker. As instruções referem-se ao comando e seus parâmetros, como **COPY**, **ADD**, **RUN** e **ENTRYPOINT**. O Dockerfile possibilita as práticas de IaC, convertendo a configuração e a implantação da aplicação em instruções que podem ter a versão controlada e ser armazenadas em um repositório de código-fonte. Vamos conferir as etapas de build na seção a seguir.

## O pipeline de build

Da perspectiva do build, não há nenhuma diferença entre o contêiner e uma solução baseada em máquinas virtuais. A etapa de build permanece a mesma. Um pipeline de lançamento típico para uma implantação de IaaS baseada em contêineres é mostrado a seguir.

## O pipeline de lançamento

A única diferença entre um pipeline de lançamento típico para uma implantação baseada em contêineres de IaaS, e o pipeline de lançamento é o gerenciamento de imagens de contêiner e a criação de contêineres usando o Dockerfile e o Docker Compose. Os utilitários avançados de gerenciamento de contêineres avançados, como Docker Swarm, DC/OS e Kubernetes, também podem ser implantados e configurados como parte do gerenciamento de versões. No entanto, note que essas ferramentas de gerenciamento de contêineres devem fazer parte do pipeline de lançamento de serviços compartilhados, conforme abordado anteriormente. A Figura 13.19 mostra um pipeline de lançamento típica para uma solução baseada em contêineres:

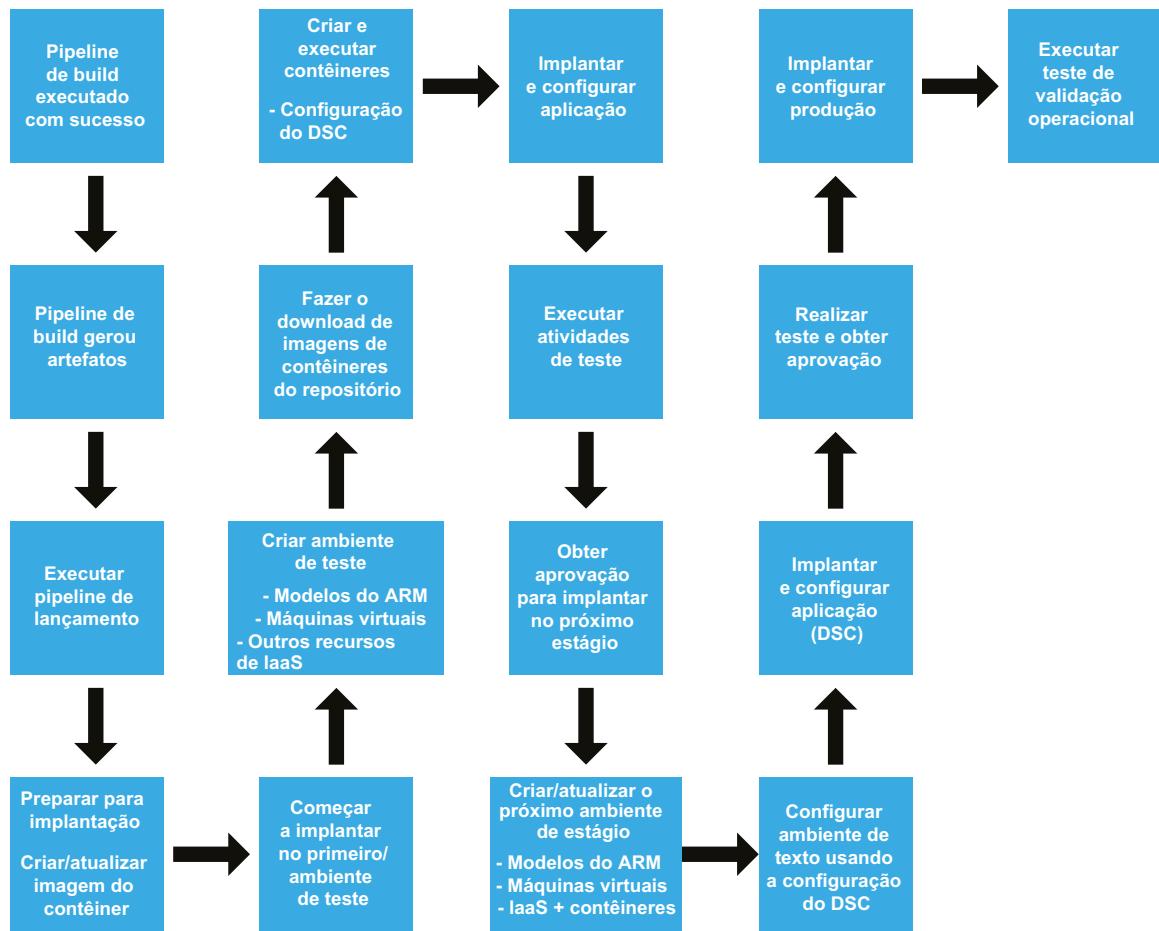


Figura 13.19: pipeline de lançamento baseado em contêineres

O foco da próxima seção é a integração com outros conjuntos de ferramentas, como o Jenkins.

## Azure DevOps e Jenkins

O Azure DevOps é um orquestrador de plataforma aberta que se integra perfeitamente a outras ferramentas de orquestrador. Ele fornece toda a infraestrutura necessária e recursos que se integram bem com Jenkins, também. As organizações com pipelines de CI/CD bem estabelecidos criados no Jenkins podem reutilizá-las com os recursos avançados mas simples do Azure DevOps para orquestrar-los.

O Jenkins pode ser usado como um repositório e pode executar pipelines de CI/CD no Azure DevOps, enquanto também é possível ter um repositório no Azure DevOps e executar pipelines de CI/CD no Jenkins.

A configuração do Jenkins pode ser adicionada ao Azure DevOps como ganchos de serviço e, sempre que qualquer alteração de código for confirmada no repositório do Azure DevOps, ele poderá acionar pipelines no Jenkins. A Figura 13.20 mostra a configuração do Jenkins da seção de configuração de gancho de serviço do Azure DevOps:

The screenshot shows the 'Project Settings > Service hooks' page in the Azure DevOps interface. The left sidebar has tabs for General, Overview, Teams, Security, Notifications, Service hooks (which is selected and highlighted in blue), and Dashboards. The main area is titled 'Service Hooks' with the sub-instruction 'Integrate with your favorite services by notifying them when events happen in your project.' Below this is a blue 'Create subscription' button.

Figura 13.20: configuração do Jenkins

Há vários gatilhos que executam os pipelines no Jenkins; um deles é o **Código enviado**, conforme mostrado na Figura 13.21:

The screenshot shows the 'Trigger' configuration screen for a new service hook subscription. The title is 'NEW SERVICE HOOKS SUBSCRIPTION'. The main section is titled 'Trigger' with the sub-instruction 'Select an event to trigger on and configure any filters.' A dropdown menu 'Trigger on this type of event' is set to 'Code pushed'. A note below says: 'Remember that selected events are visible to users of the target service, even if they don't have permission to view the related artifact.' The 'FILTERS' section includes fields for 'Repository' (set to 'dummypipelines'), 'Branch' (set to 'master'), and 'Pushed by a member of group' (set to '[Any]'), all marked as optional. At the bottom are buttons for 'Previous', 'Next', 'Test', 'Finish', and 'Cancel'.

Figura 13.21: gatilho de código enviado executado

Também é possível implantar na VM do Azure e executar pipelines de lançamento do Azure DevOps, conforme explicado aqui: <https://docs.microsoft.com/azure/virtual-machines/linux/tutorial-build-deploy-jenkins>.

O Jenkins já deve ter sido implantado antes de usá-lo em qualquer cenário. O processo de implantação no Linux pode ser encontrado em <https://docs.microsoft.com/azure/virtual-machines/linux/tutorial-jenkins-github-docker-cicd>.

A próxima seção se concentrará mais em ferramentas e serviços relacionados ao gerenciamento de configuração. A automação do Azure fornece serviços relacionados ao DSC, como o servidor de pull.

## Automação do Azure

A Automação do Azure é a plataforma da Microsoft para todas as implementações de automação referentes a implantações na nuvem, na infraestrutura local e híbridas.

A Automação do Azure é uma plataforma de automação madura que fornece recursos avançados para:

- Definição de ativos, como variáveis, conexões, credenciais, certificados e módulos
- Implementação de runbooks usando Python, scripts do PowerShell e fluxos de trabalho do PowerShell
- Fornecimento de IUs para criar runbooks
- Gerenciamento do ciclo de vida completo do runbook, incluindo criação, testes e publicação
- Agendamento de runbooks
- A capacidade de executar runbooks em qualquer lugar, na nuvem ou na infraestrutura local
- DSC como uma plataforma de gerenciamento de configuração
- Gerenciamento e configuração de ambientes: Windows e Linux, aplicações e implantação
- A capacidade de estender a Automação do Azure importando módulos personalizados

A Automação do Azure fornece um servidor de pull do DSC que ajuda na criação de um servidor de gerenciamento de configuração centralizado que consiste em configurações para nós/máquinas virtuais e seus componentes.

Ele implementa o padrão hub e spoke no qual os nós podem se conectar ao servidor de pull do DSC, fazer o download das configurações atribuídas a eles e se reconfigurar para refletir o estado desejado. Quaisquer alterações ou desvios dentro desses nós serão autocorrigidos por agentes do DSC na próxima vez que forem executados. Isso garante que os administradores não precisem monitorarativamente o ambiente para encontrar desvios.

O DSC fornece uma linguagem declarativa, na qual você define a intenção e a configuração, mas não como executar e aplicar essas configurações. Essas configurações são baseadas na linguagem do PowerShell e facilitam o processo de gerenciamento de configuração.

Nesta seção, analisaremos uma implementação simples de uso do DSC da Automação do Azure para configurar uma máquina virtual com o objetivo de instalar e configurar o servidor Web (IIS) e criar um arquivo `index.htm` que informe aos usuários que o site está em manutenção.

Em seguida, você aprenderá a provisionar uma conta da Automação do Azure.

## Provisionar uma conta da Automação do Azure

Crie uma nova conta da Automação do Azure no portal do Azure ou no PowerShell dentro de um grupo de recursos novo ou existente. Você pode notar na Figura 13.22 que a Automação do Azure fornece itens de menu para o DSC:

The screenshot shows the 'bookaccount' State configuration (DSC) blade. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Configuration Management (Inventory, Change tracking, State configuration (DSC)), Update management (Update management), and Help. The main area has a search bar and buttons for Add, Compose configuration, Refresh, and Reset filters. A central summary section displays 'Nodes' (1), 'Configuration status' (Failed: 0, Pending: 0, Not compliant: 0, In progress: 0, Unresponsive: 1, Compliant: 0), and tabs for Configurations, Compiled configurations, and Gallery. Below this is a table with columns Node, Status, Node configuration, and Last seen, showing one row for 'spark' with status 'Unresponsive' and configuration 'ensureiis.localhost'. At the bottom, there are dropdowns for Status and Node configuration.

Figura 13.22: DSC em uma conta da Automação do Azure

Ela oferece o seguinte:

- **Nós do DSC:** listam todos os contêineres e máquinas virtuais que estão registrados no atual servidor de pull do DSC da Automação do Azure. Esses contêineres e máquinas virtuais são gerenciados usando as configurações do atual servidor de pull do DSC atual.
- **Configurações do DSC:** listam todas as configurações brutas do PowerShell importadas e carregadas no servidor de pull do DSC. Elas estão em formato legível por humanos e não estão em um estado compilado.
- **Configurações de nós do DSC:** listam todas as compilações de configurações do DSC disponíveis no servidor de pull a serem atribuídas a nós, máquinas virtuais e contêineres. Uma configuração do DSC produz arquivos MOF após compilações, e eles eventualmente são usados para configurar nós.

Depois de provisionar uma conta da Automação do Azure, podemos criar uma configuração do DSC de exemplo, conforme mostrado na próxima seção.

## Criar configuração do DSC

A próxima etapa é a criação de uma configuração do DSC usando qualquer editor do PowerShell para refletir a intenção da configuração. Para este exemplo, uma única configuração, **ConfigureSiteOnIIS**, é criada. Ela importa o módulo de base do DSC, **PSDesiredStateConfiguration**, que consiste em recursos usados dentro da configuração. Ela também declara um servidor Web de nó. Quando essa configuração for carregada e compilada, ela gerará uma configuração do DSC chamada **ConfigureSiteOnIISwebserver**. Em seguida, essa configuração poderá ser aplicada aos nós.

A configuração consiste em alguns recursos. Esses recursos configuram o nó de destino. Os recursos instalam um servidor Web, o ASP.NET e a estrutura, e criam um arquivo **index.htm** no diretório **inetpub\wwwroot** com conteúdo para mostrar que o site está em manutenção. Para obter mais informações sobre a criação da configuração do DSC, consulte <https://docs.microsoft.com/powershell/scripting/dsc/getting-started/wingettingstarted?view=powershell-7>.

A próxima listagem de código mostra toda a configuração descrita no parágrafo anterior. O upload dessa configuração feito na conta da Automação do Azure:

```
Configuration ConfigureSiteOnIIS {
    Import-DscResource -ModuleName 'PSDesiredStateConfiguration'
    Node WebServer {
        WindowsFeature IIS
        {
            Name = "Web-Server"
            Ensure = "Present"
        }
        WindowsFeature AspDotNet
        {
            Name = "net-framework-45-Core"
            Ensure = "Present"
            DependsOn = "[WindowsFeature]IIS"
        }
    }
}
```

```

WindowsFeature AspNet45
{
    Ensure          = "Present"
    Name            = "Web-Asp-Net45"
    DependsOn      = "[WindowsFeature]AspNetDotNet"
}

File IndexFile
{
    DestinationPath = "C:\inetpub\wwwroot\index.htm"
    Ensure          = "Present"
    Type            = "File"
    Force           = $true
    Contents        = "<HTML><HEAD><Title> Website under construction.</
Title></HEAD><BODY> '"
                    <h1>If you are seeing this page, it means the website is under
maintenance and DSC Rocks !!!!!</h1></BODY></HTML>"
}
}
}

```

Depois de criar uma configuração do DSC de exemplo, ela deve ser importada na Automação do Azure, conforme mostrado na próxima seção.

## Importar a configuração do DSC

A configuração do DSC ainda não é conhecida pela Automação do Azure. Ela está disponível em alguns computadores locais. Ela deve ser carregada nas configurações do DSC da Automação do Azure. A Automação do Azure fornece o cmdlet **Import-AzureRmAutomationDscConfiguration** para importar a configuração:

```
Import-AzureRmAutomationDscConfiguration -SourcePath "C:\DSC\AA\DSConfigurations\
ConfigureSiteOnIIS.ps1" -ResourceGroupName "omsauto" -AutomationAccountName
"datacenterautomation" -Published -Verbose
```

Os comandos importarão a configuração na Automação do Azure. Após a importação, a configuração do DSC deve ser compilada para que possa ser atribuída a servidores para verificações de conformidade e autorremediação.

## Compilar a configuração do DSC

Depois que a configuração do DSC ficar disponível na Automação do Azure, ela poderá ser compilada. A Automação do Azure fornece outro cmdlet para isso. Use o cmdlet **Start-AzureRmAutomationDscCompilationJob** para compilar a configuração importada. O nome da configuração deve corresponder exatamente ao nome da configuração enviada no upload. O build cria um arquivo MOF cujo nome é a combinação do nome da configuração e do nó, que, nesse caso, é o servidor Web **ConfigureSiteOnIIS**. A execução do comando é mostrada aqui:

```
Start-AzureRmAutomationDscCompilationJob -ConfigurationName ConfigureSiteOnIIS  
-ResourceGroupName "omsauto" -AutomationAccountName "datacenterautomation"  
-Verbose
```

Agora, você realizou a configuração de nós do DSC. Na próxima seção, você aprenderá a atribuir configurações aos nós.

## Atribuir configurações a nós

As configurações do DSC compiladas podem ser aplicadas aos nós.

Use **Register-AzureRmAutomationDscNode** para atribuir a configuração a um nó.

O parâmetro **NodeConfigurationName** identifica o nome de configuração que deve ser aplicado ao nó. Esse é um cmdlet poderoso que também pode configurar o agente do DSC **localconfigurationmanager** em nós antes que possam fazer o download das configurações e aplicá-las. Há vários parâmetros **localconfigurationmanager** que podem ser configurados. Veja detalhes em <https://devblogs.microsoft.com/powershell/understanding-meta-configuration-in-windows-powershell-desired-state-configuration>.

Vamos resolver a configuração abaixo:

```
Register-AzureRmAutomationDscNode -ResourceGroupName "omsauto"  
-AutomationAccountName "datacenterautomation" -AzureVMName testtwo  
-ConfigurationMode ApplyAndAutocorrect -ActionAfterReboot ContinueConfiguration  
-AllowModuleOverwrite $true -AzureVMResourceGroup testone -AzureVMLocation  
"West Central US" -NodeConfigurationName "ConfigureSiteOnIIS.WebServer"  
-Verbose
```

Agora, podemos testar se a configuração foi aplicada aos servidores navegando no site recém-implantado usando um navegador. Depois que os testes forem concluídos com êxito, vamos avançar para validar as conexões.

## Validação

Se apropriado, os grupos de segurança de rede e firewalls são abertos e habilitados para a porta 80, e um IP público é atribuído à máquina virtual. O site padrão pode ser navegado usando o endereço IP. Caso contrário, faça logon na máquina virtual que é usada para aplicar a configuração do DSC e navegue até <http://localhost>.

Isso deve mostrar a seguinte página:



Figura 13.23: localhost

Esse é o poder do gerenciamento de configuração: sem escrever nenhum código significativo, a criação de uma única configuração pode ser aplicada várias vezes ao mesmo servidor e a vários outros, e você pode ter a certeza de que funcionarão no estado desejado sem nenhuma intervenção manual. Na próxima seção, vamos conferir as várias ferramentas disponíveis para o Azure DevOps.

## Ferramentas para DevOps

Como mencionado anteriormente, o Azure é uma plataforma sofisticada e madura que oferece suporte ao seguinte:

- Várias opções de linguagens
- Várias opções de sistemas operacionais
- Várias opções de ferramentas e utilitários
- Vários padrões para implantação de soluções (como máquinas virtuais, serviços de aplicativos, contêineres e microsserviços)

Com tantas opções e escolhas, o Azure oferece o seguinte:

- **Nuvem aberta:** aberta para serviços, ferramentas e produtos de Open Source, da Microsoft ou não.
- **Nuvem flexível:** é fácil o suficiente para os usuários finais e desenvolvedores para usá-la com suas habilidades e conhecimentos existentes.
- **Gerenciamento unificado:** fornece recursos de gerenciamento e monitoramento ininterruptos.

Todos os serviços e recursos mencionados aqui são importantes para a implementação bem-sucedida de DevOps. A Figura 13.24 mostra os utilitários e as ferramentas de Open Source que podem ser usados para diferentes fases do gerenciamento do ciclo de vida da aplicação e do DevOps em geral:

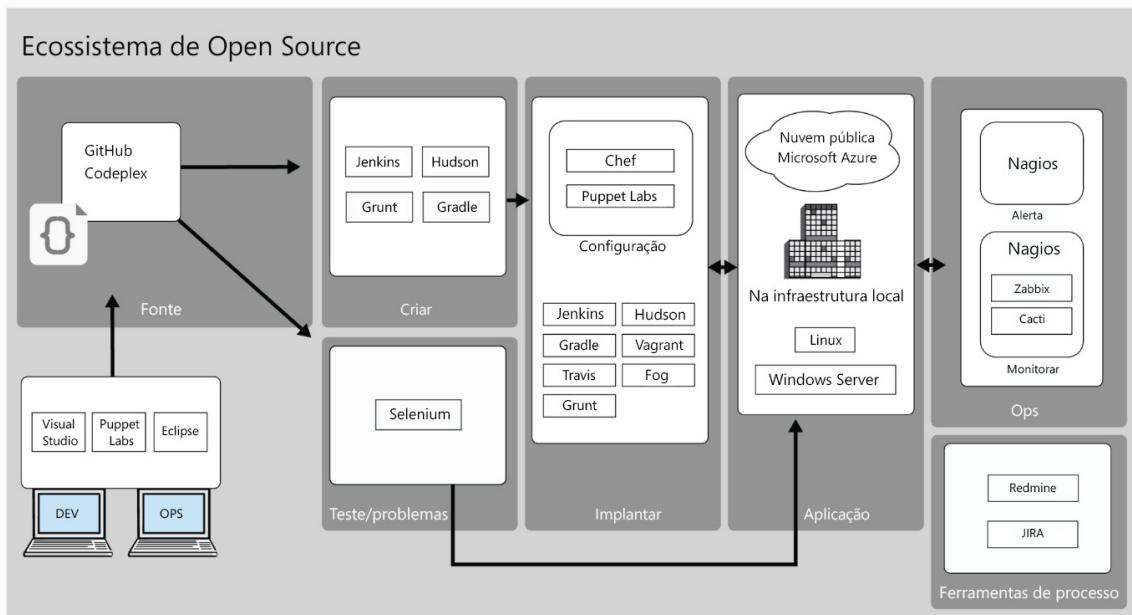


Figura 13.24: utilitários e ferramentas de Open Source

A Figura 13.24 mostra os utilitários e as ferramentas da Microsoft que podem ser usados para diferentes fases do gerenciamento do ciclo de vida da aplicação e do DevOps em geral. Novamente, essa é apenas uma pequena representação de todos os utilitários e ferramentas. Há muito mais opções disponíveis, como a seguir:

- Orquestração de build do Azure DevOps para construção de um pipeline de build
- Microsoft Test Manager e Pester para testes
- Modelos DSC, PowerShell e ARM para implantação ou gerenciamento de configuração
- Log Analytics, Application Insights e **System Center Operations Manager (SCOM)** para alertas e monitoramento
- Azure DevOps e System Center Service Manager para gerenciamento de processos:

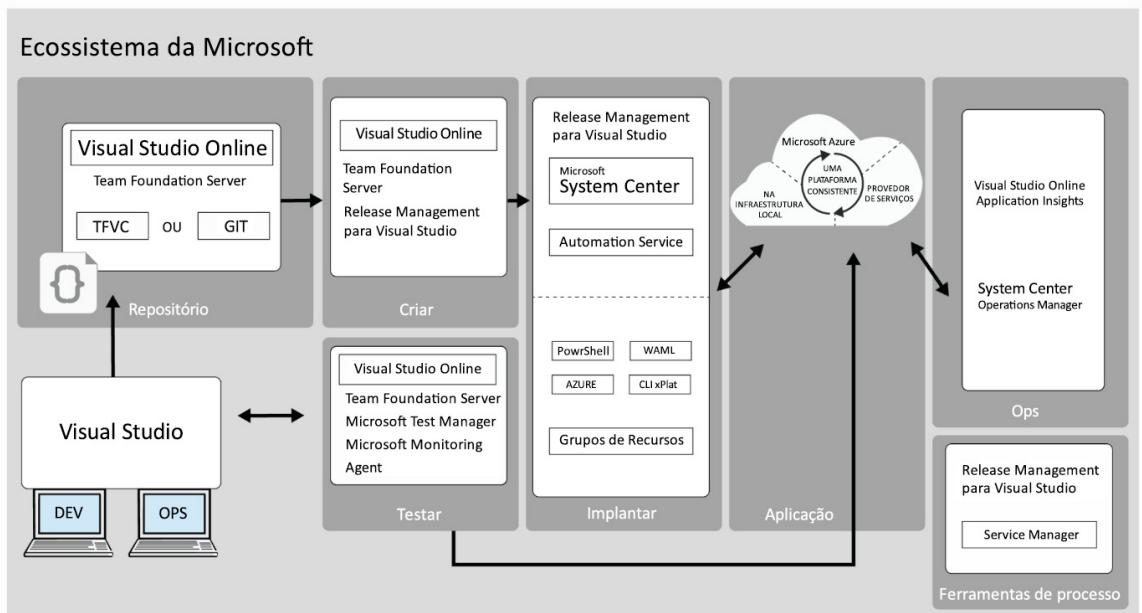


Figura 13.25: utilitários e ferramentas da Microsoft

Há muitas ferramentas disponíveis para cada uma das práticas de DevOps, e, nesta seção, você viu algumas das ferramentas e a forma de configurá-las.

## Resumo

O DevOps está ganhando muita tração e impulso na indústria. A maioria das organizações percebeu seus benefícios e está procurando implementá-lo. Isso está acontecendo enquanto a maioria delas está migrando para a nuvem. O Azure, como uma plataforma de nuvem, fornece serviços de DevOps sofisticados e maduros, facilitando a implementação de DevOps para essas organizações.

Neste capítulo, discutimos o DevOps e suas práticas fundamentais, como gerenciamento de configuração, integração contínua, entrega contínua e implantação. Também abordamos diferentes soluções de nuvem baseadas em PaaS, uma IaaS de máquina virtual e uma IaaS de contêiner, juntamente com seus respectivos recursos do Azure, os pipelines de build e lançamento.

O gerenciamento de configuração também foi explicado neste capítulo, juntamente com os serviços do DSC da Automação do Azure e o uso de servidores de pull para configurar máquinas virtuais automaticamente. Por fim, abordamos a abertura e a flexibilidade do Azure em relação à opção de linguagens, ferramentas e sistemas operacionais.

No próximo capítulo, vamos ver os detalhes do Kubernetes e seus componentes e interações, além de considerações de design e implantação de aplicações no Kubernetes.



# 14

## Arquitetar soluções de Kubernetes do Azure

Os contêineres são um dos componentes de infraestrutura mais comentados da última década. Os contêineres não são uma nova tecnologia. Eles existem há muito tempo. Eles têm sido prevalentes no mundo Linux há mais de duas décadas. Os contêineres não eram bem conhecidos na comunidade de desenvolvedores devido à sua complexidade e ao fato de que não havia muita documentação sobre eles. No entanto, em torno do início desta década, em 2013, foi criada uma empresa conhecida como Docker que mudou a percepção e a adoção de contêineres no mundo de desenvolvedores.

O Docker escreveu um wrapper de API robusto sobre os contêineres do Linux (LXC) existentes e facilitou a criação, o gerenciamento e o destriuição de contêineres na interface de linha de comando por parte dos desenvolvedores. Ao organizar aplicações em contêineres, o número de contêineres pode aumentar drasticamente com o tempo, e podemos chegar a um ponto em que precisamos gerenciar centenas ou até mesmo milhares de contêineres. É aqui que os orquestradores de contêiner desempenham uma função importante, e o Kubernetes é um deles. Usando o Kubernetes, podemos automatizar a implantação, a escalabilidade, a rede e o gerenciamento de contêineres.

Neste capítulo, vamos ver:

- Os conceitos introdutórios de contêineres
- Os conceitos do Kubernetes
- Os elementos importantes que fazem o Kubernetes funcionar
- Arquitetar soluções usando o Serviço de Kubernetes do Azure

Agora que você sabe para que o Kubernetes é usado, vamos começar do zero e discutir o que são contêineres, como eles são orquestrados usando o Kubernetes e muito mais.

## Introdução aos contêineres

Os contêineres são chamados de sistemas virtualização no nível do sistema operacional. Eles são hospedados em um sistema operacional em execução em um servidor físico ou em um servidor virtual. A natureza da implementação depende do sistema operacional do host. Por exemplo, os contêineres do Linux são inspirados por cgroups. Por outro lado, os contêineres do Windows são máquinas virtuais leves com uma pegada pequena.

Os contêineres realmente funcionam entre plataformas. As aplicações em contêineres podem ser executadas em qualquer plataforma, como Linux, Windows ou Mac, uniformemente e sem necessidade de alterações, o que as torna altamente portáteis. Isso os torna uma tecnologia perfeita para as organizações adotarem, pois são independentes de plataformas.

Além disso, os contêineres podem ser executados em qualquer ambiente de nuvem ou na infraestrutura local, sem a necessidade de alterações. Isso significa que as organizações também não estarão vinculadas a um único provedor de nuvem se implementarem contêineres como sua plataforma de hospedagem na nuvem. Elas podem migrar seu ambiente da infraestrutura local e fazer o lift and shift para a nuvem.

Os contêineres fornecem todos os benefícios que normalmente estão disponíveis com máquinas virtuais. Eles têm seus próprios endereços IP, nomes DNS, identidades, pilhas de rede, sistemas de arquivos e outros componentes que dão aos usuários a impressão de estarem usando um novo ambiente de sistema operacional intocado. Nos bastidores, o tempo de execução do Docker virtualiza vários componentes de nível do kernel do sistema operacional para dar essa impressão.

Todos esses benefícios fornecem grandes benefícios para as organizações que adotam a tecnologia de contêineres, e o Docker é um dos precursores a esse respeito. Há outras opções de tempo de execução do contêiner disponíveis, como CoreOS Rkt (pronunciado como "Rocket", fora de produção), Mesos Containerizer e contêineres LXC. As organizações podem adotar a tecnologia com a qual se sintam à vontade.

Os contêineres não estavam disponíveis anteriormente no mundo Windows, sendo disponibilizados somente para Windows 10 e Windows Server 2016. No entanto, os contêineres agora são cidadãos de primeira classe no mundo Windows.

Como mencionado na introdução, os contêineres devem ser monitorados, regidos e gerenciados apropriadamente, assim como qualquer outro componente de infraestrutura em um ecossistema. É necessário implantar um orquestrador, como o Kubernetes, que pode ajudar você a fazer isso facilmente. Na próxima seção, você conhecerá os fundamentos do Kubernetes, incluindo quais são suas vantagens.

## Fundamentos do Kubernetes

Muitas organizações ainda perguntam: "*Precisamos do Kubernetes ou, na verdade, de qualquer orquestrador de contêineres?*" Quando consideramos o gerenciamento de contêineres em grande escala, precisamos considerar vários pontos, como escalabilidade, balanceamento de carga, gerenciamento de ciclo de vida, entrega contínua, registro em log, monitoramento e muito mais.

Você pode perguntar: "*Os contêineres não devem fazer tudo isso?*" A resposta é que os contêineres são apenas uma peça de baixo nível do quebra-cabeças. Os benefícios reais são obtidos por meio das ferramentas que agem sobre os contêineres. No final do dia, precisamos de algo para ajudar com a orquestração.

Kubernetes é uma palavra grega, κυβερνήτης, que significa "timoneiro" ou "capitão do navio". Mantendo o tema marítimo dos contêineres do Docker, o Kubernetes é o capitão do navio. O Kubernetes muitas vezes é chamado de K8s, onde 8 representa as oito letras entre "K" e "s" na palavra "Kubernetes".

Como mencionado anteriormente, os contêineres são mais ágeis do que as máquinas virtuais. Eles podem ser criados em segundos e destruídos também rapidamente. Eles têm um ciclo de vida semelhante às máquinas virtuais. No entanto, eles precisam ser monitorados, regidos e gerenciados ativamente em um ambiente.

É possível gerenciá-los usando seu conjunto de ferramentas existente. Mesmo assim, ferramentas especializadas, como o Kubernetes, podem oferecer benefícios valiosos:

- O Kubernetes tem uma natureza de recuperação automática. Quando um pod (considerado como "contêiner" por enquanto) ficar inativo em um ambiente do Kubernetes, o Kubernetes garantirá que um novo pod seja criado em outro lugar no mesmo nó ou em outro nó, para responder a solicitações em nome da aplicação.
- O Kubernetes também facilita o processo de atualização de uma aplicação. Ele fornece recursos prontos para uso para ajudá-lo a executar vários tipos de atualizações com a configuração original.
- Ele ajuda a habilitar implantações azul-verde. Nesse tipo de implantação, o Kubernetes implantará a nova versão da aplicação juntamente com a anterior e, quando for confirmado que a nova aplicação funcionará conforme o esperado, uma alternância de DNS será criado para alternar para a nova versão da aplicação. A implantação da aplicação antiga pode continuar existindo para fins de reversão.
- O Kubernetes também ajuda a implementar uma estratégia de implantação de atualização de lançamento. Aqui, o Kubernetes implantará a nova versão da aplicação em um servidor por vez e desativará a antiga implantação em um servidor por vez. Ele continuará essa atividade até que não haja mais servidores na implantação antiga.
- O Kubernetes pode ser implantado em um datacenter na infraestrutura local ou na nuvem usando o paradigma de **Infraestrutura como Serviço (IaaS)**. Isso significa que os desenvolvedores criam um grupo de máquinas virtuais primeiro e implantam o Kubernetes sobre ele. Há também a abordagem alternativa de usar o Kubernetes como uma oferta de **Plataforma como Serviço (PaaS)**. O Azure fornece um serviço PaaS conhecido como **Serviço de Kubernetes do Azure (AKS)**, que fornece um ambiente do Kubernetes pronto para uso pelos desenvolvedores.

Quando se trata da implantação, o Kubernetes pode ser implantado de duas maneiras:

- **Clusters não gerenciados:** podem ser criados com a instalação do Kubernetes e quaisquer outros pacotes relevantes em uma máquina bare metal ou uma máquina virtual. Em um cluster não gerenciado, haverá nós mestre e de trabalho, anteriormente conhecidos como minions. Os nós mestre e de trabalho funcionam lado a lado para orquestrar os contêineres. Se você está se perguntando como isso é feito, exploraremos a arquitetura completa do Kubernetes mais adiante neste capítulo. Agora, apenas saiba que existem nós mestre e de trabalho.

- **Clusters gerenciados:** normalmente são oferecidos pelo provedor de nuvem, que gerencia a infraestrutura para você. No Azure, esse serviço é chamado de AKS. O Azure fornecerá suporte ativo sobre correções e gerenciamento da infraestrutura. Com a IaaS, as organizações precisam garantir a disponibilidade e a escalabilidade dos nós e da infraestrutura por conta própria. No caso do AKS, o componente mestre não será visível, pois é gerenciado pelo Azure. No entanto, os nós de trabalho (minions) estarão visíveis e serão implantados em um grupo de recursos separado, para que você possa acessar os nós, se necessário.

Alguns dos principais benefícios do uso do AKS em clusters não gerenciados são:

- Se você estiver usando clusters não gerenciados, precisará trabalhar para tornar a solução altamente disponível e escalável. Além disso, você precisa ter um gerenciamento de atualização adequado para instalar atualizações e patches. Por outro lado, no AKS, o Azure gerencia isso completamente, permitindo que os desenvolvedores economizem tempo e sejam mais produtivos.
- Integração nativa com outros serviços, como o Registro de Contêiner do Azure para armazenar suas imagens de contêiner com segurança, o Azure DevOps para integrar pipelines de CI/CD, o Azure Monitor para registro em log e o Azure Active Directory para segurança.
- Escalabilidade e velocidade de inicialização mais rápida.
- Suporte para conjuntos de escala de máquinas virtuais.

Embora não haja diferença em termos da funcionalidade básica dessas duas implantações, a forma de implantação de IaaS fornece a flexibilidade para adicionar novos plugins e configuração imediatamente que podem levar algum tempo até que a equipe do Azure disponibilize com o AKS. Além disso, as versões mais recentes do Kubernetes estão disponíveis no AKS muito rapidamente, sem muito atraso.

Abordamos os fundamentos do Kubernetes. Neste momento, você pode estar se perguntando como o Kubernetes faz tudo isso. Na próxima seção, vamos analisar os componentes do Kubernetes e como eles funcionam lado a lado.

## Arquitetura do Kubernetes

A primeira etapa para entender o Kubernetes é entender sua arquitetura. Vamos abordar os detalhes de cada componente na próxima seção, mas obter uma visão geral de alto nível da arquitetura ajudará você a entender a interação entre os componentes.

## Clusters do Kubernetes

O Kubernetes precisa de nós físicos ou virtuais para instalar dois tipos de componentes:

- Componentes do plano de controle do Kubernetes ou componentes mestre
- Nós de trabalho do Kubernetes (minions) ou componentes não mestre

A Figura 14.1 é um diagrama que oferece uma visão geral de alto nível da arquitetura do Kubernetes. Vamos abordar os componentes em mais detalhes mais adiante:

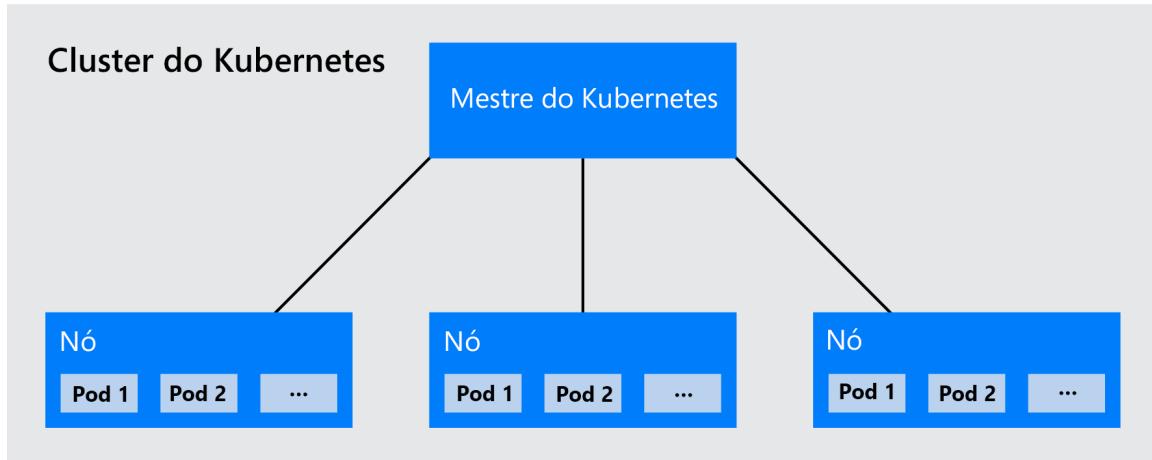


Figura 14.1: visão geral do cluster do Kubernetes

Os componentes do plano de controle são responsáveis por gerenciar e reger o ambiente do Kubernetes e os minions do Kubernetes.

Todos nós juntos (o mestre, bem como os minions) formam o cluster. Em outras palavras, um cluster é uma coleção de nós. Eles são virtuais ou físicos, conectados uns aos outros, e alcançáveis usando a pilha de rede TCP. O mundo exterior não terá ideia do tamanho ou da capacidade do seu cluster ou até mesmo dos nomes dos nós de trabalho. A única coisa que os nós têm conhecimento é o endereço do servidor de API por meio do qual eles interagem com o cluster. Para eles, o cluster é um computador grande que executa suas aplicações.

É o Kubernetes que decide internamente uma estratégia apropriada, usando controladores, para escolher um nó válido e íntegro que pode executar facilmente a aplicação.

Os componentes do plano de controle podem ser instalados em uma configuração de alta disponibilidade. Até agora, abordamos clusters e como eles funcionam. Na próxima seção, vamos ver os componentes de um cluster.

## Componentes do Kubernetes

Os componentes do Kubernetes são divididos em duas categorias: componentes mestre e componentes de nó. Os componentes mestre também são conhecidos como o plano de controle do cluster. O plano de controle é responsável por gerenciar os nós de trabalho e os pods no cluster. A autoridade de tomada de decisão de um cluster é o plano de controle e também é responsável pela detecção e pelas respostas relacionadas a eventos de cluster. A Figura 14.2 descreve a arquitetura completa de um cluster do Kubernetes:

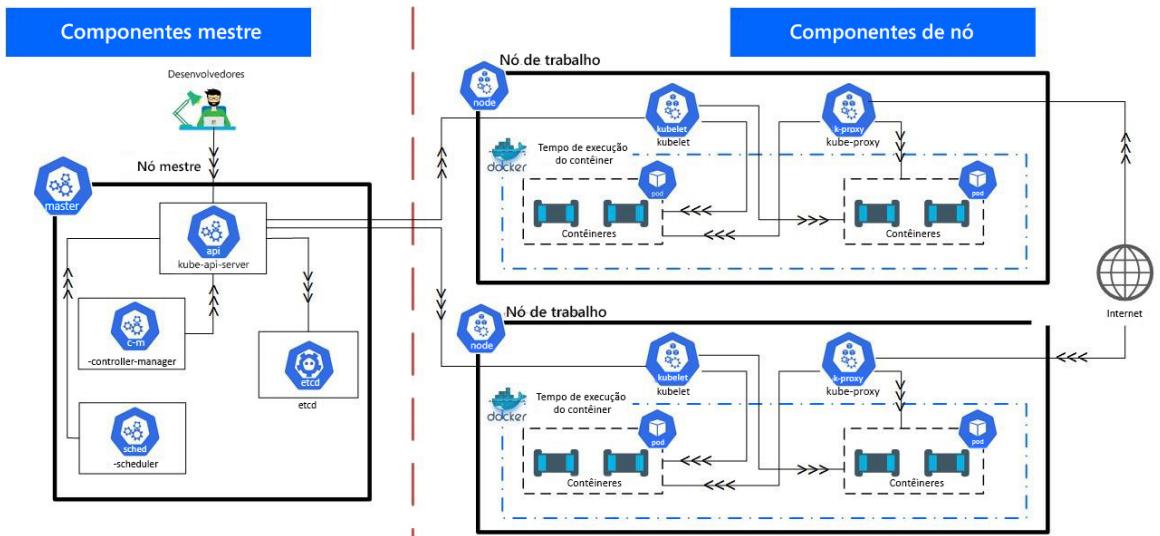


Figura 14.2: arquitetura do Kubernetes

Você precisa entender cada um desses componentes para administrar um cluster corretamente. Vamos avançar e abordar quais são os componentes mestre:

- **Servidor de API:** o servidor de API é, sem dúvida, o cérebro do Kubernetes. É o componente central que permite todas as atividades no Kubernetes. Todas as solicitações de clientes, com poucas exceções, terminam no servidor de API, que decide o fluxo para a solicitação. Ele é o único responsável pela interação com o servidor etcd.
- **etcd:** é o armazenamento de dados para o Kubernetes. Somente o servidor de API tem permissão para se comunicar com o etcd, e o servidor de API pode executar atividades de **criação, leitura, atualização e exclusão (CRUD)** no etcd. Quando uma solicitação termina no servidor de API, após a validação, o servidor de API pode executar quaisquer operações CRUD, dependendo da solicitação do etcd. O etcd é um armazenamento de dados distribuído e altamente disponível. Pode haver várias instalações do etcd, cada uma com uma cópia dos dados, e qualquer uma delas pode atender às solicitações do servidor de API. Na Figura 14.3, você pode ver que há várias instâncias em execução no plano de controle para fornecer alta disponibilidade:

## kubeadm HA topology - stacked etcd

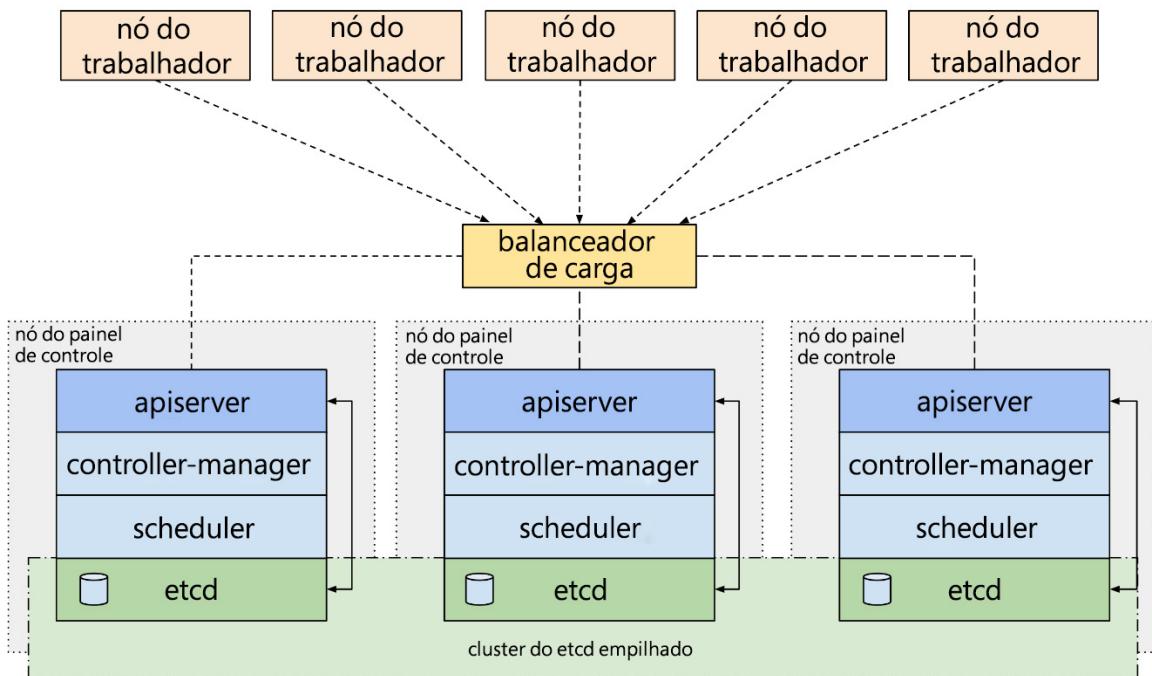


Figura 14.3: tornando o plano de controle altamente disponível

- **Gerenciador do controlador:** é o elemento fundamental do Kubernetes. Enquanto o servidor de API recebe as solicitações, o trabalho real no Kubernetes é feito pelo gerenciador do controlador. O gerenciador do controlador como o nome sugere, é o gerenciador dos controladores. Há vários controladores em um nó mestre do Kubernetes, e cada um é responsável por gerenciar um único controlador.

A principal responsabilidade de um controlador é gerenciar um único recurso em um ambiente do Kubernetes. Por exemplo, há um gerenciador do controlador de replicação para gerenciar recursos de controlador de replicação e um controlador ReplicaSet para gerenciar ReplicaSets em um ambiente do Kubernetes. O controlador fica atento ao servidor de API e, quando recebe uma solicitação de um recurso gerenciado por ele, o controlador executa seu trabalho.

Uma das principais responsabilidades dos controladores é continuar em execução em um loop e garantir que o Kubernetes esteja no estado desejado. Se houver algum desvio do estado desejado, os controladores devem trazê-lo de volta para o estado desejado. Um controlador de implantação observa todos os novos recursos de implantação criados pelo servidor de API. Se um novo recurso de implantação for encontrado, o controlador da implantação criará um novo recurso ReplicaSet e garantirá que o ReplicaSet esteja sempre no estado desejado. Um controlador de replicação continua em execução em um loop e verifica se o número real de pods no ambiente corresponde ao número desejado de pods. Se um pod ficar inativo por qualquer motivo, o controlador de replicação descobrirá que a contagem real foi desativada por um e programará um novo pod no mesmo ou em outro nó.

- **Agendador:** o trabalho de um agendador é agendar os pods nos nós minion do Kubernetes. Ele não é responsável pela criação de pods. É puramente responsável por atribuir pods aos nós minion do Kubernetes. Ele faz isso levando em consideração o estado atual dos nós, o quanto ocupados eles estão, seus recursos disponíveis e também a definição do pod. Um pod pode ter uma preferência em relação a um nó específico, e o agendador manterá essas solicitações em consideração durante o agendamento de pods para nós.

Agora, vamos explorar os componentes de nós que são implantados em cada um dos nós de trabalho no cluster:

- **Kubelet:** enquanto o servidor de API, o agendador, os controladores e o etcd são implantados em nós mestre, os kubelets são implantados em nós minion. Eles atuam como agentes dos componentes mestre do Kubernetes e são responsáveis por gerenciar pods localmente nos nós. Há um kubelet em cada nó. Um kubelet recebe comandos dos componentes mestre e também fornece informações de integridade, monitoramento e atualização sobre nós e pods para os componentes mestre, como o servidor de API e o gerenciador do controlador. Eles são o canal de comunicação administrativa entre os nós mestre e minion.
- **kube-proxy:** o kube-proxy, assim como o kubelets, é implantado em nós minion. Ele é responsável pelo monitoramento de pods e serviços, bem como pela atualização das regras de firewall netfilter e iptables locais com qualquer alteração na disponibilidade de pods e serviços. Isso garante que as informações de roteamento nos nós sejam atualizadas à medida que novos pods e serviços são criados ou que pods e serviços existentes sejam excluídos.

- **Tempo de execução do contêiner:** há muitos provedores e fornecedores de contêineres no ecossistema atualmente. O Docker é o mais famoso de todos eles, embora outros também estejam ganhando popularidade. É por isso que, em nossa arquitetura, denotamos o tempo de execução do contêiner com o logotipo do Docker. Ele é um orquestrador de contêineres genérico. Ele não pode ser estreitamente associado a um único fornecedor de contêiner, como o Docker. Deve ser possível usar qualquer tempo de execução de contêiner nos nós minion para gerenciar o ciclo de vida dos contêineres.

Para executar contêineres em pods, um padrão específico da indústria conhecido como **interface de tempo de execução do contêiner (CRI)** foi desenvolvido e é usado por todas as empresas líderes. O padrão fornece regras que devem ser seguidas para alcançar a interoperabilidade com orquestradores, como o Kubernetes. Os kubelets não sabem quais binários de contêiner são instalados nos nós. Eles podem ser binários do Docker ou quaisquer outros binários.

Como esses tempos de execução de contêiner são desenvolvidos com um padrão comum específico da indústria, independentemente do tempo de execução que você está usando, os kubelets poderão se comunicar com o tempo de execução do contêiner. Isso dissocia o gerenciamento de contêineres do gerenciamento de cluster do Kubernetes. As responsabilidades do tempo de execução do contêiner incluem a criação de contêineres, o gerenciamento da pilha de rede dos contêineres e o gerenciamento da rede de pontes. Como o gerenciamento de contêineres é separado do gerenciamento de cluster, o Kubernetes não interferirá nas responsabilidades do tempo de execução do contêiner.

Os componentes abordados são aplicáveis a clusters do AKS gerenciados e não gerenciados. No entanto, os componentes mestre não são expostos ao usuário final, pois o Azure gerencia tudo isso no caso do AKS. Mais adiante neste capítulo, abordaremos a arquitetura do AKS. Você aprenderá sobre clusters não gerenciados e entenderá as diferenças entre esses sistemas de forma mais clara.

Em seguida, você conhecerá alguns dos recursos mais importantes do Kubernetes, também conhecidos como primitivos, um conhecimento que é aplicável a clusters não gerenciados e do AKS.

## Primitivos do Kubernetes

Você aprendeu que o Kubernetes é um sistema de orquestração usado para implantar e gerenciar contêineres. O Kubernetes define um conjunto de blocos de construção, que também são conhecidos como primitivos. Esses primitivos em conjunto podem nos ajudar a implantar, manter e escalar aplicações em contêineres. Vamos ver cada um dos primitivos e entender suas funções.

## Pod

Pods são a unidade mais básica da implantação no Kubernetes. A pergunta imediata que surge em uma mente curiosa é: como um pod difere-se de um contêiner? Os pods são wrappers sobre contêineres. Em outras palavras, os contêineres estão contidos em pods. Pode haver vários contêineres dentro de um pod. No entanto, a prática recomendada é ter um relacionamento individual de um único contêiner. Isso não significa que não podemos ter mais de um contêiner em um pod. Também é possível ter vários contêineres em um pod, contanto que haja um contêiner principal e os restantes sejam contêineres auxiliares. Também há padrões, como padrões de sidecar, que podem ser implementados com pods de vários contêineres.

Cada pod tem seu próprio endereço IP e pilha de rede. Todos os contêineres compartilham a pilha e a interface de rede. Todos os contêineres dentro de um pod podem ser acessados localmente usando o nome de host.

Uma definição simples de pod no formato YAML é mostrada nas linhas de código a seguir:

```
---
apiVersion: v1
kind: Pod
metadata:
  name: tappdeployment
  labels:
    appname: tapp
    ostype: linux
spec:
  containers:
    - name: mynewcontainer
      image: "tacracr.azurecr.io/tapp:latest"
      ports:
        - containerPort: 80
          protocol: TCP
          name: http
```

A definição de pod mostrada tem um nome e define alguns rótulos, que podem ser usados pelo recurso de serviço para expor a outros pods, nós e recursos personalizados externos. Ele também define um único contêiner com base em uma imagem personalizada armazenada no Registro de Contêiner do Azure e abre a porta **80** para o contêiner.

## Serviços

O Kubernetes permite a criação de pods com várias instâncias. Esses pods devem ser acessíveis em qualquer Pod ou nó dentro de um cluster. É possível usar o endereço IP de um pod diretamente e acessar o pod. No entanto, isso está longe de ser ideal. Os pods são efêmeros e poderão obter um novo endereço IP se o pod anterior ficar inativo. Nesses casos, a aplicação falhará facilmente. O Kubernetes fornece serviços, que dissociam instâncias de pod de seus clientes. Os pods podem ser criados e removidos, mas o endereço IP de um Serviço de Kubernetes permanece constante e estável. Os clientes podem se conectar ao endereço IP do serviço, que tem um ponto de extremidade para cada pod para o qual pode enviar solicitações. Se houver várias instâncias de pod, cada um de seus endereços IP estará disponível para o serviço como um objeto de ponto de extremidade. Quando um pod fica inativo, os pontos de extremidade são atualizados para refletir as instâncias atuais do pod, juntamente com seus endereços IP.

Os serviços são altamente dissociados de pods. O principal objetivo dos serviços é enfileirar para pods que têm rótulos em suas definições do seletor de serviço. Um serviço define seletores de rótulos e, com base neles, os endereços IP de pod são adicionados ao recurso de serviço. Pods e serviços podem ser gerenciados de forma independente uns dos outros.

Um serviço fornece vários tipos de esquemas de endereço IP. Existem quatro tipos de serviços: ClusterIP, NodePort, LoadBalancer e controlador de entrada usando o Gateway de Aplicativo.

O esquema mais fundamental é conhecido como ClusterIP, e é um endereço IP interno que pode ser acessado somente no cluster. O esquema ClusterIP é mostrado na Figura 14.4:

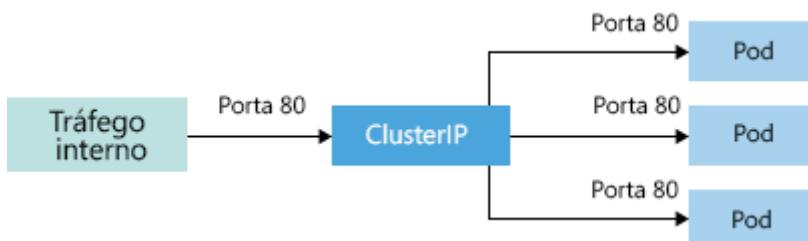


Figura 14.4: o funcionamento do ClusterIP

O ClusterIP também permite a criação de NodePort, com o qual receberá um ClusterIP. No entanto, ele também pode abrir uma porta em cada um dos nós dentro de um cluster. Os pods podem ser acessados usando endereços de ClusterIP, bem como usando uma combinação de IP do nó e porta do nó:

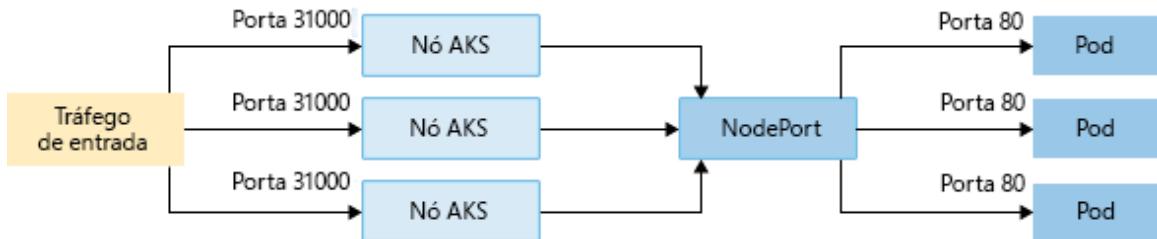


Figura 14.5: o funcionamento do NodePort

Os serviços podem se referir não apenas a pods, como também a pontos de extremidade externos. Por fim, os serviços também permitem a criação de um serviço baseado em平衡ador de carga que é capaz de receber solicitações externamente e redirecioná-las para uma instância de pod usando ClusterIP e NodePort internamente:

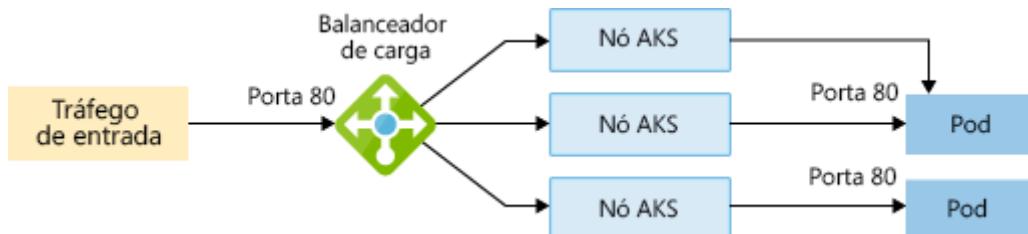


Figura 14.6: o funcionamento do balanceador de carga

Há um tipo final de serviço conhecido como controlador de entrada, que fornece funcionalidades avançadas, como roteamento baseado em URL, conforme mostrado na Figura 14.7:

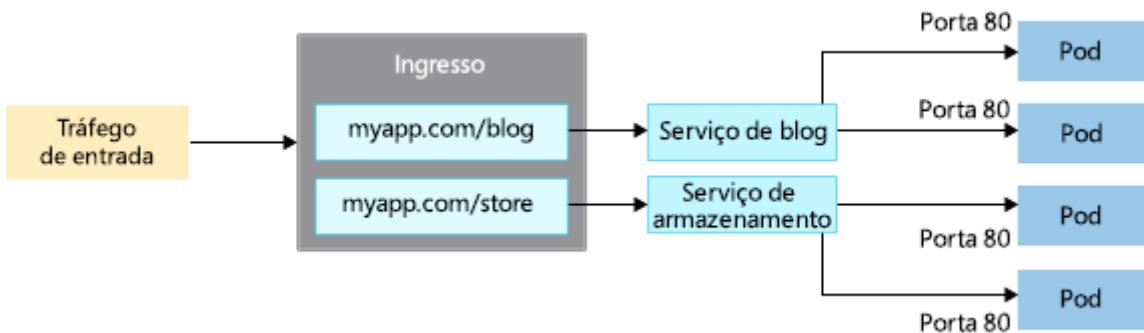


Figura 14.7: o funcionamento do controlador de entrada

Uma definição de serviço no formato YAML é mostrada aqui:

```
apiVersion: v1
kind: Service
metadata:
  name: tappservice
  labels:
    appname: tapp
    ostype: linux
spec:
  type: LoadBalancer
  selector:
    appname: myappnew
  ports:
  - name: http
    port: 8080
    targetPort: 80
    protocol: TCP
```

Essa definição de serviço cria um serviço baseado em balanceador de carga usando seletores de rótulos.

## Implantações

As implantações do Kubernetes são recursos de nível mais alto em comparação a ReplicaSets e pods. As implantações fornecem funcionalidades relacionadas à atualização e ao lançamento de uma aplicação. Os recursos de implantação criam um ReplicaSet, e o ReplicaSet gerencia o pod. É importante entender a necessidade de recursos de implantação quando ReplicaSets já existem.

As implantações desempenham uma função significativa na atualização de aplicações. Se uma aplicação já está em produção e uma nova versão da aplicação precisa ser implantada, há algumas opções para você:

1. Excluir pods existentes e criar novos pods: neste método, há tempo de inatividade para a aplicação, portanto, só deverá ser usado se o tempo de inatividade for aceitável. Haverá um risco de maior tempo de inatividade se a implantação conter bugs, e você precisará reverter para uma versão anterior.

2. Implantação azul-verde: neste método, os pods existentes continuam em execução, e um novo conjunto de pods é criado com a nova versão da aplicação. Os novos pods não são acessíveis externamente. Depois que os testes forem concluídos com êxito, o Kubernetes começará a apontar para o novo conjunto de pods. Os pods antigos podem permanecer como estão ou podem ser excluídos posteriormente.
3. Atualizações sem interrupção: neste método, pods existentes são excluídos individualmente, enquanto novos pods para a nova versão da aplicação são criados individualmente. Os novos pods são implantados incrementalmente, enquanto os pods antigos são gradativamente reduzidos, até atingirem uma contagem zero.

Todas essas abordagens teriam que ser executadas manualmente sem um recurso de implantação. Um recurso de implantação automatiza todo o processo de lançamento e atualização. Ele também poderá ajudar a reverter automaticamente para uma versão anterior se houver problemas com a implantação atual.

Uma definição de implantação é mostrada na listagem de código a seguir:

```
---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tappdeployment
  labels:
    appname: tapp
    ostype: linux
spec:
  replicas: 3
  selector:
    matchLabels:
      appname: myappnew
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
```

```
metadata:  
  name: mypod  
  labels:  
    appname: myappnew  
spec:  
  containers:  
    - name: mynewcontainer  
      image: "tacracr.azurecr.io/tapp:latest"  
      ports:  
        - containerPort: 80  
          protocol: TCP  
          name: http
```

É importante observar que uma implantação tem uma propriedade **strategy**, que determina se a estratégia **recreate** ou **RollingUpdate** é usada. A estratégia **recreate** excluirá todos os pods existentes e criará outros novos. Ela também contém detalhes de configuração relacionados a **RollingUpdate**, fornecendo o número máximo de pods que podem ser criados e destruídos em uma única execução.

## Controlador de replicação e ReplicaSet

O recurso de controlador de replicação do Kubernetes garante que um número desejado e especificado de instâncias de pod estejam sempre em execução em um cluster. Todos os desvios do estado desejado são observados pelo controlador de replicação e criam novas instâncias de pod para atender ao estado desejado.

ReplicaSets são a nova versão do controlador de replicação. ReplicaSets fornecem a mesma funcionalidade que os controladores de replicação, com algumas funcionalidades avançadas. A principal delas é a capacidade avançada de definir os seletores associados aos pods. Com ReplicaSets, é possível definir as expressões dinâmicas que faltavam com os controladores de replicação.

É recomendável usar ReplicaSets em vez de controladores de replicação.

A próxima listagem de código mostra um exemplo de como definir um recurso **ReplicaSet**:

```
---  
apiVersion: apps/v1  
kind: ReplicaSet  
metadata:
```

```
name: tappdeployment
labels:
  appname: tapp
  ostype: linux
spec:
  replicas: 3
  selector:
    matchLabels:
      appname: myappnew
  template:
    metadata:
      name: mypod
    labels:
      appname: myappnew
  spec:
    containers:
      - name: mynewcontainer
        image: "tacracr.azurecr.io/tapp:latest"
        ports:
          - containerPort: 80
            protocol: TCP
            name: http
```

É importante observar que ReplicaSets têm uma propriedade **replicas**, que determina a contagem de instâncias de pod, uma propriedade **selector**, que define os pods que devem ser gerenciados por ReplicaSet, e, por fim, a propriedade **template**, que define o pod em si.

## ConfigMaps e segredos

O Kubernetes fornece dois recursos importantes para armazenar dados de configuração. ConfigMaps são usados para armazenar dados de configuração geral que não são sensíveis à segurança. Os dados de configuração genéricos da aplicação, como nomes de pastas, nomes de volume e nomes DNS, podem ser armazenados em ConfigMaps. Por outro lado, dados confidenciais, como credenciais, certificados e segredos, devem ser armazenados nos recursos de segredos. Esses dados de segredos são criptografados e armazenados no armazenamento de dados etcd do Kubernetes.

Os dados de ConfigMaps e segredos podem ser disponibilizados como variáveis ou volumes do ambiente dentro de pods.

A definição do pod que deseja consumir esses recursos deve incluir uma referência a eles. Agora, abordamos os primitivos do Kubernetes e as funções de cada um dos blocos de construção. Em seguida, você aprenderá sobre a arquitetura do AKS.

## Arquitetura do AKS

Na seção anterior, abordamos a arquitetura de um cluster não gerenciado.

Agora, exploraremos a arquitetura do AKS. Ao ler esta seção, você poderá indicar as principais diferenças entre a arquitetura de clusters não gerenciados e gerenciados (o AKS, neste caso).

Quando uma instância do AKS é criada, somente os nós de trabalhador são criados. Os componentes mestre são gerenciados pelo Azure. Os componentes mestre são o servidor de API, o agendador, etcd e o gerenciador do controlador, que abordamos anteriormente. Os kubelets e o kube-proxy são implantados nos nós de trabalhador. A comunicação entre os nós e os componentes mestre ocorre usando kubelets, que atuam como agentes dos clusters do Kubernetes para o nó:

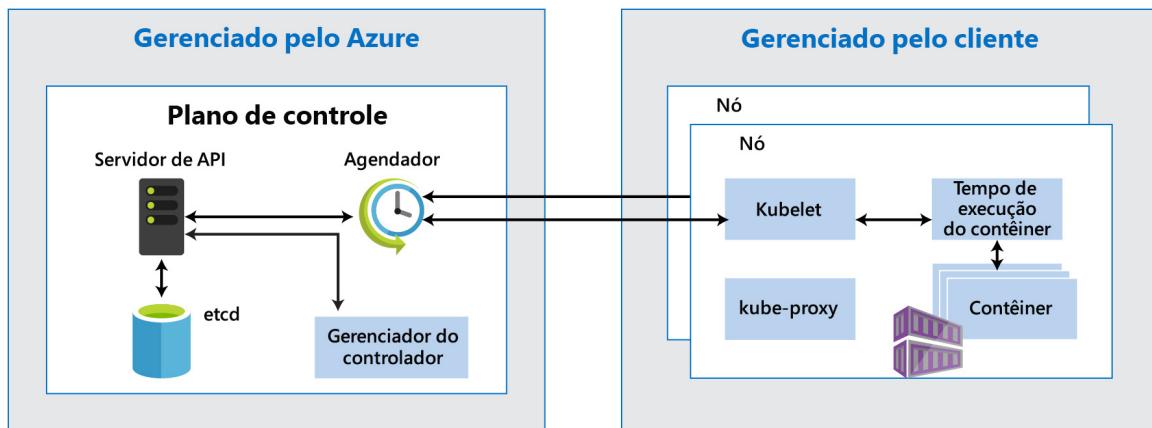


Figura 14.8: arquitetura do AKS

Quando um usuário solicita uma instância de pod, a solicitação do usuário chega no servidor de API. O servidor de API verifica e valida os detalhes da solicitação e armazena no etcd (o armazenamento de dados para o cluster) e também cria o recurso de implantação (se a solicitação do pod estiver envolvida em um recurso de implantação). O controlador da implantação mantém sob observação a criação de todos os novos recursos de implantação. Se ele vir um, criará um recurso ReplicaSet com base na definição fornecida na solicitação do usuário.

O controlador de ReplicaSet mantém sob observação a criação de novos recursos ReplicaSet e, ao ver um recurso sendo criado, solicita que o agendador agende os pods. O agendador tem seu próprio procedimento e regras para encontrar um nó apropriado para hospedar os pods. O agendador informa o kubelet do nó e o kubelet, em seguida, obtém a definição para o pod e cria os pods usando o tempo de execução do contêiner instalado nos nós. Por fim, o pod cria os contêineres dentro de sua definição.

O kube-proxy ajuda a manter a lista de endereços IP de informações de pod e serviço em nós locais, bem como atualizar as regras de firewall e roteamento locais. Para fazer uma recapitulação rápida do que abordamos até agora, começamos com a arquitetura do Kubernetes e, depois, avançamos para os primitivos e a arquitetura do AKS. Já que você entendeu bem os conceitos, vamos criar um cluster do AKS na próxima seção.

## Implantar um cluster do AKS

O AKS pode ser provisionado usando o portal do Azure, a **CLI do Azure (interface de linha de comando)**, cmdlets do Azure PowerShell, modelos do ARM, **SDKs (kits de desenvolvimento de software)** para linguagens compatíveis e até mesmo APIs REST do ARM do Azure.

O portal do Azure é a maneira mais simples de criar uma instância do AKS. No entanto, para habilitar o DevOps, é melhor criar uma instância do AKS usando modelos do ARM, a CLI ou o PowerShell.

### Criar um cluster do AKS

Vamos criar um grupo de recursos para implantar nosso cluster do AKS. Na CLI do Azure, use o comando **az group create**:

```
az group create -n AzureForArchitects -l southeastasia
```

Aqui, **-n** denota o nome do grupo de recursos e **-l** denota o local. Se a solicitação for bem-sucedida, você verá uma resposta semelhante a esta:

```
[root@rithin az group create -n AzureForArchitects -l southeastasia
{
  "id": "/subscriptions/.../resourceGroups/AzureForArchitects",
  "location": "southeastasia",
  "managedBy": null,
  "name": "AzureForArchitects",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": null
}
```

Figura 14.9: criação de um grupo de recursos

Agora que temos o grupo de recursos pronto, avançaremos e criaremos o cluster do AKS usando o comando `az aks create`. O comando a seguir criará um cluster chamado **AzureForArchitects-AKS** no grupo de recursos **AzureForArchitects** com uma contagem de nós de 2. O parâmetro `--generate-ssh-keys` permitirá a criação de pares de chaves RSA (**Rivest-Shamir-Adleman**), um criptosistema de chaves públicas:

```
az aks create --resource-group AzureForArchitects \
--name AzureForArchitects-AKS \
--node-count 2 \
--generate-ssh-keys
```

Se o comando for bem-sucedido, você poderá ver uma saída semelhante a esta:

```
[2] rithin az aks create --resource-group AzureForArchitects \
> --name AzureForArchitects-AKS \
> --node-count 2 \
> --generate-ssh-keys
{
  "aadProfile": null,
  "addonProfiles": null,
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "count": 2,
      "enableAutoScaling": null,
      "maxCount": null,
      "maxPods": 110,
      "minCount": null,
      "name": "nodepool1",
      "orchestratorVersion": "1.15.11",
      "osDiskSizeGb": 100,
      "osType": "Linux",
      "provisioningState": "Succeeded",
      "type": "AvailabilitySet",
      "vmSize": "Standard_DS2_v2",
      "vnetSubnetId": null
    }
  ],
  "apiServerAuthorizedIpRanges": null,
  "dnsPrefix": "AzureForAr-AzureForArchitec-1b2287",
  "enablePodSecurityPolicy": null,
  "enableRbac": true,
  "fqdn": "azureforar-azureforarchitec-1b2287-72aecee5.hcp.southeastasia.azmk8s.io",
}
```

Figura 14.10: criando o cluster

Ao passar pelo cluster, você verá um item de linha que diz "`nodeResourceGroup": "MC_AzureForArchitects_AzureForArchitects-AKS_southeastasia"`". Ao criar um cluster do AKS, um segundo recurso é criado automaticamente para armazenar os recursos do nó.

Nosso cluster é provisionado. Agora precisamos nos conectar ao cluster e interagir com ele. Para controlar o gerenciador de cluster do Kubernetes, usaremos o kubectl. Na próxima seção, veremos rapidamente o kubectl.

## Kubectl

O kubectl é o principal componente por meio do qual desenvolvedores e consultores de infraestrutura podem interagir com o AKS. O kubectl ajuda na criação de uma solicitação REST contendo o cabeçalho e o corpo HTTP e no envio dela para o servidor de API. O cabeçalho contém os detalhes de autenticação, como uma combinação de token ou nome de usuário/senha. O corpo contém a carga real no formato JSON.

O comando kubectl fornece detalhes de log avançados quando usado juntamente com a alternância detalhada. A alternância usa uma entrada inteira que pode variar de 0 a 9, que pode ser visualizada nos logs de detalhes.

## Conectar ao cluster

Para se conectar ao cluster localmente, precisamos instalar o kubectl. O Azure Cloud Shell já tem o kubectl instalado. Se você quiser se conectar localmente, use **az aks install-cli** para instalar o kubectl.

Para configurar o kubectl a fim de se conectar ao nosso cluster do Kubernetes, precisamos fazer o download das credenciais e configurar a CLI com elas. Isso pode ser feito usando o comando **az aks get-credentials**. Use o comando, conforme mostrado aqui:

```
az aks get-credentials \
--resource-group AzureForArchitects \
--name AzureForArchitects-AKS
```

Agora, precisamos verificar se estamos conectados ao cluster. Como mencionado anteriormente, usaremos o kubectl para nos comunicarmos com o cluster, e **kubectl get nodes** mostrará uma lista de nós no cluster. Durante a criação, definimos a contagem de nós como 2, de modo que a saída tenha dois nós. Além disso, precisamos garantir que o status do nó esteja **Pronto**. A saída deve ser semelhante à Figura 14.11:

	rithin kubectl get nodes			
NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-31051128-0	Ready	agent	64m	v1.15.11
aks-nodepool1-31051128-1	Ready	agent	64m	v1.15.11

Figura 14.11: obtendo a lista de nós

Como nosso nó está no estado **Pronto**, vamos avançar e criar um pod. Há duas formas de criar recursos no Kubernetes. Elas são:

- **Imperativo:** neste método, usamos os comandos **kubectl run** e **kubectl expose** para criar os recursos.
- **Declarativo:** descrevemos o estado do recurso por meio de um arquivo JSON ou YAML. Enquanto abordávamos os primitivos do Kubernetes, você viu muitos arquivos YAML para cada um dos blocos de construção. Vamos passar o arquivo para o comando **kubectl apply** a fim de criar os recursos, e os recursos declarados no arquivo serão criados.

Vamos adotar a abordagem imperativa primeiro, para criar um pod com o nome **webserver**, a operação de um contêiner NGINX com a porta **80** exposta:

```
kubectl run webserver --restart=Never --image nginx --port 80
```

Após a conclusão bem-sucedida do comando, a CLI informará o status:

```
[root@rithin ~]# kubectl run webserver --restart=Never --image nginx --port 80
pod/webserver created
```

Figura 14.12: criando um pod

Agora que experimentamos o método imperativo, vamos continuar com o método declarativo. Você pode usar a estrutura do arquivo YAML que abordamos na subseção *Pod* da seção *Primitivos do Kubernetes* e modificá-la de acordo com seus requisitos.

Usaremos a imagem NGINX, e o pod será chamado **webserver-2**.

Você pode usar qualquer editor de texto e criar o arquivo. O arquivo final será semelhante ao seguinte:

```
apiVersion: v1
kind: Pod
metadata:
  name: webserver-2
  labels:
    appname: nginx
    ostype: linux
spec:
```

```

containers:
  - name: wenserver-2-container
    image: nginx
    ports:
      - containerPort: 80
        protocol: TCP
        name: http

```

No comando **kubectl apply**, passaremos o nome do arquivo para o parâmetro **-f**, conforme mostrado na Figura 14.13, e você poderá ver que o pod foi criado:

```

rithin kubectl apply -f nginx.yaml
pod/webserver-2 created

```

Figura 14.13: criando um pod usando o método declarativo.

Já que criamos os pods, podemos usar o comando **kubectl get pods** para listar todos os pods. O Kubernetes usa o conceito de namespaces para o isolamento lógico dos recursos. Por padrão, todos os comandos estão apontando para o namespace **padrão**. Se você quiser executar uma ação em um namespace específico, poderá passar o nome do namespace por meio do parâmetro **-n**. Na Figura 14.14, você pode ver que **kubectl get pods** retorna os pods que criamos no exemplo anterior, que residem no namespace padrão. Além disso, quando usamos **--all-namespaces**, a saída retorna pods em todos os namespaces:

```

rithin kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webserver   1/1     Running   0          70m
webserver-2  1/1     Running   0          58m
rithin kubectl get pods --all-namespaces
NAMESPACE   NAME           READY   STATUS    RESTARTS   AGE
default     webserver      1/1     Running   0          70m
default     webserver-2    1/1     Running   0          58m
kube-system coredns-698c77c5d7-l2wbd  1/1     Running   0          145m
kube-system coredns-698c77c5d7-v96gq  1/1     Running   0          142m
kube-system coredns-autoscaler-5bd7c6759b-7kpzq 1/1     Running   0          145m
kube-system kube-proxy-95jmg       1/1     Running   0          142m
kube-system kube-proxy-qsfwq       1/1     Running   0          143m
kube-system kubernetes-dashboard-74d8c675bc-jzhcf 1/1     Running   1          145m
kube-system metrics-server-7d654ddc8b-txbt9     1/1     Running   1          145m
kube-system tunnelfront-79fc68b7f-qwzvj      1/1     Running   0          145m

```

Figura 14.14: listando todos os pods

Agora, vamos criar uma implantação simples que executa o NGINX e com um平衡ador de carga que o expõe à Internet. O arquivo YAML será semelhante ao seguinte:

```
#Creating a deployment that runs six replicas of nginx
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-server
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx-server
  template:
    metadata:
      labels:
        app: nginx-server
    spec:
      containers:
        - name: nginx-server
          image: nginx
          ports:
            - containerPort: 80
              name: http
---
#Creating Service
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
```

```

spec:
  ports:
    - port: 80
  selector:
    app: nginx-server
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: nginx-server

```

Usaremos o comando `kubectl apply` e passaremos o arquivo YAML para o parâmetro `-f`.

Após o êxito, todos os três serviços serão criados, e se você executar o comando `kubectl get deployment nginx-server`, verá seis réplicas em execução, conforme mostrado na Figura 14.15, para tornar o serviço altamente disponível:

<code>rithin kubectl get deployment nginx-server</code>
NAME READY UP-TO-DATE AVAILABLE AGE
nginx-server 6/6 6 6 8m17s
<code>rithin kubectl get pods</code>
NAME READY STATUS RESTARTS AGE
nginx-server-777ff65664-j7lgw 1/1 Running 0 8m25s
nginx-server-777ff65664-kxtzx 1/1 Running 0 8m25s
nginx-server-777ff65664-mh8hj 1/1 Running 0 8m25s
nginx-server-777ff65664-wn79x 1/1 Running 0 8m25s
nginx-server-777ff65664-xc7kc 1/1 Running 0 8m25s
nginx-server-777ff65664-xl4nk 1/1 Running 0 8m25s

Figura 14.15: verificando a implantação

Como nossa implantação é provisionada, precisamos verificar qual é o IP público do平衡ador de carga que criamos. Podemos usar o comando `kubectl get service nginx-lb --watch`. Quando o平衡ador de carga estiver inicializando, `EXTERNAL-IP` será exibido como `<pending>`, o parâmetro `--wait` deixará o comando ser executado em primeiro plano e, quando o IP público for alocado, poderemos ver uma nova linha, conforme mostrado aqui

rithin kubectl get service nginx-lb --watch					
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx-lb	LoadBalancer	10.0.140.48	<pending>	80:30686/TCP	68s
nginx-lb	LoadBalancer	10.0.140.48	52.187.70.177	80:30686/TCP	115s

Figura 14.16: encontrando o IP público do balanceador de carga

Agora que temos o IP público, podemos ir para o navegador e ver a página de destino do NGINX, conforme mostrado na Figura 14.17:

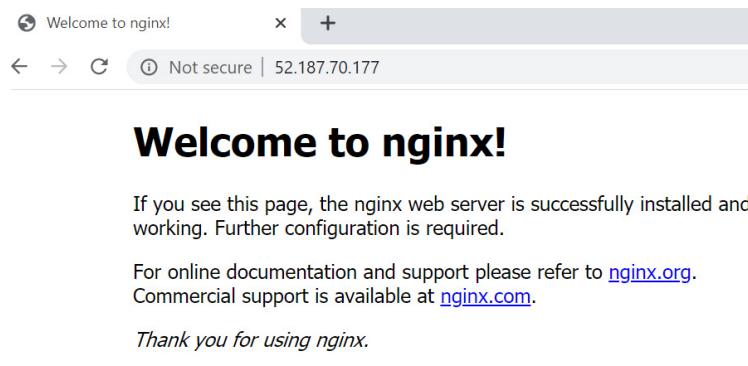


Figura 14.17: página de destino do NGINX

Da mesma forma, você pode usar os arquivos `YAML` que abordamos na seção Primitivos do Kubernetes para criar diferentes tipos de recursos.

Há muitos comandos, como `logs`, `describe`, `exec` e `delete`, que os administradores precisam usar com o comando `kubectl`. O objetivo desta seção é capacitar você a criar um cluster do AKS, se conectar ao cluster e implantar uma aplicação Web simples.

Na próxima seção, abordaremos a rede do AKS.

## Rede do AKS

A rede forma um componente principal dentro de um cluster do Kubernetes. Os componentes mestre devem conseguir alcançar os nós minion e os pods em execução sobre eles, enquanto os nós de trabalhador devem conseguir se comunicar entre si, bem como com os componentes mestre.

Pode ser uma surpresa que o Kubernetes central não gerencie a pilha de rede. É o trabalho do tempo de execução do contêiner nos nós.

O Kubernetes indicou três princípios importantes que todos os tempos de execução do contêiner devem seguir. Eles são seguintes:

- Os pods devem conseguir se comunicar com outros pods sem nenhuma transformação em seus endereços de origem ou destino, algo que é feito usando a **conversão de endereços de rede (NAT)**.
- Agentes como os kubelets devem conseguir se comunicar com pods diretamente nos nós.
- Os pods que estão hospedados diretamente na rede de host ainda devem conseguir se comunicar com todos os pods no cluster.

Cada pod obtém um endereço IP exclusivo no cluster do Kubernetes, juntamente com uma pilha de rede completa, semelhante a máquinas virtuais. Todos eles estão conectados à rede de pontes local criada pelo componente **Interface de Rede de Contêiner (CNI)**. O componente CNI também cria a pilha de rede do pod. Em seguida, a rede de pontes se comunica com a rede de host e se torna o condutor para o fluxo de tráfego de pods para rede e vice-versa.

A CNI é um padrão gerenciado e mantido pela **Cloud Native Computing Foundation (CNCF)**, e há muitos provedores que fornecem sua própria implementação da interface. O Docker é um desses provedores. Há outros, como rkt (leia como "Rocket"), weave, calico e muitos mais. Cada um tem seus próprios recursos e decide de forma independente os recursos de rede, garantindo que os principais princípios da rede do Kubernetes sejam seguidos integralmente.

O AKS fornece dois modelos de rede distintos:

- Kubenet
- CNI do Azure

## Kubenet

O Kubenet é a estrutura de rede padrão no AKS. No Kubenet, cada nó obtém um endereço IP da sub-rede da rede virtual à qual eles estão conectados. Os pods não obtêm endereços IP da sub-rede. Em vez disso, um esquema de endereçamento separado é usado para fornecer endereços IP aos serviços e pods do Kubernetes. Ao criar uma instância do AKS, é importante definir o intervalo de endereços IP para pods e serviços. Como os pods não estão na mesma rede que os nós, as solicitações de pods e para pods são sempre NATed/roteadas para substituir o IP do pod de origem pelo endereço IP do nó e vice-versa.

No roteamento definido pelo usuário, o Azure pode oferecer suporte a até 400 rotas, e você também não pode ter um cluster maior do que 400 nós. A Figura 14.18 mostra como o nó do AKS recebe um endereço IP da rede virtual, mas não os pods criados no nó:

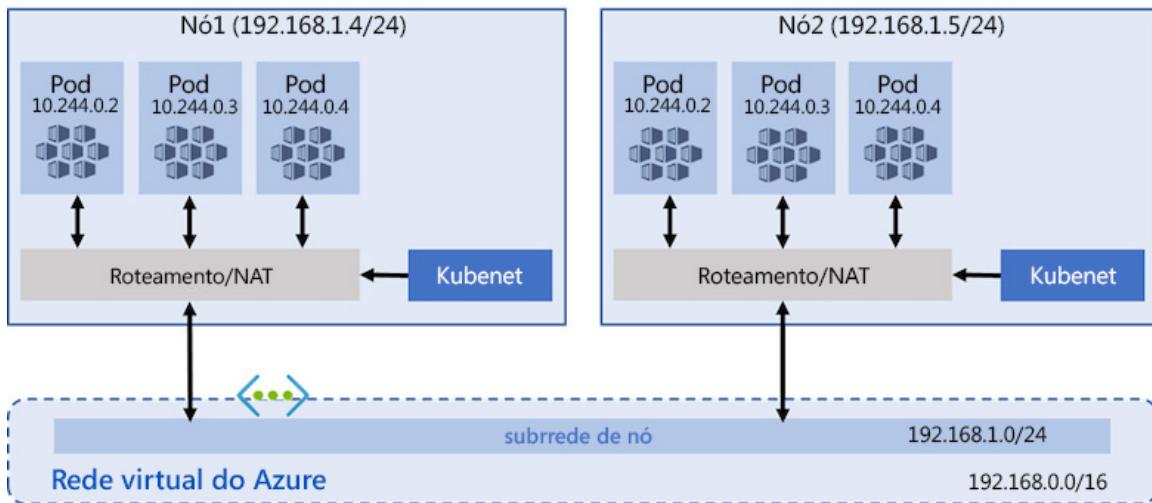


Figura 14.18: rede no AKS

Por padrão, esse Kubenet é configurado com 110 pods por nó. Isso significa que pode haver um máximo de  $110 * 400$  pods em um cluster do Kubernetes por padrão. O número máximo de pods por nó é 250.

Esse esquema deve ser usado quando a disponibilidade do endereço IP e o roteamento definido pelo usuário não são uma restrição.

Na CLI do Azure, você pode executar o seguinte comando para criar uma instância do AKS usando essa pilha de rede:

```
az aks create \
--resource-group myResourceGroup \
--name myAKSCluster \
--node-count 3 \
--network-plugin kubenet \
--service-cidr 10.0.0.0/16 \
--dns-service-ip 10.0.0.10 \
--pod-cidr 10.244.0.0/16 \
--docker-bridge-address 172.17.0.1/16 \
--vnet-subnet-id $SUBNET_ID \
--service-principal <appId> \
--client-secret <password>
```

Observe como todos os endereços IP são explicitamente fornecidos para recursos de serviço, pods, nós e pontes do Docker. Eles são intervalos de endereços IP não sobrepostos. Observe também que o Kubenet é usado como um plugin de rede.

### CNI do Azure (rede avançada)

Com a CNI do Azure, cada nó e pod recebe um endereço IP atribuído da sub-rede de rede diretamente. Isso significa que pode haver tantos pods quanto há endereços IP exclusivos disponíveis em uma sub-rede. Isso torna o planejamento de intervalo de endereços IP muito mais importante com essa estratégia de rede.

É importante observar que a hospedagem do Windows só é possível usando a pilha de rede da CNI do Azure. Além disso, alguns dos componentes do AKS, como nós virtuais e kubelets virtuais, também dependem da pilha de CNI do Azure. É necessário reservar endereços IP com antecedência, dependendo do número de pods que serão criados. Deve haver sempre endereços IP adicionais disponíveis na sub-rede, para evitar a exaustão de endereços IP ou para evitar a necessidade de recriar o cluster para uma sub-rede maior devido à demanda da aplicação.

Por padrão, essa pilha de rede é configurada para 30 pods por nó e pode ser configurada com 250 pods como o número máximo de pods por nó.

O comando para executar para criar uma instância do AKS usando essa pilha de rede é mostrado aqui:

```
az aks create \
  --resource-group myResourceGroup \
  --name myAKScluster \
  --network-plugin azure \
  --vnet-subnet-id <subnet-id> \
  --docker-bridge-address 172.17.0.1/16 \
  --dns-service-ip 10.2.0.10 \
  --service-cidr 10.2.0.0/24 \
  --generate-ssh-keys
```

Observe como todos os endereços IP são explicitamente fornecidos para recursos de serviço, pods, nós e pontes do Docker. Eles são intervalos de endereços IP não sobrepostos. Observe também que o Azure é usado como um plugin de rede.

Até agora, você aprendeu a implantar uma solução e gerenciar a rede de um cluster do AKS. A segurança é outro fator importante que precisa ser abordado. Na próxima seção, vamos nos concentrar em opções de acesso e identidade para o AKS.

## Acesso e identidade para o AKS

Os clusters do Kubernetes podem ser protegidos de várias maneiras.

A conta de serviço é um dos principais tipos de usuário no Kubernetes. A API do Kubernetes gerencia a conta de serviço. Pods autorizados podem se comunicar com o servidor de API usando as credenciais de contas de serviço, que são armazenadas como segredos do kubernetes. O Kubernetes não tem nenhum armazenamento de dados ou provedor de identidade próprio. Ele delega a responsabilidade da autenticação ao software externo. Ele fornece um plugin de autenticação que verifica as credenciais fornecidas e as mapeia para grupos disponíveis. Se a autenticação for bem-sucedida, a solicitação passará para outro conjunto de plugins de autorização para verificar os níveis de permissão do usuário no cluster, bem como os recursos com escopo do namespace.

Para o Azure, a melhor integração de segurança seria usar o Azure AD. Usando o Azure AD, você também pode trazer suas identidades na infraestrutura local para o AKS a fim de fornecer um gerenciamento centralizado de contas e segurança. O fluxo de trabalho básico da integração do Azure AD é mostrado na Figura 14.19:

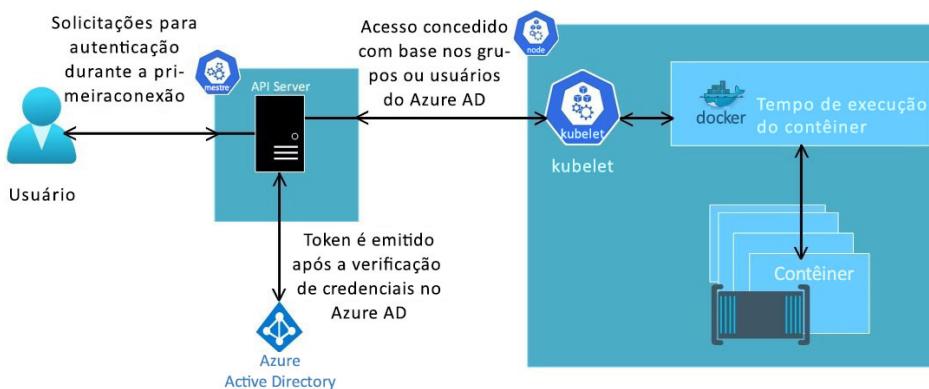


Figura 14.19: fluxo de trabalho básico da integração do Azure AD

Os usuários ou grupos podem ter acesso a recursos em um namespace ou em um cluster. Na seção anterior, usamos o comando `az aks get-credential` para obter as credenciais e o contexto de configuração do kubectl. Quando o usuário tenta interagir com o kubectl, ele é solicitado a fazer logon usando suas credenciais do Azure AD. O Azure AD valida as credenciais e um token é emitido para o usuário. Com base no nível de acesso que ele tem, é possível acessar os recursos no cluster ou no namespace.

Além disso, você pode aproveitar o **Controle de acesso baseado em função (RBAC)** do Azure para limitar o acesso aos recursos no grupo de recursos.

Na próxima seção, abordaremos o kubelet virtual, que é uma das maneiras mais rápidas de expandir um cluster.

## Kubelet virtual

O kubelet virtual está em visualização no momento e é gerenciado pela organização CNCF. É uma abordagem bastante inovadora que o AKS usa para fins de escalabilidade. O kubelet virtual é implantado no cluster do Kubernetes como um pod. O contêiner em execução no pod usa o SDK do Kubernetes para criar um novo recurso de nó e se apresenta para todo o cluster como um nó. Os componentes do cluster, incluindo o servidor de API, o agendador e os controladores, o consideram e tratam como um nó e agendam pods nele.

No entanto, quando um pod é agendado nesse nó que está disfarçado como um nó, ele se comunica com seus componentes de back-end, conhecidos como provedores, para criar, excluir e atualizar os pods. Um dos principais provedores no Azure são Instâncias de Contêiner do Azure. O Lote do Azure também pode ser usado como um provedor. Isso significa que os contêineres são realmente criados em Instâncias de Contêiner ou no Lote do Azure, em vez do próprio cluster. No entanto, eles são gerenciados pelo cluster. A arquitetura do kubelet virtual é mostrada na Figura 14.20:

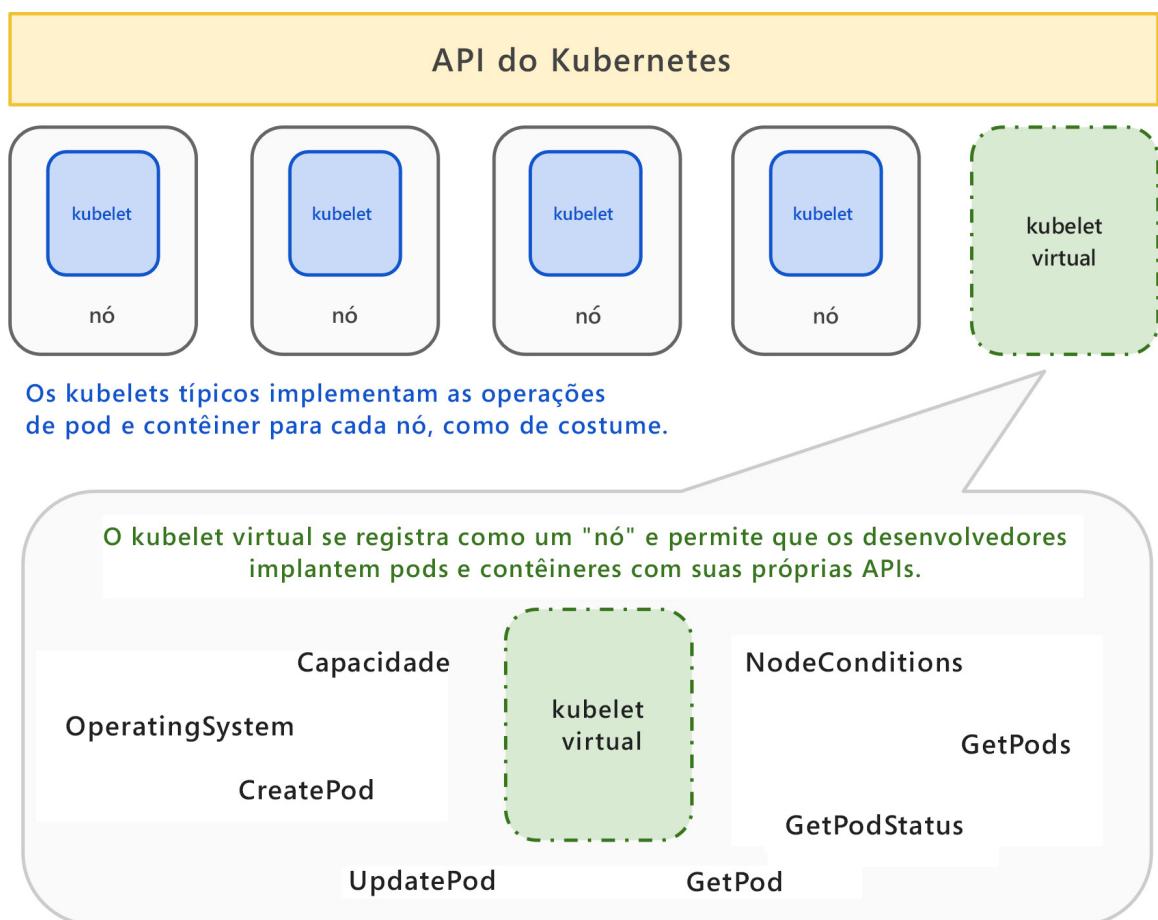


Figura 14.20: arquitetura do kubelet virtual

Observe que o kubelet virtual é representado como um nó no cluster e pode ajudar na hospedagem e no gerenciamento de pods, assim como um kubelet normal faria. No entanto, o kubelet virtual tem uma limitação. É isso que vamos abordar na próxima seção.

## Nós virtuais

Uma das limitações do virtual kubelet é que os pods implantados em provedores do kubelet virtual são isolados e não se comunicam com outros pods no cluster. Se houver a necessidade de os pods nesses provedores se comunicarem com outros pods e nós no cluster e vice-versa, nós virtuais deverão ser criados. Nós virtuais são criados em uma sub-rede diferente na mesma rede virtual que hospeda nós de cluster do Kubernetes, o que pode permitir a comunicação entre pods. No momento da redação, somente o sistema operacional Linux tem suporte para trabalhar com nós virtuais.

Nós virtuais dão a impressão de um nó. No entanto, o nó não existe. Qualquer coisa agendada nesse nó é criada em Instâncias de Contêiner do Azure. Os nós virtuais são baseados no kubelet virtual, mas têm a funcionalidade extra de comunicação integrada entre o cluster e as Instâncias de Contêiner do Azure.

Ao implantar pods em nós virtuais, a definição de pod deve conter um seletor de nó apropriado para se referir a nós virtuais e também tolerâncias, conforme mostrado no trecho de código a seguir:

```
nodeSelector:  
    kubernetes.io/role: agent  
    beta.kubernetes.io/os: linux  
    type: virtual-kubelet  
  
tolerations:  
  - key: virtual-kubelet.io/provider  
    operator: Exists  
  - key: azure.com/aci  
    effect: NoSchedule
```

Aqui, o seletor de nó está usando a propriedade **type** para se referir ao kubelet virtual e a propriedade **tolerations** para informar o Kubernetes de que nós com marcas, **virtual-kubelet.io/provider**, devem permitir a implantação desses pods neles.

## Resumo

O kubernetes é o orquestrador de contêineres mais usado e funciona com diferentes tempos de execução de contêiner e rede. Neste capítulo, você aprendeu sobre os fundamentos do kubernetes, sua arquitetura e alguns dos componentes de infraestrutura importantes, como etcd, o servidor de API, gerenciadores do controlador e o agendador, juntamente com sua finalidade. Além disso, analisamos recursos importantes que podem ser implantados para gerenciar aplicações, como pods, controladores de replicação, ReplicaSets, implantações e serviços.

O AKS fornece algumas pilhas de rede diferentes: a CNI do Azure e o Kubenet. Eles fornecem diferentes estratégias para atribuir endereços IP a pods. Enquanto a CNI do Azure fornece endereços IP para pods da sub-rede subjacente, o Kubenet usa apenas endereços IP virtuais.

Também abordamos alguns dos recursos oferecidos exclusivamente pelo Azure, como nós virtuais, e conceitos em torno do kubelet virtual. No próximo capítulo, aprenderemos sobre o provisionamento e a configuração de recursos com modelos do ARM.



# 15

## Implantações entre assinaturas usando modelos do ARM

**Os modelos do Azure Resource Manager (ARM)** são o mecanismo preferencial para provisionar recursos e configurá-los no Azure.

Os modelos do ARM ajudam a implementar um paradigma relativamente novo conhecido como **Infraestrutura como Código (IaC)**. Os modelos do ARM convertem a infraestrutura e sua configuração em código, que tem inúmeras vantagens. A IaC traz um alto nível de consistência e previsibilidade para implantações entre ambientes. Ela também garante que os ambientes possam ser testados antes de ir para a produção e, por fim, proporciona um alto nível de confiança no processo de implantação, manutenção e governança.

Os tópicos a seguir serão abordados neste capítulo:

- Modelos do ARM
- Implantar grupos de recursos com modelos do ARM
- Implantar recursos em assinaturas e grupos de recursos
- Implantar entre assinaturas e grupos de recursos usando modelos vinculados
- Criar modelos do ARM para soluções de PaaS, dados e IaaS

## Modelos do ARM

Uma vantagem proeminente da IaC é que ela pode ter controle de versão. Ele também pode ser reutilizado entre ambientes, o que fornece um alto grau de consistência e previsibilidade nas implantações e garante que o impacto e o resultado da implantação de um modelo do ARM sejam os mesmos, independentemente do número de vezes que o modelo é implantado. Esse recurso também é conhecido como **idempotência**.

Os modelos do ARM foram apresentados com a introdução da especificação do ARM. Desde então, receberam mais recursos e ganharam maturidade. É importante compreender que geralmente há uma lacuna que dura de algumas semanas a alguns meses entre a configuração de recurso real e a disponibilidade da configuração em modelos do ARM.

Cada recurso tem sua própria configuração. Essa configuração pode ser afetada de diversas maneiras, inclusive usando o Azure PowerShell, a CLI do Azure, os SDKs do Azure, as APIs REST e os modelos do ARM.

Cada uma dessas técnicas tem seu próprio ciclo de vida de desenvolvimento e lançamento, que é diferente do desenvolvimento real de recursos. Vamos tentar entender isso com a ajuda de um exemplo.

O recurso do Azure Databricks tem seu próprio ciclo de vida de cadênciia e desenvolvimento. Os consumidores desse recurso têm seu próprio ciclo de vida de desenvolvimento, que é diferente do desenvolvimento real dos recursos. Se o Databricks receber sua primeira versão em 31 de dezembro, os cmdlets do Azure PowerShell para ele podem não estar disponíveis na mesma data e podem até ser lançados em 31 de janeiro do próximo ano. De maneira semelhante, a disponibilidade desses recursos nos modelos do ARM e da API REST pode ocorrer em torno de 15 de janeiro.

Os modelos do ARM são documentos baseados em JSON que, quando executados, invocam uma API REST no plano de gerenciamento do Azure e enviam o documento inteiro para ele. A API REST tem seu próprio ciclo de vida de desenvolvimento, e o esquema JSON para o recurso tem seu próprio ciclo de vida também.

Isso significa que o desenvolvimento do Azure de uma função dentro de um recurso precisa acontecer em pelo menos três componentes diferentes antes que eles possam ser consumidos em modelos do ARM. Eles incluem:

- O próprio recurso
- A API REST do recurso
- O esquema de recursos do modelo do ARM

Cada recurso no modelo do ARM tem a propriedade **apiVersion**. Essa propriedade ajuda a decidir qual versão da API REST deve ser usada para provisionar e implantar o recurso. A Figura 15.1 mostra o fluxo de solicitações do modelo do ARM para APIs de recursos que são responsáveis pela criação, atualização e exclusão de recursos:



Figura 15.1: fluxo de solicitações

Uma configuração de recurso, como uma conta de armazenamento em um modelo do ARM, tem a seguinte aparência:

```
{
  "type": "Microsoft.Storage/storageAccounts",
  "apiVersion": "2019-04-01",
  "name": "[variables('storage2')]",
  "location": "[resourceGroup().location]",
  "kind": "Storage",
  "sku": {
    "name": "Standard_LRS"
  }
}
```

No código anterior, a disponibilidade desse esquema para definir `sku` se baseia no desenvolvimento do esquema de modelo do ARM. A disponibilidade da API REST e seu número de versão é determinada por `apiVersion`, que é `2019-04-01`. O recurso real é determinado pela propriedade `type`, que tem as duas partes a seguir:

- **Namespace do provedor de recursos:** os recursos no Azure são hospedados em namespaces, e os recursos relacionados são hospedados no mesmo namespace.
- **Tipo de recurso:** os recursos são mencionados usando o nome do tipo.

Nesse caso, o recurso é identificado pelo nome e tipo de provedor, que é `Microsoft.Storage/storageaccounts`.

Anteriormente, os modelos do ARM esperavam que os grupos de recursos estivessem disponíveis antes da implantação. Eles também eram limitados à implantação em um único grupo de recursos em uma única assinatura.

Isso significava que, até recentemente, um modelo do ARM poderia implantar todos os recursos em um único grupo de recursos. Os modelos de ARM do Azure agora têm a funcionalidade adicional para implantar recursos em vários grupos de recursos dentro da mesma assinatura ou várias assinaturas simultaneamente. Agora é possível criar grupos de recursos como parte dos modelos do ARM. Portanto, agora é possível implantar recursos de várias regiões em grupos de recursos diferentes.

Por que precisamos criar grupos de recursos com base em modelos do ARM e ter implantações entre assinaturas e grupos de recursos simultaneamente?

Para aproveitar o valor da criação de uma implantação entre assinaturas e grupos de recursos, precisamos entender como as implantações foram realizadas antes que esses recursos fossem disponibilizados.

Para implantar um modelo do ARM, um grupo de recursos é um pré-requisito. Os grupos de recursos devem ser criados antes da implantação de um modelo. Os desenvolvedores usam o PowerShell, a CLI do Azure ou a API REST para criar grupos de recursos e, em seguida, iniciar a implantação de modelos do ARM. Isso significa que qualquer implantação de ponta a ponta consiste em várias etapas. A primeira etapa é provisionar o grupo de recursos, e a próxima é a implantação do modelo do ARM neste grupo de recursos recém-criado. Essas etapas podem ser executadas usando um único script do PowerShell ou etapas individuais da linha de comando do PowerShell. O script do PowerShell deve ser concluído quanto ao código relacionado a tratar exceções, cuidar de casos de borda e garantir que não haja bugs nele para que se possa afirmar que está pronto para empresas. É importante observar que os grupos de recursos podem ser excluídos do Azure e, na próxima vez que o script for executado, a disponibilidade deles pode ser esperada. Haverá falha porque ele poderá assumir que o grupo de recursos existe. Resumindo, a implantação do modelo do ARM em um grupo de recursos deve ser uma etapa atômica, e não várias etapas.

Compare isso com a capacidade de criar grupos de recursos e seus recursos constituintes juntos nos mesmos modelos do ARM. Sempre que você implanta o modelo, ele cria grupos de recursos, se eles ainda não existirem, e continua a implantar recursos para eles após a criação.

Veremos também como esses novos recursos podem ajudar a remover algumas das restrições técnicas relacionadas a sites de Disaster Recovery.

Antes desses recursos, se você tivesse que implantar uma solução que foi criada pensando na Disaster Recovery, havia duas implantações separadas: uma para a região primária e outra para a secundária. Por exemplo, se estivesse implantando uma aplicação MVC ASP.NET usando o Serviço de Aplicativo, você criaria um serviço de aplicativo e o configuraria para a região primária. Em seguida, você conduziria outra implantação com o mesmo modelo para outra região usando um arquivo **parameters** diferente. Ao implantar outro conjunto de recursos em outra região, como mencionado anteriormente, os parâmetros usados com o modelo devem ser diferentes para refletir as diferenças entre os dois ambientes. Os parâmetros incluiriam alterações como uma cadeia de conexão do SQL, endereços de domínio e IP e outros itens de configuração exclusivos para um ambiente.

Com a disponibilidade de implantações entre assinaturas e grupos de recursos, é possível criar o site de Disaster Recovery ao mesmo tempo que o site primário. Isso elimina duas implantações e garante que a mesma configuração possa ser usada em vários sites.

## Implantar grupos de recursos com modelos do ARM

Nesta seção, um modelo do ARM será criado e implantado, o que criará alguns grupos de recursos na mesma assinatura.

Para usar o PowerShell para implantar modelos que contenham grupos de recursos e recursos entre assinaturas, é necessário usar a versão mais recente do PowerShell. No momento da redação, a versão 3.3.0 do módulo do Azure está sendo usada:

```
PS C:\WINDOWS\system32> get-module -Name az
ModuleType Version      Name
-----  -----      --
Script     3.3.0        az
```

Figura 15.2: verificando a versão mais recente do módulo do Azure

Se o módulo mais recente do Azure não estiver instalado, faça isso usando o seguinte comando:

```
install-module -Name az -Force
```

É hora de criar um modelo do ARM que criará vários grupos de recursos na mesma assinatura. O código do modelo do ARM é o seguinte:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "resourceGroupInfo": {
      "type": "array"
    },
    "multiLocation": {
      "type": "array"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "location": "[parameters('multiLocation')[copyIndex()]]",
      "name": "[parameters('resourceGroupInfo')[copyIndex()]]",
      "apiVersion": "2019-10-01",
      "copy": {
        "name": "allResourceGroups",
        "count": "[length(parameters('resourceGroupInfo'))]"
      },
      "properties": {}
    }
  ],
  "outputs": {}
}
```

A primeira seção do código é sobre os parâmetros que os modelos do ARM esperam. Eles são parâmetros obrigatórios, e qualquer pessoa que implantar esses modelos deverá fornecer valores para eles. É necessário fornecer valores de matriz para ambos os parâmetros.

A segunda seção principal é a matriz JSON **resources**, que pode conter vários recursos. Neste exemplo, estamos criando grupos de recursos. Portanto, isso é declarado na seção **resources**. Os grupos de recursos estão sendo provisionados em um loop devido ao uso do elemento **copy**. O elemento **copy** garante que o recurso seja executado em um número especificado de vezes e crie um novo recurso em todas as iterações. Se enviarmos dois valores para o parâmetro de matriz **resourceGroupInfo**, o comprimento da matriz será "dois", e o elemento **copy** garantirá que o recurso **resourceGroup** seja executado duas vezes.

Todos os nomes de recursos em um modelo devem ser exclusivos para um tipo de recurso. A função **copyIndex** é usada para atribuir o número de iteração atual ao nome geral do recurso e torná-lo único. Além disso, queremos que os grupos de recursos sejam criados em regiões diferentes usando nomes de regiões distintos enviados como parâmetros. A atribuição de um nome e local para cada grupo de recursos é feita usando a função **copyIndex**.

O código do arquivo **parameters** é mostrado a seguir. Esse código é bastante simples e fornece valores de matriz para os dois parâmetros esperados pelo modelo anterior. Os valores nesse arquivo devem ser alterados para todos os parâmetros de acordo com o seu ambiente:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentParameters.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "resourceGroupInfo": {  
            "value": [ "firstResourceGroup", "SeocndResourceGroup" ]  
        },  
        "multiLocation": {  
            "value": [  
                "West Europe",  
                "East US"  
            ]  
        }  
    }  
}
```

## Implantar modelos do ARM

Para implantar esse modelo com o PowerShell, faça logon no Azure com credenciais válidas usando o seguinte comando:

```
Login-AzAccount
```

As credenciais válidas podem ser uma conta de usuário ou uma entidade de serviço. Em seguida, use um cmdlet **New-AzDeployment** recém-lançado para implantar o modelo. O script de implantação está disponível no arquivo **multipleResourceGroups.ps1**:

```
 New-AzDeployment -Location "West Europe" -TemplateFile "c:\users\rites\source\repos\CrossSubscription\CrossSubscription\multipleResourceGroups.json" -TemplateParameterFile "c:\users\rites\source\repos\CrossSubscription\CrossSubscription\multipleResourceGroups.parameters.json" -Verbose
```

É importante entender que o cmdlet **New-AzResourceGroupDeployment** não pode ser usado aqui porque o escopo do cmdlet **New-AzResourceGroupDeployment** é um grupo de recursos e espera que um grupo de recursos esteja disponível como um pré-requisito. Para implantar recursos no nível da assinatura, o Azure lançou um novo cmdlet que pode funcionar acima do escopo do grupo de recursos. O novo cmdlet, **new-AzDeployment**, funciona no nível da assinatura. Também é possível ter uma implantação no nível do grupo de gerenciamento. Os grupos de gerenciamento estão em um nível mais alto do que as assinaturas usando o cmdlet **New-AzManagementGroupDeployment**.

## Implantação de modelos usando a CLI do Azure

O mesmo modelo também pode ser implantado usando a CLI do Azure. Veja a seguir as etapas para implantá-lo usando a CLI do Azure:

1. Use a versão mais recente da CLI do Azure para criar grupos de recursos com o modelo do ARM. No momento da redação, a versão 2.0.75 foi usada para a implantação, conforme mostrado aqui:

```
C:\Users\Ritesh>az --version
azure-cli                  2.0.75 *
command-modules-nspkg       2.0.3
core                        2.0.75 *
nspkg                       3.0.4
telemetry                   1.0.4

Extensions:
aks-preview                 0.4.18
azure-devops                0.16.0

Python location 'C:\Program Files (x86)\Microsoft SDKs\Azure\CLI2\python.exe'
Extensions directory 'C:\Users\Ritesh\.azure\cliextensions'

Python (Windows) 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 02:47:15) [MSC v.1900 32 bit (Intel)]
Legal docs and information: aka.ms/AzureCliLegal
```

Figura 15.3: verificando a versão da CLI do Azure

2. Faça logon no Azure usando o comando a seguir e selecione a assinatura certa a ser usada:

```
az login
```

3. Se o logon tiver acesso a várias assinaturas, selecione a assinatura apropriada usando o seguinte comando:

```
az account set -subscription xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

4. Execute a implantação usando o comando a seguir. O script de implantação está disponível no arquivo **multipleResourceGroupsCLI.txt**:

```
C:\Users\Ritesh>az deployment create--location westus--template-file "C:\users\rites\source\repos\CrossSubscription\CrossSubscription\azuredeploy.json--parameters @"C:\users\rites\source\repos\CrossSubscription\CrossSubscription\azuredeploy.parameters.json"--verbose
```

Depois que o comando for executado, os recursos definidos no modelo do ARM devem ser refletidos no portal do Azure.

## Implantar recursos em assinaturas e grupos de recursos

Na última seção, os grupos de recursos foram criados como parte dos modelos do ARM. Outro recurso no Azure é o provisionamento de recursos em várias assinaturas simultaneamente com base em uma única implantação usando um único modelo do ARM. Nesta seção, forneceremos uma nova conta de armazenamento em duas assinaturas e grupos de recursos diferentes. A pessoa que está implantando o modelo do ARM seleciona uma das assinaturas como a assinatura base. Com ela, essa pessoa inicia a implantação e provisiona a conta de armazenamento na assinatura atual e em outra. O pré-requisito para implantar esse modelo é que a pessoa que está fazendo a implantação deve ter acesso a pelo menos duas assinaturas e direitos de colaborador nelas. A listagem de código é mostrada aqui e está disponível no arquivo **CrossSubscriptionStorageAccount.json** no código que o acompanha:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "storagePrefix1": {  
            "type": "string",  
            "defaultValue": "st01"  
        ...  
        "type": "string",  
        "defaultValue": "rg01"  
    },  
    "remoteSub": {  
        "type": "string",  
        "defaultValue": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
    }  
...  
        }  
    }  
},  
    "outputs": {}  
}  
}  
}  
]  
,"outputs": {}  
}
```

É importante observar que os nomes do grupo de recursos usados no código já devem estar disponíveis nas respectivas assinaturas. O código lançará um erro se os grupos de recursos não estiverem disponíveis. Além disso, os nomes do grupo de recursos devem corresponder exatamente aos nomes no modelo do ARM.

O código para implantar esse modelo é mostrado a seguir. Nesse caso, usamos **New-AzResourceGroupDeployment** porque o escopo da implantação é um grupo de recursos. O script de implantação está disponível no arquivo **CrossSubscriptionStorageAccount.ps1** no pacote do código:

```
New-AzResourceGroupDeployment -TemplateFile "<< path to your  
CrossSubscriptionStorageAccount.json file >>" -ResourceGroupName "<<provide  
your base subscription resource group name>>" -storagePrefix1 <<provide prefix  
for first storage account>> -storagePrefix2 <<provide prefix for first storage  
account>> -verbose
```

Depois que o comando for executado, os recursos definidos no modelo do ARM devem ser refletidos no portal do Azure.

## Outro exemplo de implantações entre assinaturas e grupos de recursos

Nesta seção, criamos duas contas de armazenamento em duas assinaturas, grupos de recursos e regiões diferentes de um modelo do ARM e uma única implantação. Usaremos a abordagem de modelos aninhados junto com o elemento **copy** para fornecer nomes e locais diferentes para esses grupos de recursos em assinaturas distintas.

No entanto, antes de podermos executar o próximo conjunto de modelos do ARM, uma instância do Azure Key Vault deve ser provisionada como um pré-requisito, e um segredo deve ser adicionado a ela. Isso acontece porque os nomes das contas de armazenamento são recuperados do Azure Key Vault e transmitidos como parâmetros a modelos do ARM para provisionar a conta de armazenamento.

Para provisionar o Azure Key Vault usando o Azure PowerShell, o próximo conjunto de comandos pode ser executado. O código para os seguintes comandos está disponível no arquivo **CreateKeyVaultandSetSecret.ps1**:

```
New-AzResourceGroup -Location <>replace with location of your key vault>>
-Name <>replace with name of your resource group for key vault>> -verbose
New-AzureRmKeyVault -Name <>replace with name of your key vault>>
-ResourceGroupName <>replace with name of your resource group for
key vault>> -Location <>replace with location of your key vault>>
-EnabledForDeployment -EnabledForTemplateDeployment -EnabledForDiskEncryption
-EnableSoftDelete -EnablePurgeProtection -Sku Standard -Verbose
```

Você deve ter em mente que o valor **ResourceID** deve ser observado a partir do resultado do cmdlet **New-AzKeyVault**. Esse valor precisará ser substituído no arquivo **parameters**. Consulte a Figura 15.4 para obter detalhes:



Figura 15.4: criando uma instância do Key Vault

Execute o seguinte comando para adicionar um novo segredo à instância recém-criada do Azure Key Vault:

```
Set-AzKeyVaultSecret -VaultName <>replace with name of your key vault>> -Name <>replace with name of yoursecret>> -SecretValue $(ConvertTo-SecureString -String <>replace with value of your secret>> -AsPlainText -Force ) -Verbose
```

A listagem de código está disponível no arquivo

**CrossSubscriptionNestedStorageAccount.json** no pacote do código:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "hostingPlanNames": {
            "type": "array",
            "minLength": 1
        },
        ...
        "type": "Microsoft.Resources/deployments",
        "name": "deployment01",
        "apiVersion": "2019-10-01",
        "subscriptionId": "[parameters('subscriptions')[copyIndex()]]",
        "resourceGroup": "[parameters('resourceGroups')[copyIndex()]]",
        "copy": {
            "count": "[length(parameters('hostingPlanNames'))]",
            "name": "mywebsites",           "mode": "Parallel"
        },
        ...
        "kind": "Storage",
        "properties": {
        }
    }
}
...
...
```

Veja a seguir o código do arquivo **parameters**. Ele está disponível no arquivo **CrossSubscriptionNestedStorageAccount.parameters.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentParameters.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "hostingPlanNames": {  
            ...  
            "storageKey": {  
                "reference": {  
                    "keyVault": { "id": "<<replace it with the value of Key vault  
ResourceId noted before>>" },  
                    "secretName": "<<replace with the name of the secret available in  
Key vault>>"  
                }  
            }  
        }  
    }  
}
```

Veja a seguir o código do PowerShell para implantar o modelo anterior. O script de implantação está disponível no arquivo **CrossSubscriptionNestedStorageAccount.ps1**:

```
New-AzResourceGroupDeployment -TemplateFile "c:\users\rites\source\repos\  
CrossSubscription\CrossSubscription\CrossSubscriptionNestedStorageAccount.  
json" -ResourceGroupName rg01 -TemplateParameterFile "c:\  
users\rites\source\repos\CrossSubscription\CrossSubscription\  
CrossSubscriptionNestedStorageAccount.parameters.json" -Verbose
```

Depois que o comando é executado, os recursos definidos no modelo do ARM devem ser refletidos no portal do Azure.

## Implantar entre assinaturas e grupos de recursos usando modelos vinculados

O exemplo anterior usava modelos aninhados para implantação em várias assinaturas e grupos de recursos. No próximo exemplo, implantaremos vários planos do Serviço de Aplicativo em assinaturas e grupos de recursos separados usando modelos vinculados. Os modelos vinculados são armazenados no Armazenamento de Blobs do Azure, que é protegido com políticas. Isso significa que somente o titular da chave da conta de armazenamento ou uma assinatura de acesso compartilhado válida pode acessar esse modelo. A chave de acesso é armazenada no Azure Key Vault e acessada no arquivo **parameters** usando referências do elemento **storageKey**. Os leitores devem carregar o arquivo **website.json** em um contêiner do Armazenamento de Blobs do Azure. O arquivo **website.json** é um modelo vinculado responsável por provisionar um plano do Serviço de Aplicativo e um serviço de aplicativo. O arquivo é protegido usando a política **Privada (sem acesso anônimo)**, conforme mostrado na Figura 15.5. Uma política de privacidade garante que o acesso anônimo não seja permitido. Para este exemplo, criei um contêiner chamado **armtemplates** e o defini com uma política privada:

The screenshot shows the Azure Storage Explorer interface. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Storage Explorer (preview). The main area shows a storage account named 'st02gwldcxm5suwe - Blobs'. Inside, there's a container named 'armtemplates'. An 'Access policy' blade is open on the right, titled 'armtemplates'. It shows a 'Public access level' dropdown set to 'Private (no anonymous access)'. Below it, there's a table for 'Stored access policies' with columns: IDENTIFIER, START TIME, EXPIRY TIME, and PERMISSIONS. The table shows 'No results'. At the bottom of the blade is a '+ Add policy' button.

Figura 15.5: definindo uma política privada para o contêiner

Esse arquivo só pode ser acessado usando as chaves **Assinatura de Acesso Compartilhado (SAS)**. As chaves SAS podem ser geradas no portal do Azure para uma conta de armazenamento usando o item **Assinatura de Acesso Compartilhado** no menu à esquerda, conforme mostrado na Figura 15.6. Você deve clicar no botão **Gerar a cadeia de conexão e de SAS** para gerar o token SAS. Lembre-se de que um token SAS é exibido uma vez e não é armazenado no Azure. Então, copie e armazene-o em algum lugar para que ele possa ser carregado no Azure Key Vault. A Figura 15.6 mostra a geração do token SAS:

Figura 15.6: gerando um token SAS no portal do Azure

Usaremos a mesma instância do Key Vault que foi criada na seção anterior. Só precisamos garantir que haja dois segredos disponíveis na instância do Key Vault. O primeiro segredo é **StorageName**, e o outro é **StorageKey**. Os comandos para criar esses segredos na instância do Key Vault são:

```
Set-AzKeyVaultSecret -VaultName "testkeyvaultbook" -Name "storageName"
-SecretValue $(ConvertTo-SecureString -String "uniquename" -AsPlainText
-Force ) -Verbose
```

```
Set-AzKeyVaultSecret -VaultName "testkeyvaultbook" -Name "storageKey"
-SecretValue $(ConvertTo-SecureString -String "?sv=2020-03-28&ss=bfqt&srt=sc
o&sp=rwdlacup&se=2020-03-30T21:51:03Z&st=2020-03-30T14:51:03Z&spr=https&sig=
gTynGhj20er6pDl7Ab%2Bpc29W03%2BJhvi%2Bff%2F6rHYWp4g%3D" -AsPlainText -Force
) -Verbose
```

É recomendável alterar os nomes da instância do Key Vault e o valor da chave secreta com base na sua respectiva conta de armazenamento.

Depois de garantir que a instância do Key Vault tenha os segredos necessários, o código do arquivo do modelo do ARM poderá ser usado para implantar os modelos aninhados entre assinaturas e grupos de recursos.

O código do modelo do ARM está disponível no arquivo **CrossSubscriptionLinkedStorageAccount.json** e também é mostrado aqui. É recomendável alterar o valor da variável **templateUrl** nesse arquivo. Ele deve ser atualizado com um local de arquivo válido do Armazenamento de Blobs do Azure:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanNames": {
      "type": "array",
      "minLength": 1
    },
    ...
    "type": "Microsoft.Resources/deployments",
    "name": "fsdfsdf",
    "apiVersion": "2019-10-01",
    "subscriptionId": "[parameters('subscriptions')[copyIndex()]]",
    "resourceGroup": "[parameters('resourceGroups')[copyIndex()]]",
    "copy": {
      "count": "[length(parameters('hostingPlanNames'))]",
      "name": "mywebsites",
      "mode": "Parallel"
    },
    ...
  }
}
```

O código do arquivo **parameters** é mostrado a seguir. É recomendável alterar os valores dos parâmetros, incluindo o **resourceid** da instância do Key Vault e o nome do segredo. Os nomes dos serviços de aplicativo devem ser exclusivos ou o modelo não será implantado. O código do arquivo **parameters** está disponível no arquivo de código **CrossSubscriptionLinkedStorageAccount.parameters.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentParameters.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "hostingPlanNames": {  
            "value": [ "firstappservice", "secondappservice" ]  
        },  
        ...  
        "storageKey": {  
            "reference": {  
                "keyVault": { "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-  
xxxxxxxxxxxx/resourceGroups/keyvaluedemo/providers/Microsoft.KeyVault/  
vaults/forsqlvault1" },  
                "secretName": "storageKey"  
            }  
        }  
    }  
}
```

Veja a seguir o comando para implantar o modelo. O script de implantação está disponível no arquivo **CrossSubscriptionLinkedStorageAccount.ps1**:

```
New-AzureRmResourceGroupDeployment -TemplateFile "c:\users\  
rites\source\repos\CrossSubscription\CrossSubscription\  
CrossSubscriptionLinkedStorageAccount.json" -ResourceGroupName <>replace  
with the base subscription resource group name >> -TemplateParameterFile  
"c:\users\rites\source\repos\CrossSubscription\CrossSubscription\  
CrossSubscriptionLinkedStorageAccount.parameters.json" -Verbose
```

Depois que o comando é executado, os recursos definidos no modelo do ARM devem ser refletidos no portal do Azure.

Agora que você sabe como provisionar recursos entre grupos de recursos e assinaturas, veremos algumas das soluções que podem ser criadas usando modelos do ARM.

## Soluções de máquina virtual usando modelos do ARM

**Os recursos e as soluções de infraestrutura como serviço (IaaS)** podem ser implantados e configurados usando modelos do ARM. Os principais recursos relacionados à IaaS são recursos de máquina virtual.

A criação de um recurso de máquina virtual depende de vários outros recursos no Azure. Alguns dos recursos necessários para criar uma máquina virtual incluem:

- Uma conta de armazenamento ou um disco gerenciado para hospedar o sistema operacional e o disco de dados
- Uma rede virtual juntamente com sub-redes
- Uma placa de interface de rede

Há outros recursos que são opcionais, incluindo:

- Azure Load Balancer
- Grupos de segurança de rede
- Endereço IP público
- Tabelas de rotas e muito mais

Esta seção tratará do processo de criação de máquinas virtuais usando modelos do ARM. Como mencionado anteriormente nesta seção, precisamos criar alguns recursos, dos quais o recurso de máquina virtual dependerá, antes de criar o recurso de máquina virtual em si.

É importante observar que nem sempre é necessário criar os recursos dependentes. Eles devem ser criados somente se já não existirem. Se eles já estão disponíveis na assinatura do Azure, o recurso de máquina virtual pode ser provisionado, fazendo referência a esses recursos dependentes.

O modelo depende de alguns parâmetros que devem ser fornecidos a ele no momento da execução do modelo. Essas variáveis estão relacionadas à localização dos recursos e alguns de seus valores de configuração. Esses valores são extraídos de parâmetros porque podem mudar de uma implantação para outra. Portanto, o uso de parâmetros ajuda a manter o modelo genérico.

A primeira etapa é criar uma conta de armazenamento, conforme mostrado no código a seguir:

```
{  
    "type": "Microsoft.Storage/storageAccounts",  
    "name": "[variables('storageAccountName')]",  
    "apiVersion": "2019-04-01",  
    "location": "[parameters('location')]",  
    "sku": {  
        "name": "Standard_LRS"  
    },  
    "kind": "Storage",  
    "properties": {}  
},
```

Depois de criar uma conta de armazenamento, uma rede virtual deve ser definida no modelo do ARM. É importante observar que não há nenhuma dependência entre uma conta de armazenamento e uma rede virtual. Elas podem ser criadas em paralelo. O recurso de rede virtual tem uma sub-rede como seu sub-recurso. Os dois são configurados com seus intervalos de IP; a sub-rede normalmente tem um intervalo menor do que o intervalo de IP da rede virtual:

```
{  
    "apiVersion": "2019-09-01",  
    "type": "Microsoft.Network/virtualNetworks",  
    "name": "[variables('virtualNetworkName')]",  
    "location": "[parameters('location')]",  
    "properties": {  
        "addressSpace": {  
            "addressPrefixes": [  
                "[variables('addressPrefix')]"  
            ]  
        },  
        "subnets": [  
            {  
                "name": "[variables('subnetName')]",  
                "properties": {  
                    "addressPrefix": "[variables('subnetPrefix')]"  
                }  
            }  
        ]  
    }  
},
```

```
    }
]
},
},
```

Se a máquina virtual precisar ser acessada pela Internet pública, um endereço IP público também poderá ser criado, conforme mostrado no código a seguir. Novamente, é um recurso completamente independente e pode ser criado em paralelo com a conta de armazenamento e a rede virtual:

```
{
  "apiVersion": "2019-11-01",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "[variables('publicIPAddressName')]",
  "location": "[parameters('location')]",
  "properties": {
    "publicIPAllocationMethod": "Dynamic",
    "dnsSettings": {
      "domainNameLabel": "[parameters('dnsLabelPrefix')]"
    }
  }
},
```

Depois de criar a rede virtual, a conta de armazenamento e o endereço IP público, uma interface de rede pode ser criada. Uma interface de rede depende de uma rede virtual e de um recurso de sub-rede. Opcionalmente, ela também pode ser associada a um endereço IP público. Isso é mostrado no código a seguir:

```
{
  "apiVersion": "2019-11-01",
  "type": "Microsoft.Network/networkInterfaces",
  "name": "[variables('nicName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
```

```
    "[resourceId('Microsoft.Network/publicIPAddresses/',
variables('publicIPAddressName'))]",
    "[resourceId('Microsoft.Network/virtualNetworks/',
variables('virtualNetworkName'))]"
],
"properties": {
    "ipConfigurations": [
        {
            "name": "ipconfig1",
            "properties": {
                "privateIPAllocationMethod": "Dynamic",
                "publicIPAddress": {
                    "id": "[resourceId('Microsoft.Network/
publicIPAddresses',variables('publicIPAddressName'))]"
                },
                "subnet": {
                    "id": "[variables('subnetRef')]"
                }
            }
        }
    ]
},
},
```

É importante observar que o endereço IP público e a sub-rede são referenciados por seus identificadores exclusivos do Azure.

Após a criação da interface de rede, temos todos os recursos necessários para criar uma máquina virtual. O próximo bloco de código mostra como criar uma máquina virtual usando um modelo do ARM. Ele tem uma dependência da placa de rede e da conta de armazenamento. Isso indiretamente cria dependências na rede virtual, na sub-rede e no endereço IP público.

Para a máquina virtual, definimos a configuração de recursos obrigatórios, incluindo **type**, **apiVersion**, **location** e **name**, juntamente com todas as dependências, conforme mostrado no código a seguir:

```
{  
  "apiVersion": "2019-07-01",  
  "type": "Microsoft.Compute/virtualMachines",  
  "name": "[variables('vmName')]",  
  "location": "[resourceGroup().location]",  
  "tags": {  
    "displayName": "VirtualMachine"  
  },  
  "dependsOn": [  
    "[concat('Microsoft.Storage/storageAccounts/',  
    variables('storageAccountName'))]",  
    "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"  
  ],  
  "properties": {  
    "hardwareProfile": { "vmSize": "[variables('vmSize')]" },  
    "availabilitySet": {  
      "id": "[resourceId('Microsoft.Compute/availabilitySets',  
      parameters('adAvailabilitySetName'))]"  
    },  
    "osProfile": {  
      "computerName": "[variables('vmName')]",  
      "adminUsername": "[parameters('adminUsername')]",  
      "adminPassword": "[parameters('adminPassword')]"  
    },  
    "storageProfile": {  
      "imageReference": {
```

```
"publisher": "[variables('imagePublisher')]",  
"offer": "[variables('imageOffer')]",  
"sku": "[parameters('windowsOSVersion')]",  
"version": "latest"  
},  
"osDisk": { "createOption": "FromImage" },  
"copy": [  
    {  
        "name": "dataDisks",  
        "count": 3,  
        "input": {  
            "lun": "[copyIndex('dataDisks')]",  
            "createOption": "Empty",  
            "diskSizeGB": "1023",  
            "name": "[concat(variables('vmName'), '-datadisk', copyIndex('dataDisks'))]"  
        }  
    }  
]  
},  
"networkProfile": {  
    "networkInterfaces": [  
        {  
            "id": "[resourceId('Microsoft.Network/networkInterfaces',  
variables('nicName'))]"  
        }  
    ]  
}  
}
```

No código anterior, a máquina virtual é configurada com:

- Um perfil de hardware: o tamanho da máquina virtual.
- Um perfil do sistema operacional: o nome e as credenciais para fazer logon na máquina virtual.
- Um perfil de armazenamento: a conta de armazenamento na qual armazenar o arquivo do **disco rígido virtual (VHD)** para a máquina virtual, incluindo discos de dados.
- Um perfil de rede: a referência à placa de interface de rede.

A próxima seção mostrará um exemplo de como usar modelos do ARM para provisionar uma solução de Plataforma como Serviço.

## Soluções PaaS usando modelos do ARM

**Os recursos e as soluções de Plataforma como Serviço (PaaS)** podem ser implantados usando modelos do ARM. Um dos principais recursos relacionados à PaaS é os Aplicativos Web do Azure, e nesta seção, vamos nos concentrar em criar aplicativos Web no Azure usando modelos do ARM.

O modelo espera que alguns parâmetros sejam fornecidos durante a execução. Os parâmetros necessários são a SKU para o plano do Serviço de Aplicativo, a região do Azure que hospeda os recursos e a capacidade de SKU do plano do Serviço de Aplicativo.

Há algumas variáveis declaradas no modelo para torná-lo genérico e sustentável. O primeiro, **hostingPlanName**, é para o nome do plano do Serviço de Aplicativo, e o próximo, **WebSiteName**, é para o próprio serviço de aplicativo.

Há no mínimo dois recursos que devem ser declarados e provisionados para um aplicativo Web funcional no Azure. São eles:

- O plano do Serviço de Aplicativo do Azure
- Serviço de Aplicativo do Azure

A primeira etapa na criação de um aplicativo Web no Azure é definir a configuração de um plano do Serviço de Aplicativo do Azure. O código a seguir define um novo plano do Serviço de Aplicativo. É importante observar que o tipo de recurso é **Microsoft.Web/serverfarms**. A maioria dos valores de configuração do plano, como **location**, **name** e **capacity**, são parâmetros do modelo do ARM:

```
{  
    "apiVersion": "2019-08-01",  
    "name": "[variables('hostingPlanName')]",  
    "type": "Microsoft.Web/serverfarms",  
    "location": "[parameters('location')]",  
    "tags": {  
        "displayName": "HostingPlan"  
    },  
    "sku": {  
        "name": "[parameters('skuName')]",  
        "capacity": "[parameters('skuCapacity')]"  
    },  
    "properties": {  
        "name": "[variables('hostingPlanName')]"  
    }  
},
```

O próximo recurso que deve ser provisionado após um plano é o serviço de aplicativo em si. É importante que uma dependência entre esses recursos seja criada de forma que um plano já seja criado antes que o serviço de aplicativo em si seja criado:

```
{  
    "apiVersion": "2019-08-01",  
    "name": "[variables('webSiteName')]",  
    "type": "Microsoft.Web/sites",  
    "location": "[parameters('location')]",  
    "dependsOn": [  
        "[variables('hostingPlanName')]"  
    ],  
    "properties": {  
        "name": "[variables('webSiteName')]",  
        "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',  
            variables('hostingPlanName'))]"  
    },  
    "resources": [  
        {  
            "type": "Microsoft.Web/sites/config",  
            "name": "[concat(variables('webSiteName'), '/DefaultConfig')]",  
            "dependsOn": [  
                "[variables('hostingPlanName')]"  
            ],  
            "properties": {  
                "name": "[concat(variables('webSiteName'), '/DefaultConfig')]",  
                "hostingEnvironment": "Standard",  
                "connectionStrings": [  
                    {  
                        "name": "[concat(variables('webSiteName'), '/MyConnectionString')]",  
                        "connectionString": "[variables('myConnectionString')]"  
                    }  
                ]  
            }  
        }  
    ]  
}
```

```
{  
  "apiVersion": "2019-08-01",  
  "type": "config",  
  "name": "connectionstrings",  
  "dependsOn": [  
    "[variables('webSiteName')]"  
  ],  
  "properties": {  
    "DefaultConnection": {  
      "value": "[concat( 'sql connection string here')]",  
      "type": "SQLAzure"  
    }  
  }  
}  
]  
}
```

No código anterior, um recurso do tipo **Microsoft.Web/sites** é definido e tem uma dependência do plano. Ele também está usando o plano do Serviço de Aplicativo e é associado a ele usando o **serverFarmId**. Ele ainda declara uma cadeia de conexão que pode ser usada para se conectar ao SQL Server.

Esta seção mostrou um exemplo de como criar uma solução PaaS no Azure usando um modelo do ARM. Da mesma forma, outras soluções PaaS, incluindo aplicativos do Azure Functions, Serviço de Kubernetes e Service Fabric, entre muitas outras, podem ser criadas usando modelos do ARM.

## Soluções relacionadas a dados usando modelos do ARM

Há muitos recursos no Azure relacionados ao gerenciamento e armazenamento de dados. Alguns dos recursos importantes relacionados a dados incluem o SQL do Azure, o Azure Cosmos DB, o Azure Data Lake Storage, o Data Lake Analytics, o Azure Synapsis, o Databricks e o Data Factory.

Todos esses recursos podem ser provisionados e configurados usando um modelo do ARM. Nesta seção, criaremos um modelo do ARM para provisionar um recurso do Data Factory responsável pela migração de dados do Armazenamento de Blobs do Azure para o Banco de Dados SQL do Azure usando procedimentos armazenados.

Você encontrará o arquivo de parâmetros juntamente com o modelo. Esses valores podem mudar de uma implantação para outra. Manteremos o modelo genérico para que você possa personalizá-lo e usá-lo facilmente com outras implantações também.

Todos os códigos desta seção podem ser encontrados em <https://github.com/Azure/azure-quickstart-templates/blob/master/101-data-factory-blob-to-sql-copy-stored-proc>.

A primeira etapa é declarar a configuração para o data factory no modelo do ARM, conforme mostrado no código a seguir:

```
"name": "[variables('dataFactoryName')]",  
"apiVersion": "2018-06-01",  
"type": "Microsoft.DataFactory/datafactories",  
"location": "[parameters('location')]",
```

Cada data factory tem vários serviços vinculados. Esses serviços vinculados atuam como conectores para obter dados no data factory, ou o data factory pode enviar dados para eles. A seguinte listagem de código cria um serviço vinculado para a conta de armazenamento do Azure na qual os blobs serão lidos para o data factory e outro serviço vinculado para o Banco de Dados SQL do Azure:

```
{  
  "type": "linkedservices",  
  "name": "[variables('storageLinkedServiceName')]",  
  "apiVersion": "2018-06-01",  
  "dependsOn": [  
    "[variables('dataFactoryName')]"  
  ],  
  "properties": {  
    "type": "AzureStorage",  
    "description": "Azure Storage Linked Service",  
    "typeProperties": {  
      "connectionString":  
        "[concat('DefaultEndpointsProtocol=https;  
        AccountName=', parameters('storageAccountName'), ',  
        AccountKey=', parameters('storageAccountKey'))]"  
    },  
    }  
  },  
  {  
    "type": "linkedservices",
```

```
"name": "[variables('sqlLinkedServiceName')]",  
"apiVersion": "2018-06-01",  
"dependsOn": [  
    "[variables('dataFactoryName')]"  
],  
"properties": {  
    "type": "AzureSqlDatabase",  
    "description": "Azure SQL linked service",  
    "typeProperties": {  
        "connectionString": "[concat('Data Source=tcp:', parameters('sqlServerName'),  
        '.database.windows.net,1433;Initial Catalog=', parameters('sqlDatabaseName'),  
        ';Integrated Security=False;User ID=', parameters('sqlUserId'), ', Password=',  
        parameters('sqlPassword'), ', Connect Timeout=30;Encrypt=True')]"  
    }  
},  
},
```

Após os serviços vinculados, é hora de definir os conjuntos de dados para o Azure Data Factory. Os conjuntos de dados ajudam a identificar os dados que devem ser lidos e colocados no data factory. Eles também podem representar os dados temporários que precisam ser armazenados pelo data factory durante a transformação ou até mesmo o local de destino onde os dados serão gravados. O próximo bloco de código cria três conjuntos de dados, um para cada um dos aspectos dos conjuntos de dados que acabaram de ser mencionados.

O conjunto de dados de leitura é mostrado no seguinte bloco de código:

```
{  
    "type": "datasets",  
    "name": "[variables('storageDataset')]",  
    "dependsOn": [  
        "[variables('dataFactoryName')]",  
        "[variables('storageLinkedServiceName')]"  
    ],  
    "apiVersion": "2018-06-01",  
    "properties": {  
        "type": "AzureBlob",  
        "linkedServiceName": "[variables('storageLinkedServiceName')]"  
    },  
    "typeProperties": {}  
}
```

```
"typeProperties": {  
    "folderPath": "[concat(parameters('sourceBlobContainer'), '/')]",  
    "fileName": "[parameters('sourceBlobName')]",  
    "format": {  
        "type": "TextFormat"  
    },  
    "availability": {  
        "frequency": "Hour",  
        "interval": 1  
    },  
    "external": true  
},  
},
```

O conjunto de dados intermediário é mostrado nas seguintes linhas de código:

```
{  
    "type": "datasets",  
    "name": "[variables('intermediateDataset')]",  
    "dependsOn": [  
        "[variables('dataFactoryName')]",  
        "[variables('sqlLinkedServiceName')]"  
    ],  
    "apiVersion": "2018-06-01",  
    "properties": {  
        "type": "AzureSqlTable",  
        "linkedServiceName": "[variables('sqlLinkedServiceName')]",  
        "typeProperties": {  
            "tableName": "[variables('intermediateDataset')]"  
        },  
        "availability": {  
            "frequency": "Hour",  
            "interval": 1  
        }  
    }  
}
```

```
    }
}
},
```

Por fim, o conjunto de dados usado para o destino é mostrado aqui:

```
{
  "type": "datasets",
  "name": "[variables('sqlDataset')]",
  "dependsOn": [
    "[variables('dataFactoryName')]",
    "[variables('sqlLinkedServiceName')]"
  ],
  "apiVersion": "2018-06-01",
  "properties": {
    "type": "AzureSqlTable",
    "linkedServiceName": "[variables('sqlLinkedServiceName')]",
    "typeProperties": {
      "tableName": "[parameters('sqlTargetTable')]"
    },
    "availability": {
      "frequency": "Hour",
      "interval": 1
    }
  }
},
```

Por fim, precisamos de um pipeline no data factory que possa reunir todos os conjuntos de dados e serviços vinculados e ajudar na criação de soluções de dados de extração, transação e carregamento. Um pipeline consiste em várias atividades, cada uma cumprindo uma tarefa específica. Todas essas atividades podem ser definidas no modelo do ARM, como você verá agora. A primeira atividade copia os blobs na conta de armazenamento para um SQL Server intermediário, conforme mostrado no código a seguir:

```
{
  "type": "dataPipelines",
  "name": "[variables('pipelineName')]",
  "dependsOn": [
```

```
"[variables('dataFactoryName')]",
"[variables('storageLinkedServiceName')]",
"[variables('sqlLinkedServiceName')]",
"[variables('storageDataset')]",
"[variables('sqlDataset')]"
],
"apiVersion": "2018-06-01",
"properties": {
"description": "Copies data from Azure Blob to Sql DB while invoking stored procedure",
"activities": [
{
"name": "BlobtoSqlTableCopyActivity",
"type": "Copy",
"typeProperties": {
"source": {
"type": "BlobSource"
},
"sink": {
"type": "SqlSink",
"writeBatchSize": 0,
"writeBatchTimeout": "00:00:00"
}
},
"inputs": [
{
"name": "[variables('storageDataset')]"
}
],
"outputs": [
{
"name": "[variables('intermediateDataset')]"
}
```

```
        }
    ],
},
{
"name": "SqlTabletoSqlDbSprocActivity",
"type": "SqlServerStoredProcedure",
"inputs": [
    {
        "name": "[variables('intermediateDataset')]"
    }
],
"outputs": [
    {
        "name": "[variables('sqlDataset')]"
    }
],
"typeProperties": {
    "storedProcedureName": "[parameters('sqlWriterStoredProcName')]"
},
"scheduler": {
    "frequency": "Hour",
    "interval": 1
},
"policy": {
    "timeout": "02:00:00",
    "concurrency": 1,
    "executionPriorityOrder": "NewestFirst",
    "retry": 3
}
},
],
],
```

```
"start": "2020-10-01T00:00:00Z",
"end": "2020-10-02T00:00:00Z"
}
}
]
}
```

A última atividade copia os dados do conjunto de dados intermediário para o conjunto de dados de destino final.

Há também as horas de início e término durante as quais o pipeline devem estar em execução.

Esta seção se concentrou na criação de um modelo do ARM para uma solução relacionada a dados. Na próxima seção, lidaremos com modelos do ARM para a criação de datacenters no Azure com o Active Directory e o DNS.

## Criar uma solução IaaS no Azure com o Active Directory e o DNS

Criar uma solução IaaS no Azure significa criar várias máquinas virtuais, promover uma máquina virtual para ser um controlador de domínio e fazer com que outras máquinas virtuais sejam associadas ao controlador de domínio como nós associados ao domínio. Isso também significa instalar um servidor DNS para a resolução de nomes e, opcionalmente, um servidor de salto para acessar essas máquinas virtuais com segurança.

O modelo cria uma floresta do Active Directory nas máquinas virtuais. Ele cria várias máquinas virtuais com base nos parâmetros fornecidos.

O modelo cria:

- Alguns conjuntos de disponibilidade
- Uma rede virtual
- Grupos de segurança de rede para definir as portas e endereços IP permitidos e não permitidos

Em seguida, o modelo faz o seguinte:

- Provisiona um ou dois domínios. O domínio raiz é criado por padrão; o domínio filho é opcional
- Provisiona dois controladores de domínio por domínio
- Executa os scripts de configuração de estado desejados para promover uma máquina virtual para ser um controlador de domínio

Podemos criar várias máquinas virtuais usando a abordagem abordada na seção *Soluções de máquina virtual usando modelos do ARM*. No entanto, essas máquinas virtuais deverão fazer parte de um conjunto de disponibilidade se precisarem estar altamente disponíveis. Observe que os conjuntos de disponibilidade fornecem 99,95% de disponibilidade para aplicações implantadas nessas máquinas virtuais, enquanto as zonas de disponibilidade fornecem 99,99% de disponibilidade.

Um conjunto de disponibilidade pode ser configurado conforme mostrado no código a seguir:

```
{  
  "name": "[variables('adAvailabilitySetNameRoot')]",  
  "type": "Microsoft.Compute/availabilitySets",  
  "apiVersion": "2019-07-01",  
  "location": "[parameters('location')]",  
  "sku": {  
    "name": "Aligned"  
  },  
  "properties": {  
    "PlatformUpdateDomainCount": 3,  
    "PlatformFaultDomainCount": 2  
  }  
},
```

Quando o conjunto de disponibilidade é criado, um perfil adicional deve ser adicionado à configuração da máquina virtual para associar a máquina virtual ao conjunto de disponibilidade, conforme mostrado no código a seguir:

```
"availabilitySet" : {  
  "id": "[resourceId('Microsoft.Compute/availabilitySets',  
  parameters('adAvailabilitySetName'))]"  
}
```

Você deve observar que os conjuntos de disponibilidade são obrigatórios para usar平衡adores de carga com máquinas virtuais.

Outra alteração necessária na configuração da rede virtual é a adição de informações de DNS, conforme mostrado no código a seguir:

```
{  
  "name": "[parameters('virtualNetworkName')]",  
  "type": "Microsoft.Network/virtualNetworks",  
  "location": "[parameters('location')]",  
  "apiVersion": "2019-09-01",  
  "properties": {  
    "addressSpace": {  
      "addressPrefixes": [  
        "[parameters('virtualNetworkAddressRange')]"  
      ]  
    },  
    "dhcpOptions": {  
      "dnsServers": "[parameters('DNSServerAddress')]"  
    },  
    "subnets": [  
      {  
        "name": "[parameters('subnetName')]",  
        "properties": {  
          "addressPrefix": "[parameters('subnetRange')]"  
        }  
      }  
    ]  
  },  
},
```

Por fim, para converter uma máquina virtual no Active Directory, um script do PowerShell ou um script de **Desired State Configuration (DSC)** deve ser executado na máquina virtual. Mesmo para associar outras máquinas virtuais ao domínio, outro conjunto de scripts deve ser executado nessas máquinas virtuais.

Os scripts podem ser executados na máquina virtual usando o recurso **CustomScriptExtension**, conforme mostrado no código a seguir:

```
{  
  "type": "Microsoft.Compute/virtualMachines/extensions",  
  "name": "[concat(parameters('adNextDCVMName'), '/PrepareNextDC')]",  
  "apiVersion": "2018-06-01",  
  "location": "[parameters('location')]",  
  "properties": {  
    "publisher": "Microsoft.Powershell",  
    "type": "DSC",  
    "typeHandlerVersion": "2.21",  
    "autoUpgradeMinorVersion": true,  
    "settings": {  
      "modulesURL": "[parameters('adNextDCConfigurationModulesURL')]",  
      "configurationFunction": "[parameters('adNextDCConfigurationFunction')]",  
      "properties": {  
        "domainName": "[parameters('domainName')]",  
        "DNSServer": "[parameters('DNSServer')]",  
        "DNSForwarder": "[parameters('DNSServer')]",  
        "adminCreds": {  
          "userName": "[parameters('adminUserName')]",  
          "password": "privateSettingsRef:adminPassword"  
        }  
      }  
    },  
    "protectedSettings": {  
      "items": {  
        "adminPassword": "[parameters('adminPassword')]"  
      }  
    }  
  },  
},
```

Nesta seção, criamos um datacenter no Azure usando o paradigma de IaaS. Criamos várias máquinas virtuais e transformamos uma delas no controlador de domínio, instalamos o DNS e atribuímos um domínio a ela. Agora, outras máquinas virtuais na rede podem ser associadas a esse domínio e podem formar um datacenter completo no Azure.

Consulte <https://github.com/Azure/azure-quickstart-templates/tree/master/301-create-ad-forest-with-subdomain> para obter a listagem completa de códigos para criar um datacenter no Azure.

## Resumo

A opção de implantar recursos usando uma única implantação para várias assinaturas, grupos de recursos e regiões proporciona melhor capacidade de implantar, reduzir bugs na implantação e acessar benefícios avançados, como criar sites de Disaster Recovery e alcançar alta disponibilidade.

Neste capítulo, você viu como criar alguns tipos diferentes de solução usando modelos do ARM. Isso incluiu a criação de uma solução baseada em infraestrutura que inclui máquinas virtuais; uma solução baseada em plataformas usando o Serviço de Aplicativo do Azure; uma solução relacionada a dados usando o recurso Data Factory (incluindo sua configuração); e um datacenter no Azure com máquinas virtuais, o Active Directory e o DNS instalados sobre a máquina virtual.

No próximo capítulo, vamos nos concentrar na criação de modelos do ARM modulares, uma habilidade essencial para os arquitetos que realmente querem levar seus modelos do ARM para o próximo nível. O capítulo também mostrará várias maneiras de desenvolver modelos do ARM e criar modelos do ARM reutilizáveis e modulares.



# 16

## Design modular e implementação de modelos do ARM

Sabemos que há várias maneiras de criar um modelo do **Azure Resource Manager (ARM)**. É muito fácil criar um que provisione todos os recursos necessários no Azure usando o Visual Studio e o Visual Studio Code. Um único modelo do ARM pode consistir em todos os recursos necessários para uma solução no Azure. Esse modelo pode ter apenas alguns recursos ou ser maior e conter muitos recursos.

Enquanto a criação de um único modelo que consiste em todos os recursos é muito tentadora, é aconselhável planejar com antecedência uma implementação do modelo do ARM dividida em vários modelos do ARM menores para que problemas futuros relacionados a eles possam ser evitados.

Neste capítulo, veremos como escrever modelos do ARM de maneira modular para que eles evoluam ao longo de um período com o mínimo de envolvimento em termos de alterações e esforço em testes e implantação.

No entanto, antes de escrever modelos modulares, é melhor entender os problemas resolvidos escrevendo-os de forma modular.

Os tópicos a seguir serão abordados neste capítulo:

- Problemas com um único modelo
- Entender a implantação aninhada e vinculada
- Modelos vinculados
- Modelos aninhados
- Configurações de fluxo livre
- Configurações conhecidas

Agora, vamos explorar em detalhes os tópicos mencionados anteriormente, o que ajudará você a criar modelos modulares usando as práticas recomendadas da indústria.

## **Problemas com a abordagem de um único modelo**

Superficialmente, pode não parecer que um único modelo grande que consiste em todos os recursos terá problemas, mas eles poderão surgir no futuro. Vamos entender os problemas que podem surgir com modelos grandes únicos.

### **Flexibilidade reduzida na alteração de modelos**

O uso de um único modelo grande com todos os recursos dificulta sua alteração no futuro. Com todas as dependências, parâmetros e variáveis em um único modelo, a alteração dele pode levar um tempo considerável em comparação com modelos menores. A alteração pode ter um impacto em outras seções do modelo, que podem passar despercebidas, bem como introduzir bugs.

### **Solucionar problemas com modelos grandes**

É difícil solucionar problemas com modelos grandes. Esse é um fato conhecido. Quanto maior o número de recursos em um modelo, mais difícil será solucionar os problemas dele. Um modelo implanta todos os recursos nele, e encontrar um bug envolve a implantação do modelo com bastante frequência. Os desenvolvedores reduzem a produtividade enquanto aguardam a conclusão da implantação do modelo.

Além disso, a implantação de um único modelo é mais demorada do que implantar modelos menores. Os desenvolvedores precisam aguardar a implantação de recursos que contenham erros antes de tomar qualquer medida.

## Abuso de dependência

As dependências entre recursos também tendem a se tornar mais complexas em modelos maiores. É muito fácil abusar do uso do recurso `dependsOn` em modelos do ARM devido à maneira como eles funcionam. Cada recurso em um modelo pode se referir a todos os seus recursos anteriores, em vez de criar uma árvore de dependências. Os modelos do ARM não reclamarão se um único recurso depender de todos os outros no modelo do ARM, mesmo que esses outros recursos possam ter interdependências. Isso torna a mudança de modelos do ARM propensa a bugs e, às vezes, nem sequer é possível alterá-los.

## Agilidade reduzida

Geralmente, há várias equipes em um projeto, cada uma com seus próprios recursos no Azure. Essas equipes terão dificuldades para trabalhar com um único modelo do ARM porque um único desenvolvedor deve atualizá-lo. A atualização de um único modelo com várias equipes pode causar conflitos e mesclagens difíceis de resolver. Com vários modelos menores, cada equipe pode criar seu próprio modelo do ARM.

## Sem reutilização

Não há problema se você tem um único modelo, e o uso dele implica na implantação de todos os recursos. Não há nenhuma possibilidade de selecionar recursos individuais sem algumas manobras, como a adição de recursos condicionais. Um único modelo grande perde a reutilização, pois ou você implanta todos os recursos ou não implanta nenhum deles.

Sabendo que modelos grandes únicos têm tantos problemas, é recomendável criar modelos modulares para ter benefícios como os seguintes:

- Várias equipes podem trabalhar em seus próprios modelos isoladamente.
- Os modelos podem ser reutilizados em projetos e soluções.
- É fácil depurar e solucionar problemas dos modelos.

Agora que abordamos alguns dos problemas com modelos grandes e únicos, na próxima seção, consideraremos o ponto crucial dos modelos modulares e como eles podem ajudar os desenvolvedores a implementar implantações eficientes.

## Entender o princípio da responsabilidade única

O **princípio da responsabilidade única** é um dos princípios fundamentais dos princípios de design sólidos. Ele informa que um segmento de classe ou código deve ser responsável por uma única função e deve possuir essa funcionalidade completamente. O código deve ser alterado ou evoluído somente se houver uma alteração funcional ou um bug na funcionalidade atual, e não o contrário. Esse código não deve ser alterado devido a mudanças em algum outro componente ou código que não faz parte do componente atual.

Aplicar o mesmo princípio aos modelos do ARM nos ajuda a criar modelos que têm a única responsabilidade de implantar um único recurso ou funcionalidade, em vez de implantar todos os recursos e uma solução completa.

O uso desse princípio ajudará você a criar vários modelos, cada um responsável por um único recurso ou um grupo menor de recursos, e não por todos os recursos.

### Solução de problemas e depuração mais rápidas

Cada implantação de modelo é uma atividade distinta no Azure e é uma instância separada que consiste em entradas, saídas e logs. Quando vários modelos são implantados para uma solução, cada implantação de modelo tem entradas de log separadas, além de suas descrições de entrada e saída. É muito mais fácil isolar bugs e solucionar problemas usando esses logs independentes de várias implantações, em comparação com um único modelo grande.

### Modelos modulares

Quando um único modelo grande é decomposto em vários modelos em que cada modelo menor cuida de seus próprios recursos, e esses recursos são exclusivamente mantidos e são de propriedade e responsabilidade do modelo que os contém, podemos dizer que temos modelos modulares. Cada modelo dentro desses modelos segue o princípio da responsabilidade única.

Antes de aprender a dividir um modelo grande em vários modelos reutilizáveis menores, é importante entender a tecnologia por trás da criação de modelos menores e como criá-los para implantar soluções completas.

## Recursos de implantação

O ARM facilita a vinculação de modelos. Embora já tenhamos visto os modelos vinculados em detalhes, vou mencioná-los aqui para ajudar você a entender como os modelos de vinculação ajudam a alcançar a modularidade, a composição e a reutilização.

Os modelos do ARM fornecem recursos especializados conhecidos como **implantações**, que estão disponíveis no namespace **Microsoft.Resources**. Um recurso de implantações em um modelo do ARM tem aparência muito semelhante à do segmento de código a seguir:

```
"resources": [
  {
    "apiVersion": "2019-10-01",
    "name": "linkedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
      "mode": "Incremental",
      <nested-template-or-external-template>
    }
  }
]
```

Esse modelo é auto-explicativo, e as duas configurações mais importantes no recurso do modelo são o tipo e as propriedades. O tipo aqui refere-se ao recurso de implantações, e não a qualquer recurso específico do Azure (armazenamento, máquina virtual e assim por diante), e as propriedades especificam a configuração de implantação, incluindo uma implantação de modelo vinculado ou uma implantação de modelo aninhada.

No entanto, o que o recurso de implantações faz? A função de um recurso de implantações é implantar outro modelo. Outro modelo pode ser um externo em um arquivo de modelo do ARM separado ou um modelo aninhado. Portanto, é possível invocar outros modelos de um modelo, assim como uma chamada de função.

Pode haver níveis aninhados de implantações em modelos do ARM. Isso significa que um único modelo pode chamar outro, e o modelo chamado pode chamar outro, e isso pode continuar por cinco níveis de chamadas aninhadas.

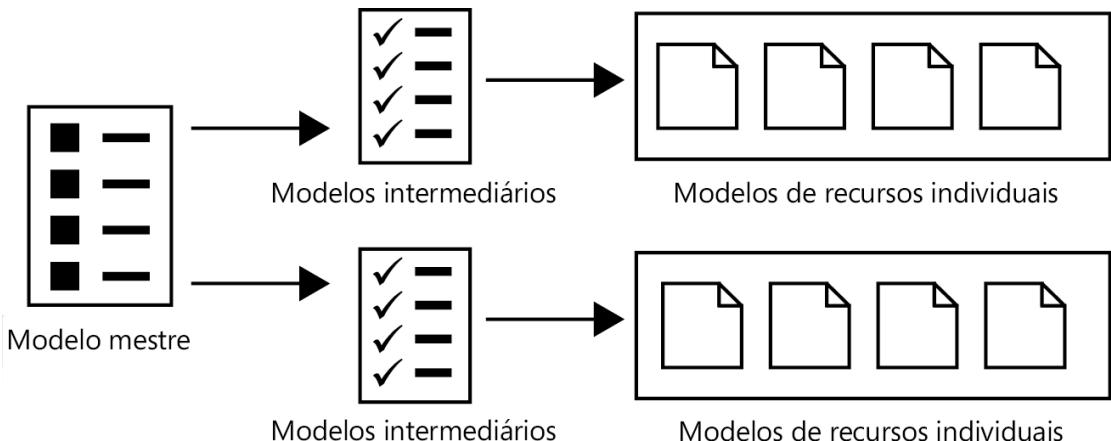


Figura 16.1: decomposição de modelo em modelos menores

Agora que entendemos que os modelos grandes podem ser modulares com recursos separados em modelos diferentes, precisamos vinculá-los e reuni-los para implantar recursos no Azure. Os modelos vinculados e aninhados são maneiras de compor vários modelos juntos.

## Modelos vinculados

Os modelos vinculados são modelos que invocam modelos externos. Os modelos externos são armazenados em arquivos de modelo do ARM diferentes. Um exemplo de modelos vinculados é:

```

"resources": [
  {
    "apiVersion": "2019-10-01",
    "name": "linkedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
      "mode": "Incremental",
      "templateLink": {
        "uri": "https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAccount.json",
        "contentVersion": "1.0.0.0"
      },
      "parametersLink": {
        "uri": "https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAccount.parameters.json",
        "contentVersion": "1.0.0.0"
      }
    }
  }
]
  
```

```
        }
    }
}
]
```

Propriedades adicionais importantes neste modelo em comparação com o modelo anterior são **templateLink** e **parametersLink**. Agora, **templateLink** refere-se à URL real do local do arquivo de modelo externo, e **parametersLink** é o local da URL do arquivo **parameters** correspondente. É importante lembrar que o modelo do chamador deve ter direitos de acesso para o local do modelo chamado. Por exemplo, se os modelos externos são armazenados no Armazenamento de Blobs do Azure, que é protegido por chaves, as chaves de **Assinatura de Acesso Seguro (SAS)** apropriadas devem estar disponíveis para que o modelo do chamador possa acessar os modelos vinculados.

Também é possível fornecer parâmetros em linha explícitos em vez do valor **parametersLink**, conforme mostrado aqui:

```
"resources": [
{
    "apiVersion": "2019-10-01",
    "name": "linkedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
        "mode": "Incremental",
        "templateLink": {
            "uri": "https://mystorageaccount.blob.core.windows.net/AzureTemplates/newStorageAccount.json",
            "contentVersion": "1.0.0.0"
        },
        "parameters": {
            "StorageAccountName": {"value": "
                [parameters('StorageAccountName')]"
        }
    }
}
]
```

Agora você tem uma boa compreensão dos modelos vinculados. Um tópico intimamente relacionado é modelos aninhados, que a próxima seção abordará detalhadamente.

## Modelos aninhados

Os modelos aninhados são um recurso relativamente novo em modelos do ARM em comparação com modelos vinculados externos.

Os modelos aninhados não definem recursos em arquivos externos. Os recursos são definidos no próprio modelo do chamador e no recurso de implantações, conforme mostrado aqui:

```
"resources": [
  {
    "apiVersion": "2019-10-01",
    "name": "nestedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
      "mode": "Incremental",
      "template": {
        "$schema": "https://schema.management.azure.com/schemas/2015-
          01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "resources": [
          {
            "type": "Microsoft.Storage/storageAccounts",
            "name": "[variables('storageName')]",
            "apiVersion": "2019-04-01",
            "location": "West US",
            "properties": {
              "accountType": "Standard_LRS"
            }
          }
        ]
      }
    }
  }
]
```

Neste segmento de código, podemos ver que o recurso de conta de armazenamento está aninhado dentro do modelo original como parte do recurso de implementações. Em vez de usar os atributos **templateLink** e **parametersLink**, uma matriz **resources** é usada para criar vários recursos como parte de uma única implantação. A vantagem de usar uma implantação aninhada é que os recursos dentro de um pai podem ser usados para reconfigurá-los usando seus nomes. Normalmente, um recurso com um nome pode existir apenas uma vez dentro de um modelo. Os modelos aninhados nos permitem usá-los dentro do mesmo modelo e garantir que todos os modelos sejam autossuficientes, em vez de serem armazenados separadamente, e eles podem ou não estar acessíveis para esses arquivos externos.

Agora que entendemos a tecnologia por trás dos modelos do ARM modulares, como devemos dividir um modelo grande em modelos menores?

Há várias maneiras como um modelo grande pode ser decomposto em modelos menores. A Microsoft recomenda o seguinte padrão para a decomposição de modelos do ARM:

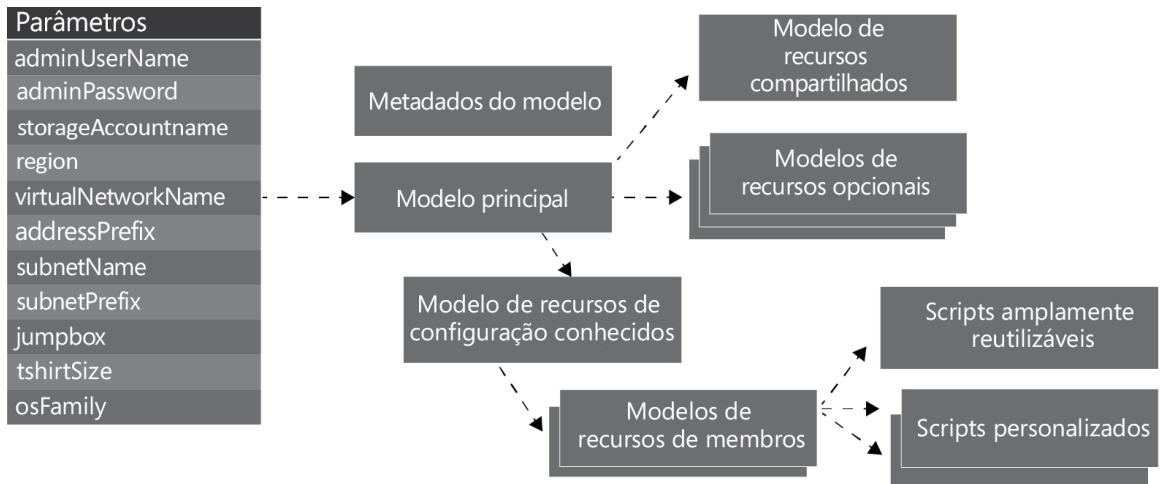


Figura 16.2: estratégia de decomposição de modelos

Ao decompor um modelo grande em modelos menores, há sempre o modelo principal, que é usado para implantar a solução. Esse modelo principal ou mestre invoca internamente outros modelos aninhados ou vinculados e eles, por sua vez, invocam outros modelos. Por fim, os modelos que contêm recursos do Azure são implantados.

O modelo principal pode invocar um modelo de recurso de configuração conhecido que, por sua vez, invocará os modelos que compõem os recursos do Azure. O modelo de recurso de configuração conhecido é específico de um projeto ou solução e não tem muitos fatores reutilizáveis associados a ele. Os modelos de recurso de membro são modelos reutilizáveis invocados pelo modelo de recurso de configuração conhecido.

Opcionalmente, o modelo mestre pode invocar modelos de recursos compartilhados e outros modelos de recursos, se existirem.

É importante entender as configurações conhecidas. Os modelos podem ser criados como configurações conhecidas ou de fluxo livre.

## Configurações de fluxo livre

Os modelos do ARM podem ser criados como modelos genéricos em que a maioria (se não todos) dos valores atribuídos às variáveis é obtida como parâmetros. Isso permite que a pessoa que usa o modelo transmita qualquer valor que considere necessário para implantar recursos no Azure. Por exemplo, a pessoa que está implantando o modelo pode escolher uma máquina virtual de qualquer tamanho, qualquer número de máquinas virtuais e qualquer configuração para seu armazenamento e redes. Ela é conhecida como configuração de fluxo livre, onde a maior parte da configuração é permitida e os modelos são provenientes do usuário, e não declarados no modelo.

Há desafios nesse tipo de configuração. O maior deles é que não há suporte para todas as configurações em todos os datacenters e regiões do Azure. Os modelos falharão ao criar recursos se eles não tiverem permissão para serem criados em regiões ou locais específicos. Outro problema com a configuração de fluxo livre é que os usuários podem fornecer qualquer valor que considerem necessário e um modelo os atenderá, aumentando assim o custo e a pegada de implantação, mesmo que eles não sejam completamente necessários.

## Configurações conhecidas

Por outro lado, configurações conhecidas são configurações pré-determinadas específicas para implantar um ambiente usando modelos do ARM. Essas configurações pré-determinadas são conhecidas como **configurações de dimensionamento**.

**t-shirt.** Semelhante ao modo como um t-shirt está disponível em uma configuração pré-determinada, como pequenas, médias e grandes, os modelos do ARM podem ser pré-configurados para implantar um ambiente pequeno, médio ou grande, dependendo dos requisitos. Isso significa que os usuários não podem determinar qualquer tamanho personalizado aleatório para o ambiente, mas podem escolher entre várias opções fornecidas, e os modelos do ARM executados durante o tempo de execução garantirão que uma configuração apropriada do ambiente seja fornecida.

Assim, a primeira etapa da criação de um modelo do ARM modular é decidir quais serão as configurações conhecidas de um ambiente.

Como exemplo, veja a seguir a configuração de uma implantação de datacenter no Azure:

Tamanho da camiseta	Configuração do modelo do ARM
Pequena	4 máquinas virtuais com 7 GB de memória, além de 4 núcleos de CPU
Média	8 máquinas virtuais com 14 GB de memória, além de 8 núcleos de CPU
Grande	16 máquinas virtuais com 28 GB de memória, além de 8 núcleos de CPU

Tabela 16.1: configuração de uma implantação de datacenter no Azure

Agora que conhecemos as configurações, podemos criar modelos do ARM modulares.

Há duas maneiras de escrever modelos do ARM modulares:

- **Modelos compostos:** modelos compostos vinculados a outros modelos. Exemplos de modelos compostos são os modelos mestre e intermediário.
- **Modelos de nível folha:** são modelos que contêm um único recurso do Azure.

Os modelos do ARM podem ser divididos em modelos modulares com base em:

- Tecnologia
- Funcionalidade

Uma maneira ideal de decidir qual método modular será usado para criar um modelo do ARM é:

- Defina modelos de nível folha ou recursos que consistem em recursos únicos. No diagrama a seguir, os modelos de extrema direita são modelos de nível folha. No diagrama, as máquinas virtuais, a rede virtual, o armazenamento e outros na mesma coluna representam modelos de nível folha.
- Componha modelos específicos do ambiente usando os modelos de nível folha. Esses modelos específicos do ambiente fornecem um ambiente do Azure, como um ambiente do SQL Server, de Serviço de Aplicativo ou de datacenter. Vamos nos aprofundar um pouco mais neste tópico. Vamos ver o exemplo de um ambiente de SQL do Azure. Para criar um ambiente de SQL do Azure, são necessários vários recursos. No mínimo, um SQL Server lógico, um banco de dados SQL e alguns recursos de firewall do SQL devem ser provisionados. Todos esses recursos são definidos em modelos individuais ao nível folha. Esses recursos podem ser compostos em conjunto em um único modelo que tem a capacidade de criar um ambiente SQL do Azure. Qualquer pessoa que queira criar um ambiente SQL pode usar esse modelo composto. A Figura 16.3 tem **Datacenter**, **Mensagens** e **Serviço de Aplicativo** como modelos específicos do ambiente.
- Crie modelos com maior abstração compondo vários modelos específicos do ambiente em soluções. Esses modelos são compostos por modelos específicos do ambiente que foram criados na etapa anterior. Por exemplo, para criar uma solução de inventário de comércio eletrônico que precise de um ambiente de Serviço de Aplicativo e um ambiente SQL, dois modelos de ambiente, Serviço de Aplicativo e SQL Server podem ser compostos em conjunto. A Figura 16.3 tem os modelos **Funcional 1** e **Funcional 2**, que são compostos por modelos filhos.
- Por fim, um modelo mestre deve ser criado e deve ser composto por vários modelos em que cada um deles pode implantar uma solução.

As etapas anteriores para criar um modelo projetado modular podem ser facilmente entendidas por meio da Figura 16.3:

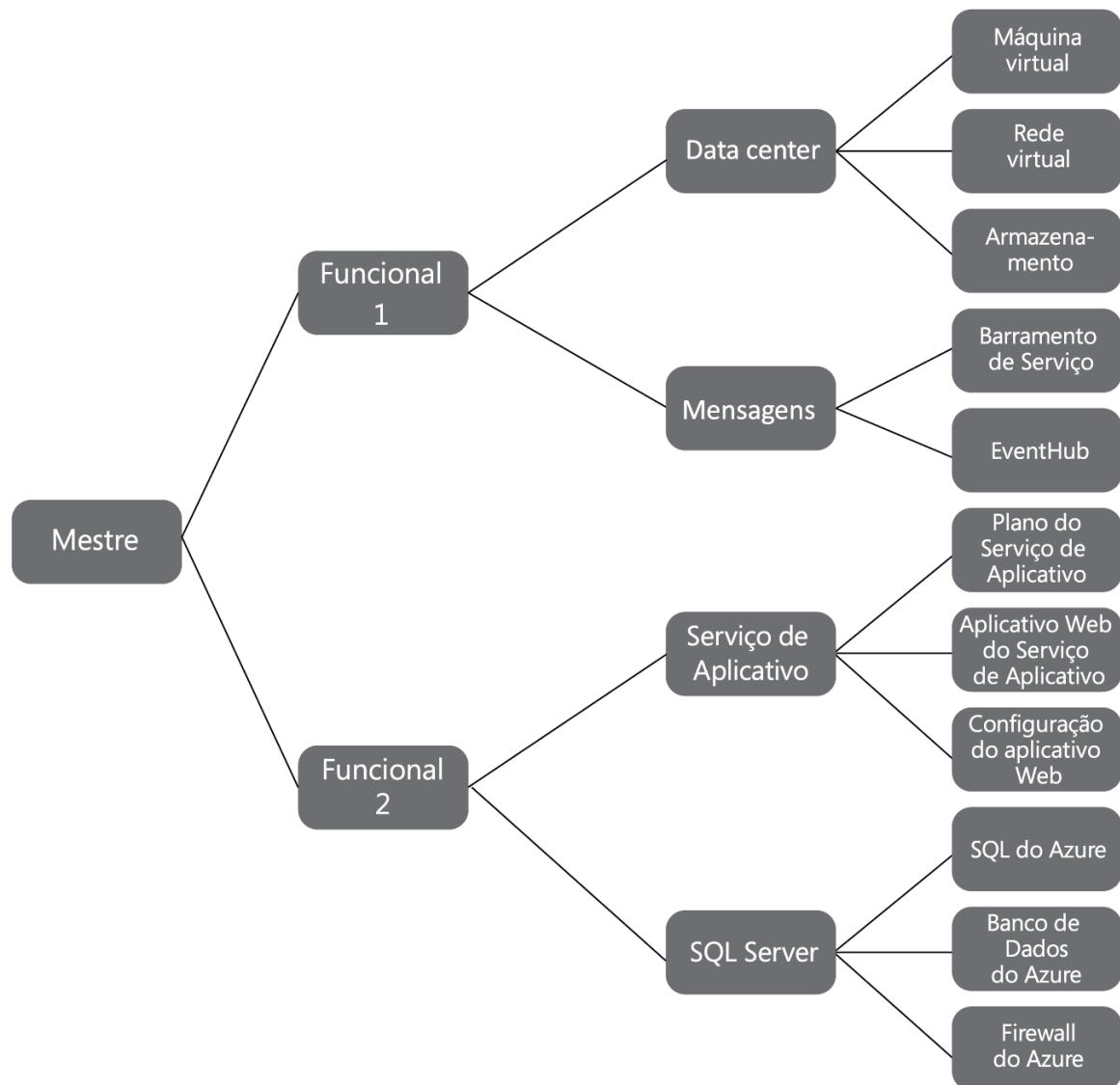


Figura 16.3: mapeamento de modelos e recursos

Agora, vamos implementar uma parte da funcionalidade mostrada no diagrama anterior. Nesta implementação, forneceremos uma máquina virtual com uma extensão de script usando uma abordagem modular. A extensão de script personalizada implanta binários do Docker e prepara um ambiente de contêiner em uma máquina virtual do Windows Server 2016.

Agora, vamos criar uma solução usando modelos do ARM com uma abordagem modular. Como mencionado anteriormente, a primeira etapa é criar modelos de recursos individuais. Tais modelos serão usados para compor modelos adicionais capazes de criar um ambiente. Esses modelos serão necessários para criar uma máquina virtual. Todos os modelos do ARM mostrados aqui estão disponíveis no código do capítulo que os acompanha. Os nomes e o código desses modelos são:

- **Storage.json**
- **virtualNetwork.json**
- **PublicIPAddress.json**
- **NIC.json**
- **VirtualMachine.json**
- **CustomScriptExtension.json**

Primeiro, vejamos o código do modelo **Storage.json**. Esse modelo fornece uma conta de armazenamento de que todas as máquinas virtuais precisam para armazenar seus arquivos de sistema operacional e disco de dados:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "storageAccountName": {  
            "type": "string",  
            "minLength": 1  
        },  
        "storageType": {  
            "type": "string",  
            "minLength": 1  
        },  
        ...  
    },  
    "outputs": {  
        "resourceDetails": {  
            "type": "object",  
            "value": "[reference(parameters('storageAccountName'))]"  
        }  
    }  
}
```

Em seguida, veremos o código do modelo de endereço IP público. Uma máquina virtual que deve ser acessível pela Internet precisa de um recurso de endereço IP público atribuído à sua placa de interface de rede. Embora a exposição de uma máquina virtual à Internet seja opcional, esse recurso pode ser usado para criar uma máquina virtual. O código a seguir está disponível no arquivo **PublicIPAddress.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "publicIPName": {  
            "type": "string",  
            "minLength": 1  
        },  
        "publicIPType": {  
            "type": "string",  
            "minLength": 1  
        },  
        ...  
    },  
    "outputs": {  
        "resourceDetails": {  
            "type": "object",  
            "value": "[reference(parameters('publicIPName'))]"  
        }  
    }  
}
```

Em seguida, veremos o código da rede virtual. As máquinas virtuais no Azure precisam de uma rede virtual para comunicação. Esse modelo será usado para criar uma rede virtual no Azure com um intervalo de endereços pré-definidos e sub-redes. O código a seguir está disponível no arquivo **virtualNetwork.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "virtualNetworkName": {  
            "type": "string",  
            "minLength": 1  
        ...  
    },  
        "subnetPrefix": {  
            "type": "string",  
            "minLength": 1  
        },  
        "resourceLocation": {  
            "type": "string",  
            "minLength": 1  
        }  
    ...  
        "subnets": [  
            {  
                "name": "[parameters('subnetName')]",  
                "properties": {  
                    "addressPrefix": "[parameters('subnetPrefix')]"  
                }  
            }  
        ]  
    }  
},  
    "outputs": {  
        "resourceDetails": {  
            "type": "object",  
            "value": "[reference(parameters('virtualNetworkName'))]"  
        }  
    }  
}
```

Em seguida, veremos o código da placa da interface de rede. Uma placa de rede virtual é necessária para que uma máquina virtual se conecte a uma rede virtual, além de aceitar e enviar solicitações da Internet e para ela. O código a seguir está disponível no arquivo **NIC.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "nicName": {  
            "type": "string",  
            "minLength": 1  
        },  
        "publicIpReference": {  
            "type": "string",  
            "minLength": 1  
        }  
    },  
    "...  
    [resourceId(subscription().subscriptionId,resourceGroup().name, 'Microsoft.  
Network/publicIPAddresses', parameters('publicIpReference'))]",  
    "vnetRef": "[resourceId(subscription().  
subscriptionId,resourceGroup().name, 'Microsoft.Network/virtualNetworks',  
parameters('virtualNetworkReference'))]",  
    "subnet1Ref": "[concat(variables('vnetRef'), '/subnets/',  
parameters('subnetReference'))]"  
},  
...  
        "id": "[variables('subnet1Ref')]"  
    }  
}  
}  
]  
}  
]  
}  
],  
"outputs": {  
    "resourceDetails": {  
        "type": "object",  
        "value": "[reference(parameters('nicName'))]"  
    }  
}  
}
```

Em seguida, veremos o código para criar uma máquina virtual. Cada máquina virtual é um recurso no Azure. Além disso, lembre-se de que esse modelo não tem nenhuma referência ao armazenamento, à rede, aos endereços IP públicos ou a outros recursos criados anteriormente. Essa referência e composição acontecerá posteriormente nesta seção usando outro modelo. O código a seguir está disponível no arquivo **VirtualMachine.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "vmName": {  
            "type": "string",  
            "minLength": 1  
        },  
        ...  
    },  
    "imageOffer": {  
        "type": "string",  
        "minLength": 1  
    },  
    "windowsOSVersion": {  
        "type": "string",  
        "minLength": 1  
    },  
    ...  
    "outputs": {  
        "resourceDetails": {  
            "type": "object",  
            "value": "[reference(parameters('vmName'))]"  
        }  
    }  
}
```

Em seguida, veremos o código para criar uma extensão de script personalizado. Esse recurso executa um script do PowerShell em uma máquina virtual depois que é provisionado. Esse recurso permite executar tarefas pós-provisionamento em Máquinas Virtuais do Azure. O código a seguir está disponível no arquivo **CustomScriptExtension.json**:

```
{  
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/  
deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "VMName": {  
            "type": "string",  
            "defaultValue": "sqldock",  
            "metadata": {  
                ...  
                "commandToExecute": "[concat('powershell -ExecutionPolicy  
Unrestricted -file docker.ps1')]"  
            },  
            "protectedSettings": {  
                ...  
            }  
        }  
    },  
    "outputs": {}  
}
```

Em seguida, veremos o código do PowerShell de extensão de script personalizado que prepara o ambiente do Docker. Uma reinicialização de máquina virtual pode ocorrer durante a execução do script do PowerShell, dependendo de o recurso de contêineres do Windows já estar instalado ou não. O script a seguir instala o pacote NuGet, o provedor **DockerMsftProvider** e o arquivo executável do Docker. O arquivo **docker.ps1** está disponível com o código de capítulo que o acompanha:

```
#  
# docker.ps1  
#  
Install-PackageProvider -Name Nuget -Force -ForceBootstrap -Confirm:$false  
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force  
-Confirm:$false -verboseInstall-Package -Name docker -ProviderName  
DockerMsftProvider -Force -ForceBootstrap -Confirm:$false
```

Todos os modelos vinculados anteriormente vistos devem ser carregados em um contêiner em uma conta de Armazenamento de Blobs do Azure. Esse contêiner pode ter uma política de acesso privado aplicada, como você viu no capítulo anterior. No entanto, para este exemplo, definiremos a política de acesso como **container**. Isso significa que esses modelos vinculados podem ser acessados sem um token SAS.

Por fim, vamos nos concentrar em escrever o modelo mestre. No modelo mestre, todos os modelos vinculados são compostos juntos para criar uma solução, para implantar uma máquina virtual e executar um script nela. A mesma abordagem pode ser usada para criar outras soluções, como fornecer um datacenter que consiste em várias máquinas virtuais interconectadas. O código a seguir está disponível no arquivo **Master.json**:

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "storageAccountName": {  
            "type": "string",  
            "minLength": 1  
        ...  
    },  
        "subnetName": {  
            "type": "string",  
            "minLength": 1  
        },  
        "subnetPrefix": {  
            "type": "string",  
            "minLength": 1  
        ...  
        "windowsOSVersion": {  
            "type": "string",  
            "minLength": 1  
        },  
        "vhdStorageName": {  
            "type": "string",  
            "minLength": 1  
        },  
        "vhdStorageContainerName": {  
            "type": "string",  
            "minLength": 1  
        ...[concat('https://',parameters('storageAccountName'),'armtfiles.blob.  
core.windows.net/',variables('containerName'), '/Storage.json')]",  
            "contentVersion": "1.0.0.0"
```

```
},
"parameters": {
    "storageAccountName": {
        "value": "[parameters('storageAccountName')]"
    },
    "storageType": {
        "value": "[parameters('storageType')]"
    },
    "resourceLocation": {
        "value": "[resourceGroup().location]"
    }
    ...
},
"outputs": {
    "resourceDetails": {
        "type": "object",
        "value": "[reference('GetVM').outputs.resourceDetails.value]"
    }
}
}
```

Os modelos mestre invocam os modelos externos e também coordenam interdependências entre eles.

Os modelos externos devem estar disponíveis em um local bem conhecido para que o modelo mestre possa acessá-los e invocá-los. Neste exemplo, os modelos externos são armazenados no contêiner de Armazenamento de Blobs do Azure, e essas informações foram transmitidas para o modelo do ARM por meio de parâmetros.

Os modelos externos no Armazenamento de Blobs do Azure podem ter acesso protegido por meio da configuração de políticas de acesso. O comando usado para implantar o modelo mestre é mostrado a seguir. Pode parecer um comando complexo, mas a maioria dos valores é usada como parâmetros. É recomendável alterar os valores desses parâmetros antes da execução. Os modelos vinculados foram carregados em uma conta de armazenamento denominada **st02gvw1dcxm5suwe** no contêiner **armtemplates**. O grupo de recursos deverá ser criado se não existir no momento. O primeiro comando é usado para criar um novo grupo de recursos na região **Europa Ocidental**:

```
New-AzResourceGroup -Name "testvmrg" -Location "West Europe" -Verbose
```

O restante dos valores de parâmetro é necessário para configurar cada recurso. O nome da conta de armazenamento e o valor **dnsNameForPublicIP** devem ser únicos no Azure:

```
New-AzResourceGroupDeployment -Name "testdeploy1" -ResourceGroupName testvmrg -Mode Incremental -TemplateFile "C:\chapter 05\Master.json" -storageAccountName "st02gvwlcdxm5suwe" -storageType "Standard_LRS" -publicIPAddressName "uniipaddname" -publicIPAddressType "Dynamic" -dnsNameForPublicIP "azureforarchitectsbook" -virtualNetworkName vnetwork01 -addressPrefix "10.0.1.0/16" -subnetName "subnet01" -subnetPrefix "10.0.1.0/24" -nicName nic02 -vmSize "Standard_DS1" -adminUsername "sysadmin" -adminPassword $(ConvertTo-SecureString -String sysadmin@123 -AsPlainText -Force) -vhdStorageName oddnewuniqueacc -vhdStorageContainerName vhds -OSDiskName mynewvm -vmName vm10 -windowsOSVersion 2012-R2-Datacenter -imagePublisher MicrosoftWindowsServer -imageOffer WindowsServer -containerName armtemplates -Verbose
```

Nesta seção, abordamos as práticas recomendadas para decompor modelos grandes em modelos menores reutilizáveis e combiná-los em tempo de execução para implantar soluções completas no Azure. À medida que avançamos no livro, vamos modificar o modelo do ARM passo a passo, até explorarmos suas partes principais. Usamos cmdlets do Azure PowerShell para iniciar a implantação de modelos no Azure.

Vamos avançar para o tópico de **copy** e **copyindex**.

## Entender copy e copyIndex

Há muitas vezes em que são necessárias várias instâncias de um determinado recurso ou um grupo de recursos. Por exemplo, você pode precisar provisionar 10 máquinas virtuais do mesmo tipo. Nesses casos, não é prudente implantar modelos 10 vezes para criar essas instâncias. Uma abordagem alternativa melhor é usar os recursos **copy** e **copyIndex** dos modelos do ARM.

**copy** é um atributo de cada definição de recurso. Isso significa que ele pode ser usado para criar várias instâncias de qualquer tipo de recurso.

Vamos entender isso com a ajuda de um exemplo de criação de várias contas de armazenamento dentro de uma única implantação de modelo do ARM.

O próximo trecho de código cria 10 contas de armazenamento serialmente. Eles poderiam ter sido criados em paralelo usando **Parallel** em vez de **Serial** para a propriedade **mode**:

```
"resources": [
    {
        "apiVersion": "2019-06-01",
        "type": "Microsoft.Storage/storageAccounts",
        "location": "[resourceGroup().location]",
        "name": "[concat(variables('storageAccountName'), copyIndex())]",
        "tags": {
            "displayName": "[variables('storageAccountName')]"
        },
        "sku": {
            "name": "Premium_ZRS"
        },
        "kind": "StorageV2",
        "copy": {
            "name": "storageInstances",
            "count": 10,
            "mode": "Serial"
        }
    },
],
]
```

No código anterior, **copy** foi usada para provisionar 10 instâncias da conta de armazenamento serialmente, ou seja, uma após a outra. Os nomes de conta de armazenamento devem ser exclusivos para todas as 10 instâncias, e **copyIndex** foi usado para torná-los exclusivos, concatenando o nome de armazenamento original com o valor do índice. O valor retornado pela função **copyIndex** muda em cada iteração; ele começará em 0 e continuará por 10 iterações. Isso significa que ele retornará **9** para a última iteração.

Agora que aprendemos a criar várias instâncias de um modelo do ARM, vamos nos aprofundar em como proteger esses modelos de vulnerabilidades conhecidas.

## Proteger modelos do ARM

Outro aspecto importante relacionado à criação de modelos do ARM empresariais é protegê-los adequadamente. Os modelos do ARM contêm a configuração de recursos e as informações essenciais sobre a infraestrutura. Portanto, eles não devem ser comprometidos ou acessíveis a pessoas não autorizadas.

A primeira etapa para proteger os modelos do ARM é armazená-los em contas de armazenamento e interromper o acesso anônimo ao contêiner da conta de armazenamento. Além disso, os tokens SAS devem ser gerados para contas de armazenamento e usados em modelos do ARM para consumir modelos vinculados. Isso garantirá que apenas os titulares de tokens SAS possam acessar os modelos. Além disso, esses tokens SAS devem ser armazenados no Azure Key Vault, em vez de serem codificados em modelos do ARM. Isso garantirá que até mesmo as pessoas responsáveis pela implantação não tenham acesso ao token SAS.

Outra etapa para proteger os modelos do ARM é garantir que quaisquer informações confidenciais e segredos, como cadeias de conexão de banco de dados, identificadores de assinatura e locatário do Azure, identificadores de principais serviços, endereços IP e assim por diante, não devam ser codificados em modelos do ARM. Eles devem ser parametrizados, e os valores devem ser obtidos no tempo de execução do Azure Key Vault. No entanto, antes de usar essa abordagem, é importante que esses segredos sejam armazenados no Key Vault antes de executar modelos do ARM.

O código a seguir mostra uma das maneiras que os valores podem ser extraídos do Azure Key Vault no tempo de execução usando o arquivo de parâmetros:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2016-01-01/
deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "storageAccountName": {
            "reference": {
                "keyVault": {
                    "id": "/subscriptions/--subscription id --/
resourceGroups/rpname/providers/Microsoft.KeyVault/vaults/keyvaultbook"),
                    "secretName": "StorageAccountName"
                }
            }
        }
    }
}
```

Nesta listagem de código, um parâmetro é definido que faz referência ao Azure Key Vault para obter valores no tempo de execução durante a implantação. O identificador do Azure Key Vault e o nome do segredo foram fornecidos como valores de entrada.

Agora que você aprendeu a proteger modelos do ARM, vamos ver a identificação das várias dependências entre eles e como podemos habilitar a comunicação entre vários modelos.

## Usar saídas entre modelos do ARM

Um dos aspectos importantes que podem ser facilmente negligenciados ao usar modelos vinculados é que pode haver dependências de recursos dentro de modelos vinculados. Por exemplo, um recurso do SQL Server pode estar em um modelo vinculado que é diferente do recurso de uma máquina virtual. Se quisermos abrir o firewall do SQL Server para o endereço IP da máquina virtual, devemos conseguir passar essas informações dinamicamente para o recurso de firewall do SQL Server após o provisionamento da máquina virtual.

Isso pode ser feito usando o método simples de se referir ao recurso de endereço IP usando a função **REFERENCES** se os recursos do SQL Server e da máquina virtual estiverem no mesmo modelo.

Isso se torna um pouco mais complexo no caso de modelos vinculados se quisermos compartilhar valores de propriedade do tempo de execução de um recurso para outro quando eles estão em modelos diferentes.

Os modelos do ARM fornecem uma configuração **outputs**, que é responsável por gerar saídas da implantação do modelo atual e retorná-las ao usuário. Por exemplo, podemos gerar um objeto completo, conforme mostrado na listagem de código a seguir, usando a função **reference** ou podemos simplesmente gerar um endereço IP como um valor de cadeia:

```
"outputs": {  
    "storageAccountDetails": {  
        "type": "object",  
        "value": "[reference(resourceid  
            ('Microsoft.Storage/storageAccounts',  
            variables('storageAccountName')))]",  
    },  
    "virtualMachineIPAddress": {  
        "type": "string",  
        "value": "[reference(variables  
            ('publicIPAttributeName')).properties.ipAddress]"  
    },  
}
```

Os parâmetros dentro de um modelo vinculado podem ser utilizados pelo modelo mestre. Quando um modelo vinculado é chamado, a saída está disponível para o modelo mestre que pode ser fornecido como um parâmetro para o próximo modelo vinculado ou aninhado. Dessa forma, é possível enviar os valores de configuração do tempo de execução de recursos de um modelo para outro.

O código no modelo mestre seria semelhante ao que é mostrado aqui. Este é o código que é usado para chamar o primeiro modelo:

```
{  
    "type": "Microsoft.Resources/deployments",  
    "apiVersion": "2017-05-10",  
    "name": "createvm",  
    "resourceGroup": "myrg",  
    "dependsOn": [  
        "allResourceGroups"  
    ],  
    "properties": {  
        "mode": "Incremental",  
        "templateLink": {  
            "uri": "[variables(  
                'templateRefSharedServicesTemplateUri')]",  
            "contentVersion": "1.0.0.0"  
        },  
        "parameters": {  
            "VMName": {  
                "value": "[variables('VmName')]"  
            }  
        }  
    }  
}
```

O trecho de código anterior do modelo mestre está chamando um modelo aninhado responsável por provisionar uma máquina virtual. O modelo aninhado tem uma seção de saída que fornece o endereço IP da máquina virtual. O modelo mestre terá outro recurso de implantação em seu modelo que levará o valor de saída e o enviará como um parâmetro para o próximo modelo aninhado, passando o endereço IP no tempo de execução. Isso é mostrado no código a seguir:

```
{  
    "type": "Microsoft.Resources/deployments",  
    "apiVersion": "2017-05-10",  
    "name": "createSQLServer",  
    "resourceGroup": "myrg",  
    "dependsOn": [  
        "createvm"  
    ],  
    "properties": {  
        "mode": "Incremental",  
        "templateLink": {  
            "uri": "[variables('templateRefsql')]",  
            "contentVersion": "1.0.0.0"  
        },  
        "parameters": {  
            "VMName": {  
                "value": "[reference  
('createvm').outputs.virtualMachineIPAddress.value]"  
            }  
        }  
    }  
}
```

Na listagem de código anterior, um modelo aninhado está sendo invocado e um parâmetro está sendo passado para ele. O valor do parâmetro é derivado da saída do modelo vinculado anterior, que é chamado **virtualMachineIPAddress**. Agora, o modelo aninhado obterá dinamicamente o endereço IP da máquina virtual e poderá usá-lo como um endereço IP na lista permissões.

Usando essa abordagem, podemos passar valores de tempo de execução de um modelo aninhado para outro.

## Resumo

Os modelos do ARM são os meios preferidos de provisionamento de recursos no Azure. Eles são idempotentes por natureza, trazendo consistência, previsibilidade e reutilização para a criação do ambiente. Neste capítulo, vimos como criar um modelo do ARM modular. É importante que as equipes passem um tempo criando modelos do ARM de modo adequado para que várias equipes possam trabalhar neles juntas. Eles são altamente reutilizáveis e exigem mudanças mínimas para evoluir. Neste capítulo, aprendemos a criar modelos que são seguros por design, a provisionar várias instâncias de recursos em uma única implantação e a passar saídas de um modelo aninhado para outro usando a seção saídas de modelos do ARM.

O próximo capítulo abordará uma vertente diferente e muito popular da tecnologia conhecida como função sem servidor no Azure. O Azure Functions é um dos principais recursos sem servidor do Azure. Ele será abordado de maneira detalhada, incluindo o Durable Functions.



# 17

## Projetar soluções IoT

No capítulo anterior, você aprendeu sobre modelos do ARM e, até agora, lidamos com preocupações arquitetônicas e suas soluções no Azure em geral. No entanto, este capítulo não se baseia na arquitetura generalizada. De fato, ele explora uma das tecnologias mais revolucionárias deste século. Este capítulo abordará os detalhes da **Internet das Coisas (IoT)** e do Azure.

O Azure IoT refere-se a uma coleção de serviços de nuvem gerenciados pela Microsoft que podem conectar, monitorar e controlar bilhões de ativos de IoT. Em outras palavras, uma solução IoT compõe um ou mais dispositivos IoT que se comunicam constantemente com um ou mais servidores de back-end na nuvem.

Neste capítulo, vamos abordar os seguintes tópicos:

- Azure e IoT
- Visão geral do Azure IoT
- Gerenciamento de dispositivos
- Registro de dispositivos
- Comunicação do dispositivo para o hub IoT
- Escala de soluções IoT
- Alta disponibilidade para soluções IoT
- Protocolos IoT
- Usar as propriedades da mensagem para rotear mensagens

## IoT

A Internet foi inventada na década de 1980 e, mais tarde, tornou-se amplamente disponível. Quase todo mundo avançou em direção a ter uma presença na Internet e começou a criar suas próprias páginas da Web estáticas. Finalmente, o conteúdo estático tornou-se dinâmico e pôde ser gerado dinamicamente com base no contexto. Em quase todos os casos, um navegador era necessário para acessar a Internet. Havia uma infinidade de navegadores disponíveis e, sem eles, o uso da Internet era um desafio.

Durante a primeira década deste século, houve um acontecimento interessante no desenvolvimento: o surgimento de dispositivos portáteis, como telefones celulares e tablets. Os celulares começaram a ficar cada dia mais baratos e disponíveis de forma onipresente. Os recursos de hardware e software desses dispositivos portáteis estavam melhorando consideravelmente, tanto que as pessoas começaram a usar navegadores em seus dispositivos móveis, em vez dos desktops. Mas uma mudança particularmente distinta foi o surgimento de aplicativos móveis. Esses aplicativos móveis eram obtidos de alguma loja por download e eram conectados à Internet para se comunicarem com sistemas de back-end. Até o final da década passada, havia milhões de aplicativos disponíveis com quase todas as funcionalidades imagináveis dentro deles. O sistema de back-end para esses aplicativos foi criado na nuvem para que eles pudessem ser escalados rapidamente. Essa foi a era da conexão entre aplicações e servidores.

Mas esse foi o auge da inovação? Qual foi a próxima evolução da Internet? Bem, outro paradigma agora está tomando o centro do palco: IoT. Em vez de apenas celulares e tablets conectados à internet, por que não pode haver outros dispositivos conectados à internet? Anteriormente, esses dispositivos estavam disponíveis apenas em mercados selecionados; eles eram caros, não estavam disponíveis para as massas e tinham recursos limitados de hardware e software. No entanto, desde a primeira parte desta década, a comercialização desses dispositivos tem crescido em grande escala. Esses dispositivos estão se tornando menores e menores, têm mais capacidade em termos de hardware e software, têm mais poder de computação e armazenamento, podem se conectar à internet em vários protocolos e podem ser conectados a quase tudo. Esta é a era da conexão de dispositivos a servidores, aplicações e outros dispositivos.

Isso levou à formulação da ideia de que os aplicativos de IoT podem, de fato, mudar a forma como as indústrias estão operando. As soluções mais recentes que eram inéditas começaram a se tornar realidade. Agora, esses dispositivos podem ser conectados a qualquer coisa; podem obter informações e enviá-las para um sistema de back-end que pode assimilar informações de todos os dispositivos e executar ações ou relatar incidentes.

Os sensores e controles de IoT podem ser aproveitados em muitos casos de uso de negócios. Por exemplo, eles podem ser usados em sistemas de rastreamento de veículos, que podem rastrear todos os parâmetros essenciais de um veículo e enviá-los para um armazenamento de dados centralizado para análise. Iniciativas de cidades inteligentes também podem usar vários sensores para rastrear os níveis de poluição, temperatura e congestionamento nas ruas. A IoT também encontrou seu caminho para atividades relacionadas à agricultura, como medir a fertilidade do solo, a umidade e muito mais. Você pode acessar os casos de sucesso técnicos da Microsoft para IoT, em <https://microsoft.github.io/techcasestudies/#technology=IoT&sortBy=featured>, para obter exemplos reais de como as organizações aproveitam o Azure IoT.

Antes de explorarmos as ferramentas e os serviços relacionados à IoT, abordaremos primeiro a arquitetura de detalhadamente.

## Arquitetura de IoT

Antes de entrar no Azure e em seus recursos e serviços relacionados à IoT, é importante compreender os vários componentes necessários para criar soluções IoT completas.

Considere dispositivos IoT em todo o mundo estão enviando milhões de mensagens a cada segundo para um banco de dados centralizado. Por que esses dados estão sendo coletados? Bem, a resposta é extraír informações valiosas sobre eventos, anomalias e exceções que estão relacionadas a tudo quando esses dispositivos estão monitorando.

Vamos entender isso mais detalhadamente.

A arquitetura de IoT pode ser dividida em fases distintas, como a seguir:

1. **Conectividade:** essa fase envolve uma conexão entre um dispositivo e o serviço IoT.
2. **Identidade:** depois de se conectar ao serviço IoT, a primeira coisa que ocorre é a identificação do dispositivo e a garantia de que ele tem permissão para enviar telemetria do dispositivo para o serviço IoT. Isso é feito usando um processo de autenticação.
3. **Captura:** durante essa fase, a telemetria do dispositivo é capturada e recebida pelo serviço IoT.
4. **Ingestão:** nessa fase, o serviço IoT ingere a telemetria do dispositivo.
5. **Armazenamento:** a telemetria do dispositivo é armazenada. Pode ser um armazenamento temporário ou permanente.
6. **Transformação:** os dados de telemetria são transformados para processamento adicional. Isso inclui aumentar os dados existentes e inferir dados.
7. **Análise:** os dados transformados são usados para encontrar padrões, anomalias e insights.
8. **Apresentação:** os insights são mostrados como painéis e relatórios. Além disso, novos alertas podem ser gerados que podem invocar scripts e processos de automação.

A Figura 17.1 mostra uma arquitetura genérica baseada em IoT. Os dados são gerados ou coletados por dispositivos e enviados por meio do gateway de nuvem. O gateway de nuvem, por sua vez, envia os dados para vários serviços de back-end para processamento. Os gateways de nuvem são componentes opcionais. Eles devem ser usados quando os próprios dispositivos não são capazes de enviar solicitações para serviços de back-end, seja por causa das limitações de recursos ou pela falta de uma rede confiável. Esses gateways de nuvem podem agrupar dados de vários dispositivos e enviá-los para os serviços de back-end. Em seguida, os dados podem ser processados pelos serviços de back-end e mostrados como insights ou painéis aos usuários:

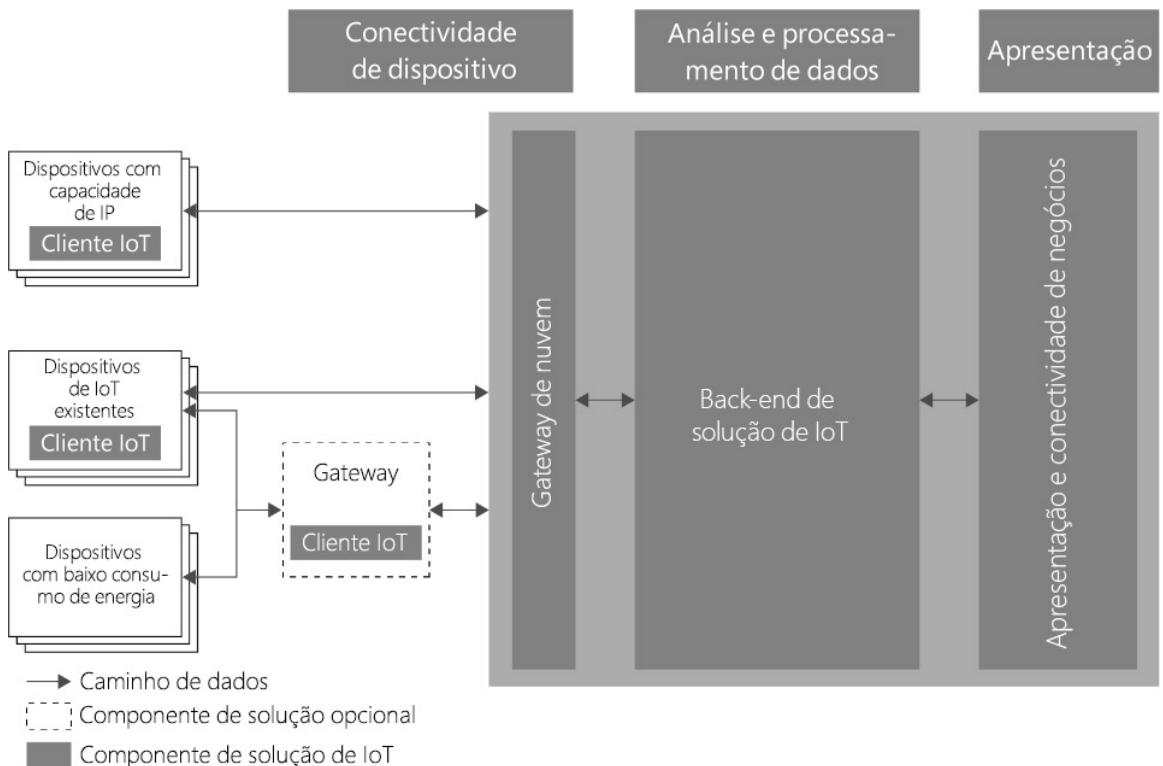


Figura 17.1: uma arquitetura de aplicação IoT genérica

Agora que a arquitetura está clara, vamos avançar e entender como os dispositivos IoT se comunicam com outros dispositivos.

## Conectividade

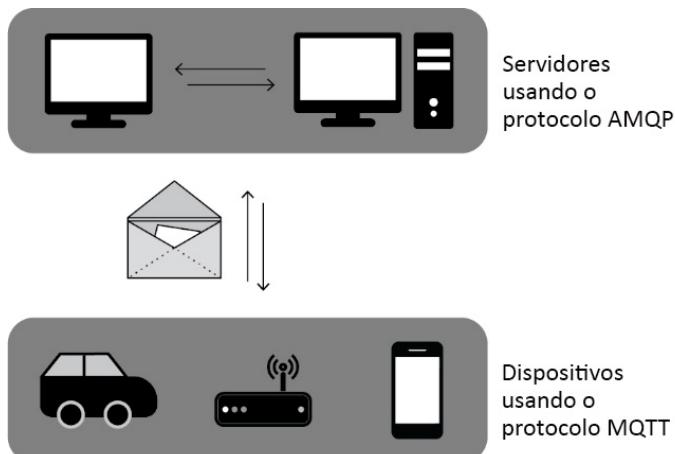
Os dispositivos de IoT precisam se comunicar para se conectar a outros dispositivos. Existem vários tipos de conectividade; por exemplo, a conectividade pode existir entre dispositivos em uma região, entre dispositivos e um gateway centralizado e entre dispositivos e uma plataforma IoT.

Em todos esses casos, os dispositivos IoT precisam de capacidade de conectividade. Essa capacidade poderia ser na forma de conectividade com a Internet, Bluetooth, infravermelho ou qualquer outra comunicação com dispositivos próximos.

No entanto, alguns dispositivos IoT podem não ter a capacidade de se conectar à Internet. Nesses casos, eles podem se conectar por um gateway que, por sua vez, possui conectividade com a Internet.

Os dispositivos IoT usam protocolos para enviar mensagens. Os principais protocolos são o **Advanced Message Queuing Protocol (AMQP)** e o **Message Queue Telemetry Transport (MQTT)**.

Os dados de dispositivos devem ser enviados para uma infraestrutura de TI. O protocolo MQTT é um protocolo de dispositivo para servidor que os dispositivos podem usar para enviar dados de telemetria e outras informações para servidores. Uma vez que o servidor recebe uma mensagem por meio do protocolo MQTT, precisa transportar a mensagem para outros servidores usando uma tecnologia confiável baseada em mensagens e filas. AMQP é o protocolo preferencial para mover mensagens de forma confiável e previsível entre os servidores em uma infraestrutura de TI:



**Figura 17.2: funcionamentos dos protocolos MQTT e AMQP**

Os servidores que recebem as mensagens iniciais de dispositivos IoT devem enviar essas mensagens para outros servidores para qualquer processamento que seja necessário, como salvamento em logs, avaliação, análise e apresentação.

Alguns dispositivos não têm a capacidade de se conectar à Internet ou não oferecem suporte a protocolos que são compatíveis com outras tecnologias de servidor. Para permitir que esses dispositivos funcionem com uma plataforma IoT e na nuvem, gateways intermediários podem ser usados. Os gateways ajudam na integração de dispositivos cuja conectividade e capacidade de rede sejam lentas e não consistentes; esses dispositivos podem usar protocolos não padronizados ou suas capacidades podem ser limitadas em termos de recursos e potência.

Nas circunstâncias em que os dispositivos precisam de infraestrutura adicional para se conectar a serviços back-end, os gateways cliente podem ser implantados. Esses gateways recebem mensagens de dispositivos próximos e os encaminham (ou enviam por push) para a infraestrutura de TI e para a plataforma de IoT para mais consumo. Esses gateways podem traduzir protocolos, se necessário.

Nesta seção, você aprendeu como a comunicação é implementada com outros dispositivos e sobre a função que os gateways desempenham em termos de comunicação. Na próxima seção, vamos falar sobre identidade.

## Identidade

Os dispositivos IoT devem ser registrados em uma plataforma de nuvem. Os dispositivos que não estão registrados não devem ser autorizados a se conectar a uma plataforma de nuvem. Os dispositivos devem ser registrados e receber uma identidade. Um dispositivo deve enviar suas informações de identidade durante a conexão com a nuvem. Se o dispositivo não conseguir enviar essas informações de identidade, a conectividade deverá falhar. Você verá mais adiante neste capítulo como gerar uma identidade para um dispositivo usando uma aplicação de simulação. Como você já sabe, os dispositivos IoT são implantados para capturar informações e, na próxima seção, vamos falar um pouco sobre o processo de captura.

## Captura

Os dispositivos IoT devem ser capazes de capturar informações. Por exemplo, eles devem ser capazes de ler ou monitorar o conteúdo sobre a umidade no ar ou no solo. Essas informações podem ser capturadas com base na frequência, talvez até uma vez por segundo. Assim que as informações são capturadas, o dispositivo deve ser capaz de enviá-las para a plataforma IoT para processamento. Se um dispositivo não conseguir se conectar diretamente à plataforma IoT, poderá se conectar a um gateway de nuvem intermediário e ter que enviar por push as informações capturadas.

O tamanho dos dados capturados e a frequência de captura são as coisas mais importantes para o dispositivo. O fato de um dispositivo ter armazenamento local para conseguir armazenar temporariamente os dados capturados é outro aspecto importante que deve ser considerado. Por exemplo, um dispositivo poderá funcionar em modo offline se houver armazenamento local disponível suficiente. Até mesmo os dispositivos móveis às vezes atuam como dispositivos IoT conectados a diversos instrumentos e têm a capacidade de armazenar dados. Depois de capturar os dados, precisamos ingeri-los em uma plataforma IoT para análise adicional e, na próxima seção, exploraremos a ingestão.

## Ingestão

Os dados capturados e gerados por dispositivos devem ser enviados para uma plataforma IoT capaz de ingerir e consumir esses dados para extrair deles informações significativas e insights. O serviço de ingestão é essencial porque a disponibilidade e a escalabilidade do serviço afetam a taxa de transferência dos dados de entrada. Se os dados começarem a ficar limitados por problemas de escalabilidade ou se a ingestão não for possível devido a problemas de disponibilidade, os dados serão perdidos e o conjunto de dados poderá ser tornar tendencioso ou distorcido. Temos os dados capturados e precisamos de um lugar para armazenar esses dados. Na próxima seção, você aprenderá sobre o armazenamento.

## Armazenamento

As soluções IoT geralmente lidam com milhões ou até mesmo bilhões de registros, com terabytes ou petabytes de dados. Esses são dados valiosos que podem fornecer insights sobre as operações e sua integridade. Esses dados precisam ser armazenados de tal forma que possam ser analisados. O armazenamento deve estar prontamente disponível para análise, aplicações e serviços para consumi-lo. As soluções de armazenamento devem fornecer taxa de transferência e latência adequadas de uma perspectiva de performance, bem como ser altamente disponíveis, escaláveis e seguras. A próxima seção aborda a transformação de dados, que é necessária para armazenar e analisar dados.

## Transformação

Em geral, as soluções IoT são orientadas a dados e têm um volume de dados consideravelmente alto para lidar. Imagine que cada carro tenha um dispositivo e envie mensagens a cada cinco segundos. Se existirem um milhão de carros enviando mensagens, isso será igual a 288 milhões de mensagens por dia e 8 bilhões de mensagens por mês. Juntos, esses dados têm muitas informações e insights ocultos; no entanto, extrair informações desse tipo de dados por mera análise é bastante difícil.

Os dados capturados e armazenados por dispositivos IoT podem ser consumidos para resolver problemas de negócios, mas nem todos os dados capturados são importantes. Apenas um subconjunto de dados pode ser necessário para resolver um problema. Além disso, os dados que os dispositivos IoT coletam podem não ser consistentes. Para garantir que os dados sejam consistentes e não tendenciosos ou distorcidos, as transformações adequadas devem ser executadas neles para torná-los prontos para análise. Durante essa transformação, os dados são filtrados, classificados, removidos, enriquecidos e transformados em uma estrutura, de forma que os dados possam ser consumidos por componentes e aplicações downstream. Precisamos realizar algumas análises com os dados transformados antes de apresentá-los. Como a próxima etapa do fluxo de trabalho, vamos abordar a análise.

## Análise

Os dados transformados na etapa anterior tornam-se a entrada para a etapa de análise. Dependendo do problema em questão, existem diferentes tipos de análise que podem ser executados nos dados transformados.

A seguir estão os diferentes tipos de análise que podem ser executados:

- **Análise descritiva:** esse tipo de análise ajuda na procura de padrões e detalhes sobre o status dos dispositivos IoT e de sua integridade geral. Essa etapa identifica e resume os dados para consumo adicional por meio de etapas de análise mais avançadas. Ela ajudará no resumo, na descoberta de estatísticas relacionadas à probabilidade, na identificação do desvio e em outras tarefas estatísticas.

- **Análise de diagnóstico:** esse tipo de análise é mais avançado do que a análise descritiva. Ela se baseia na análise descritiva e tenta responder às consultas sobre o porquê de certas coisas acontecerem. Isto é, ela tenta encontrar as causas raiz dos eventos. Ela tenta encontrar respostas usando conceitos avançados, como hipótese e correlação.
- **Análise preditiva:** esse tipo de análise tenta prever acontecimentos com uma alta probabilidade de acontecer no futuro. Ela gera previsões que são baseadas em dados antigos; a regressão é um dos exemplos que se baseia em dados antigos. A análise preditiva poderia, por exemplo, prever o preço de um carro, o comportamento do estoque no mercado de ações, quando um pneu de carro vai estourar, e muito mais.
- **Análise prescritiva:** esse tipo de análise é o mais avançado. Essa etapa ajuda a identificar a ação que deve ser realizada para garantir que a integridade dos dispositivos e das soluções não seja prejudicada e a identificar que medidas proativas sejam realizadas. Os resultados desta etapa de análise podem ajudar a evitar problemas futuros e eliminar os problemas em suas causas raiz.

Na fase final, a saída da análise é apresentada de maneira legível por humanos para que um público mais amplo possa compreender e interpretá-la. Na próxima parte, vamos abordar a apresentação.

## Apresentação

A análise ajuda na identificação de respostas, padrões e insights com base nos dados. Esses insights também precisam estar disponíveis para todos os stakeholders em formatos que eles consigam compreender. Para essa finalidade, painéis e relatórios podem ser gerados, de forma estatística ou dinâmica, e então podem ser apresentados aos stakeholders. Os stakeholders podem consumir esses relatórios para executar outras ações e melhorar suas soluções continuamente.

Recapitulando rapidamente todas as etapas anteriores, começamos esta seção analisando a conectividade, em que apresentamos gateways para enviar os dados de dispositivos que não são compatíveis com os protocolos padrão. Em seguida, falamos sobre a identidade e como os dados são capturados. Em seguida, os dados capturados são ingeridos e armazenados para transformação adicional. Após a transformação, a análise é feita nos dados para que sejam apresentados a todos os stakeholders. Como estamos trabalhando no Azure, na próxima seção, abordaremos o que é o Azure IoT e consideraremos os conceitos básicos que aprendemos até agora do ponto de vista do Azure.

## Azure IoT

Agora, você aprendeu sobre os vários estágios de soluções IoT completas. Cada um desses estágios é fundamental, e sua implementação é imprescindível para o sucesso de qualquer solução. O Azure fornece muitos serviços para cada um desses estágios. Além desses serviços, o Azure fornece o Hub IoT do Azure, que é o principal serviço e plataforma IoT do Azure. Ele é capaz de hospedar soluções IoT complexas, altamente disponíveis e escaláveis. Nos aprofundaremos no Hub IoT depois de passarmos por alguns outros serviços:

Dispositivos	Conectividade de dispositivo	Armazenamento	Análise	Apresentação e ação
	Eventos	Banco de Dados SQL	Machine Learning	Serviço de Aplicativo
	Barramento de Serviço	Armazenamento de Blobs/tabela	Stream Analytics	Power BI
	Fontes de dados externos	Cosmos DB	HDInsight	Hubs de notificação
		Fontes de dados externos	Data Factory	Serviços móveis
		Insights de Séries Temporais	Data bricks	Aplicativos Lógicos

Figura 17.3: lista de dispositivos e serviços para soluções IoT

Na próxima seção, seguiremos um padrão semelhante ao da nossa cobertura da arquitetura de IoT para saber mais sobre comunicação, identidade, captura, ingestão, armazenamento, transformação, análise e apresentação com o Azure IoT.

## Conectividade

O Hub IoT fornece todos os conjuntos de protocolos importantes para dispositivos se conectarem a hubs de IoT.

Ele oferece:

- **HTTPS:** o método Transport Protocol Secure HyperText usa certificados que consistem em um par de chaves, conhecidas como chaves privadas/públicas, que são usadas para criptografar e descriptografar dados entre um dispositivo e o Hub IoT. Ele fornece comunicação unidirecional de um dispositivo para a nuvem.
- **AMQP:** é um padrão da indústria para enviar e receber mensagens entre aplicações. Ele fornece uma infraestrutura avançada para a segurança e a confiabilidade das mensagens, e essa é uma das razões pelas quais é amplamente usado no espaço de IoT. Ele fornece recursos do dispositivo para o Hub, bem como do Hub para o dispositivo, e os dispositivos podem usá-lo para autenticar usando o **Claims-Based Security (CBS)** ou **Simple Authentication and Security Layer (SASL)**. Ele é usado principalmente em cenários em que existem gateways de campo, e uma única identidade relacionada a vários dispositivos pode transmitir dados de telemetria para a nuvem.

- **MQTT:** é um padrão da indústria para enviar e receber mensagens entre aplicações. Ele fornece recursos do dispositivo para o Hub, bem como do Hub para o dispositivo. Ele é usado principalmente em cenários em que cada dispositivo tem sua própria identidade e é autenticado diretamente com a nuvem.

Na próxima seção, abordaremos a identidade e como os dispositivos são autenticados.

## Identidade

O Hub IoT também fornece serviços para autenticar dispositivos. Ele oferece uma interface para a geração de hashes de identidade exclusiva para cada dispositivo. Quando os dispositivos enviarem mensagens contendo um hash, o Hub IoT poderá autenticá-las após a verificação do seu próprio banco de dados em busca da existência desse hash. Agora, vamos ver como os dados são capturados.

## Captura

O Azure fornece gateways IoT, que habilitam dispositivos que não estão em conformidade com o Hub IoT para se adaptar e enviar dados por push. Gateways locais ou intermediários podem ser implantados perto de dispositivos de forma que vários dispositivos possam se conectar a um único gateway para conectar e enviar suas informações. Da mesma forma, vários desses clusters de dispositivos com gateways locais também podem ser implantados. Pode haver um gateway de nuvem implantado na própria nuvem, o qual é capaz de capturar e aceitar dados de várias fontes e ingeri-los para o Hub IoT. Como abordado anteriormente, precisamos ingerir os dados que capturamos. Na próxima seção, você aprenderá sobre a ingestão com o Hub IoT.

## Ingestão

Um Hub IoT pode ser um único ponto de contato para dispositivos e outras aplicações. Em outras palavras, a ingestão de mensagens IoT é de responsabilidade do serviço do Hub IoT. Existem outros serviços, tais como Hubs de Eventos e a infraestrutura de serviço de mensagens do Barramento de Serviço, que podem ingerir as mensagens recebidas. No entanto, os benefícios e as vantagens de usar o Hub IoT para ingerir dados da IoT superam muito o uso de Hubs de Eventos e serviço de mensagens de Barramento de Serviço. Na verdade, o Hub IoT foi criado especificamente com a finalidade de ingerir mensagens IoT dentro do ecossistema do Azure, de forma que os outros serviços e componentes possam agir sobre eles. Os dados ingeridos são armazenados no armazenamento. Antes de fazermos qualquer tipo de transformação ou análise, vamos explorar a função de armazenamento no fluxo de trabalho na próxima seção.

## Armazenamento

O Azure fornece várias maneiras de armazenar mensagens de dispositivos IoT. Elas incluem o armazenamento de dados relacionais, dados NoSQL sem esquemas e blobs:

- **Banco de Dados SQL:** fornece armazenamento de dados relacionais, JSON e documentos XML. Ele fornece uma linguagem de consulta SQL rica e usa um SQL Server completo como um serviço. Os dados dos dispositivos poderão ser armazenados em bancos de dados SQL caso estejam bem definidos e o esquema não precisará passar por mudanças frequentes.
- **Armazenamento do Azure:** fornece armazenamento de tabelas e de blobs. O armazenamento de tabelas ajuda no armazenamento de dados como entidades, onde o esquema não é importante. Os blobs ajudam no armazenamento de arquivos em contêineres como blobs.
- **Cosmos DB:** é um banco de dados NoSQL de escala empresarial completo. Ele está disponível como um serviço que é capaz de armazenar dados sem esquema. É um banco de dados globalmente distribuído que pode se estender por continentes, fornecendo alta disponibilidade e escalabilidade de dados.
- **Fontes de dados externas:** além dos serviços do Azure, os clientes podem trazer ou usar seus próprios armazenamentos de dados, como um SQL Server em máquinas virtuais do Azure, e podem usá-los para armazenar dados no formato relacional.

A próxima seção é sobre transformação e análise.

## Transformação e análise

O Azure fornece vários recursos para executar tarefas e atividades em dados ingeridos. Alguns deles são listados aqui:

- **Data Factory:** o Azure Data Factory é um serviço de integração de dados baseados em nuvem que nos permite criar fluxos de trabalho orientados por dados na nuvem para orquestração e automatização da movimentação de dados e transformação de dados. O Azure Data Factory ajuda a criar e a agendar fluxos de trabalho orientados a dados (chamados pipelines) que podem ingerir dados de armazenamentos de dados diferentes; processar e transformar os dados usando serviços de computação, como **Azure HDInsight**, **Hadoop**, **Spark**, **Azure Data Lake Analytics**, **Azure Synapse Analytics** e **Azure Machine Learning**; publicar dados de saída para um data warehouse para aplicações de **Business Intelligence (BI)** em vez de uma plataforma tradicional de **Extrair, Transformar e Carregar (ETL)**.
- **Azure Databricks:** fornece um ambiente Spark completo e gerenciado. Ele pode ajudar na transformação de dados usando Scala e Python. Ele também fornece uma biblioteca de SQL para manipular dados usando a sintaxe SQL tradicional. É mais eficiente do que os ambientes Hadoop.

- **Azure HDInsight:** a Microsoft e a Hortonworks se uniram para ajudar as empresas oferecendo uma solução analítica de big data com o Azure. O HDInsight é um ambiente de serviço de nuvem totalmente gerenciado, de alta performance, desenvolvida com Apache Hadoop e Apache Spark usando Microsoft Azure HDInsight. Ajuda na aceleração de workloads com facilidade usando o serviço de nuvem de big data líder da indústria da Microsoft e da Hortonworks.
- **Azure Stream Analytics:** é um serviço de análise de dados em tempo real totalmente gerenciado que ajuda na realização de cálculo e transformação no streaming de dados. O Stream Analytics pode examinar grandes volumes de dados fluindo de dispositivos ou processos, extrair informações do fluxo de dados e procurar padrões, tendências e relacionamentos.
- **Machine Learning:** é uma técnica de ciência de dados que permite que os computadores usem os dados existentes para previsão de comportamentos futuros, resultados e tendências. Usando o aprendizado de máquina, os computadores aprendem comportamentos com base no modelo que criamos. O Azure Machine Learning é um serviço de análise preditiva baseado em nuvem que possibilita a rápida criação e implantação de modelos preditivos como soluções de análise. Ele fornece uma biblioteca de algoritmos pronta para uso para criar modelos em um computador conectado à Internet e implantar soluções preditivas rapidamente.
- **Azure Synapse Analytics:** anteriormente conhecido como SQL Data Warehouse do Azure. O Azure Synapse Analytics fornece serviços de análise ilimitado que são ideais para data warehouses corporativos e análise de big data. Ele oferece suporte à ingestão direta de streaming, que pode ser integrada ao Hub IoT do Azure.

Agora que você está familiarizado com as ferramentas de transformação e análise usadas no Azure para os dados ingeridos pelos dispositivos IoT, vamos avançar e aprender como esses dados podem ser apresentados.

## Apresentação

Após a execução de análises adequadas nos dados, os dados deverão ser apresentados aos stakeholders em um formato consumível por eles. Existem inúmeras maneiras de apresentar insights de dados. Elas incluem a apresentação de dados por meio de aplicações Web implantadas usando o Serviço de Aplicativo do Azure, enviando dados para hubs de notificação que podem notificar aplicações móveis e muito mais. No entanto, a abordagem ideal para apresentar e consumir insights é por meio de relatórios e painéis do **Power BI**. O Power BI é uma ferramenta de visualização da Microsoft que é usada para renderizar relatórios dinâmicos e painéis na Internet.

Para concluir, o IoT do Azure está estreitamente alinhado com os conceitos básicos de arquitetura de IoT. Ele segue o mesmo processo. No entanto, o Azure nos dá a liberdade de escolher diferentes serviços e dependências com base em nossos requisitos. Na próxima seção, vamos nos concentrar no Hub IoT do Azure, um serviço hospedado na nuvem e totalmente gerenciado pelo Azure.

## Hub IoT do Azure

Os projetos de IoT são geralmente complexos por natureza. A complexidade surge devido ao alto volume de dispositivos e dados. Os dispositivos são incorporados em todo o mundo; por exemplo, monitorando e auditando dispositivos, que são usados para armazenar dados, transformar e analisar petabytes de dados e, por fim, realizar ações com base em insights. Além disso, esses projetos têm longos períodos de gestação, e seus requisitos mudam constantemente por conta dos prazos.

Se uma empresa quer iniciar uma jornada com um projeto de IoT, acabará percebendo que os problemas que mencionamos não são facilmente resolvidos. Esses projetos precisam de hardware suficiente em termos de computação e armazenamento para lidar, além de serviços que possam trabalhar com grandes volumes de dados.

O Hub IoT é uma plataforma criada para proporcionar entrega mais rápida, melhor e mais fácil de projetos de IoT. Ele fornece todos os recursos e os serviços necessários, incluindo o seguinte:

- Registro do dispositivo
- Conectividade de dispositivo
- Gateways de campo
- Gateways de nuvem
- Implementação de protocolos de indústria, como AMQP e MQTT
- Um hub para armazenar as mensagens recebidas
- O roteamento de mensagens baseadas em propriedades e conteúdo de mensagem
- Vários pontos de extremidade para diferentes tipos de processamento
- Conectividade com outros serviços no Azure para análise em tempo real e muito mais

Abordamos uma visão geral do Hub IoT do Azure. Agora, vamos no aprofundar para entender mais sobre os protocolos e como os dispositivos são registrados no Hub IoT do Azure.

## Protocolos

O Hub IoT do Azure dá suporte nativo à comunicação sobre os protocolos HTTP, MQTT e AMQP. Em alguns casos, dispositivos ou gateways de campo podem não ser capazes de usar um desses protocolos padrão e exigirão adaptação do protocolo. Nesses casos, gateways personalizados podem ser implantados. O gateway de protocolo do IoT do Azure pode habilitar adaptação do protocolo para pontos de extremidade do Hub IoT ao criar uma ponte para o tráfego de e para o Hub IoT. Na próxima seção, abordaremos como os dispositivos são registrados no Hub IoT do Azure.

## Registro do dispositivo

Os dispositivos devem ser registrados antes que eles possam enviar mensagens para o Hub IoT. O registro de dispositivos pode ser feito manualmente usando o portal do Azure ou pode ser automatizado usando o SDK do Hub IoT. O Azure também fornece exemplos de aplicações de simulação, por meio dos quais fica mais fácil registrar dispositivos virtuais para fins de desenvolvimento e de testes. Há também um simulador online do Raspberry Pi que pode ser usado como um dispositivo virtual, e depois, obviamente, haverá outros dispositivos físicos que poderão ser configurados para se conectar ao Hub IoT.

Se você desejar simular um dispositivo de um computador local que geralmente é usado para fins de testes e de desenvolvimento, há tutoriais disponíveis na documentação do Azure em vários idiomas. Esses tutoriais estão disponíveis em <https://docs.microsoft.com/azure/iot-hub/iot-hub-get-started-simulated>.

O simulador online do Raspberry Pi está disponível em <https://docs.microsoft.com/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started>, e para dispositivos físicos que precisam ser registrados com o Hub IoT, as etapas mencionadas em <https://docs.microsoft.com/azure/iot-hub/iot-hub-get-started-physical> devem ser usadas.

Para adicionar manualmente um dispositivo usando o portal do Azure, o Hub IoT fornece um menu **Dispositivos IoT**, que pode ser usado para configurar um novo dispositivo. Selecionar a opção **Novo** permitirá criar um novo dispositivo, conforme mostrado na Figura 17.4:

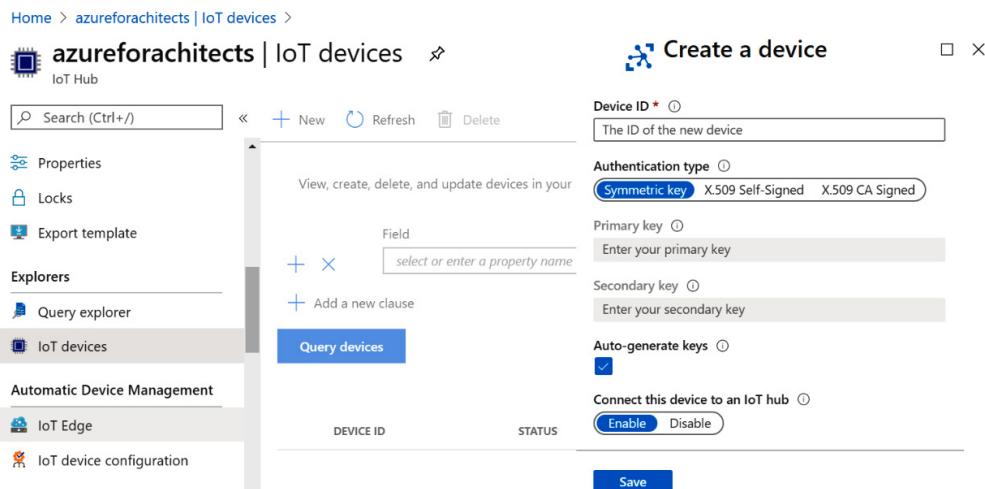


Figura 17.4: adicionando um dispositivo por meio do portal do Azure

Após a criação da identidade do dispositivo, uma cadeia de conexão da chave primária para o Hub IoT deve ser usada em cada dispositivo para se conectar a ele:

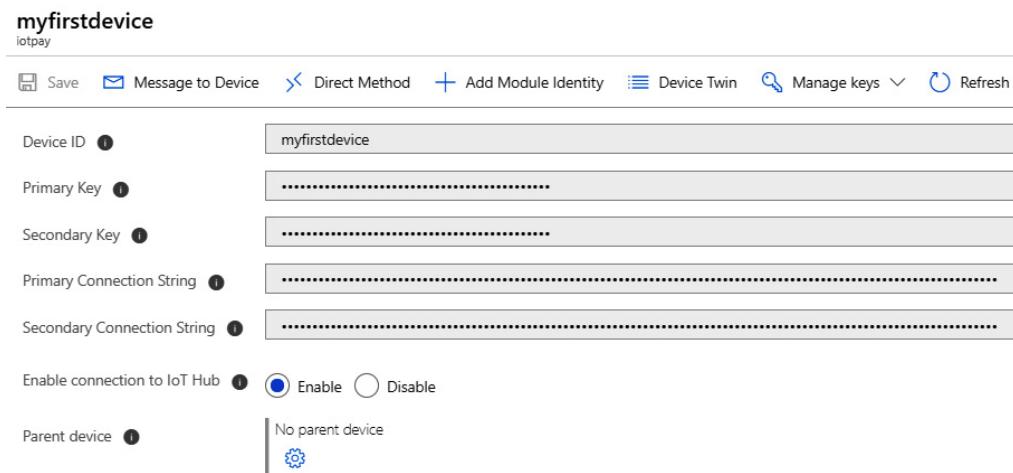


Figura 17.5: criando cadeias de conexão para cada dispositivo

Nesta fase, o dispositivo foi registrado com o Hub IoT, e nossa próxima missão é fazer com que a comunicação aconteça entre o dispositivo e o Hub IoT. A próxima seção oferece uma boa compreensão de como o gerenciamento de mensagens é feito.

## Gerenciamento de mensagens

Depois que os dispositivos forem registrados no Hub IoT, eles poderão começar a interagir com ele. O gerenciamento de mensagens refere-se a como a comunicação ou a interação entre o dispositivo IoT e o Hub IoT é feita. Essa interação pode ser do dispositivo para a nuvem ou da nuvem para o dispositivo.

## Serviço de mensagens do dispositivo para a nuvem

Uma das melhores práticas que devem ser seguidas nesta comunicação é que, embora o dispositivo possa estar capturando muitas informações, somente os dados de alguma relevância devem ser transmitidos para a nuvem. O tamanho da mensagem é muito importante em soluções IoT, pois as soluções IoT em geral têm volumes muito elevados de dados. Até mesmo 1 KB de fluxo de dados extras pode resultar em um gigabyte de armazenamento e processamento desperdiçados. Cada mensagem tem propriedades e cargas. As propriedades definem os metadados para a mensagem. Esses metadados contêm dados sobre o dispositivo, identificação, marcas e outras propriedades que são úteis no roteamento e na identificação das mensagens.

Os dispositivos ou os gateways de nuvem devem se conectar ao Hub IoT para transferir dados. O Hub IoT fornece pontos de extremidade públicos que podem ser utilizados por dispositivos para a conexão e o envio de dados. O Hub IoT deve ser considerado como o primeiro ponto de contato para processamento de back-end. O Hub IoT é capaz de transmitir e rotear essas mensagens para vários serviços. Por padrão, as mensagens são armazenadas em hubs de eventos. Vários hubs de eventos podem ser criados para diferentes tipos de mensagens. Os pontos de extremidade internos usados pelos dispositivos para enviar e receber dados podem ser vistos na folha **Pontos de extremidade internos** no Hub IoT. A Figura 17.6 mostra como você pode encontrar os pontos de extremidade internos:

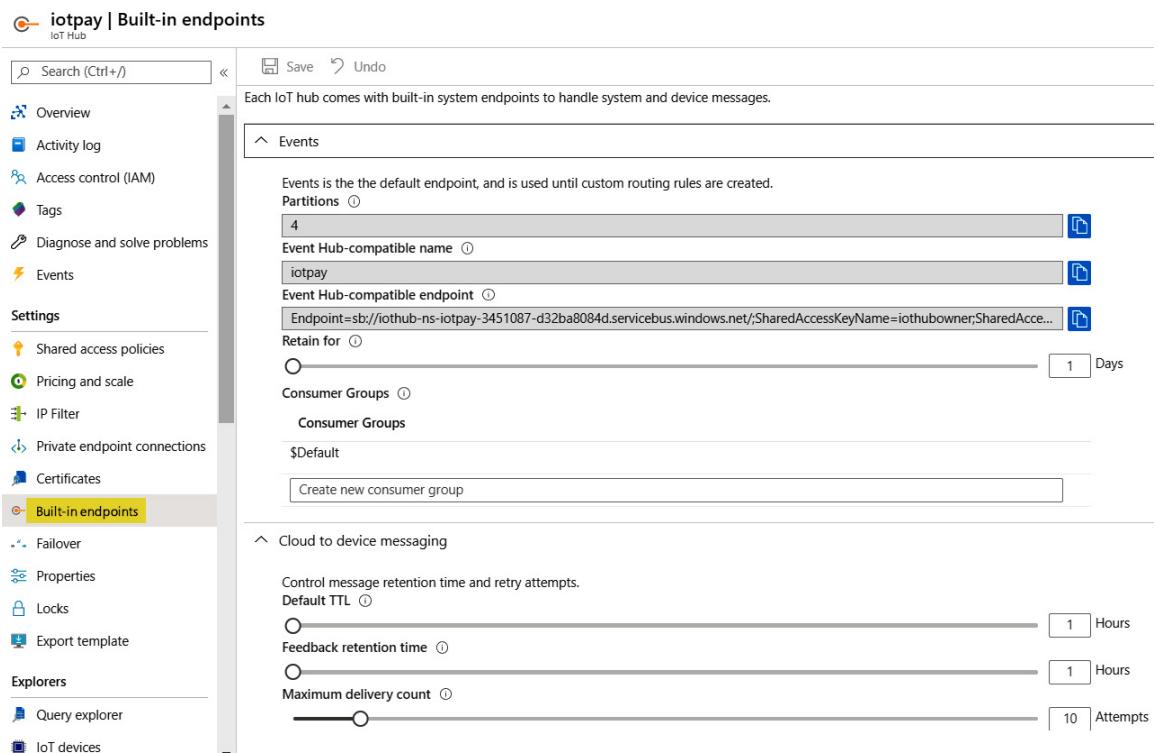


Figura 17.6: criando vários hubs de eventos

As mensagens podem ser roteadas para diferentes pontos de extremidade com base em propriedades de cabeçalho e de corpo da mensagem, conforme mostrado na Figura 17.7:

The screenshot shows the 'Add a route' interface in the Azure IoT hub portal. It includes fields for 'Name' (with a placeholder 'name'), 'Endpoint' (with a dropdown menu showing 'Event hubs', 'Service bus queue', 'Service bus topic', and 'Storage'), 'Data source' (set to 'Device Telemetry Messages'), 'Enable route' (set to 'Enable'), and a 'Routing query' field containing '1 true'. A yellow box highlights the '+ Add endpoint' button and the 'Event hubs' option in the endpoint dropdown.

Figura 17.7: adicionando uma nova rota a pontos de extremidade diferentes

As mensagens em um hub IoT permanecem lá por sete dias por padrão, e seu tamanho pode ser até 256 KB.

Há um simulador de exemplo fornecido pela Microsoft para simular o envio de mensagens para a nuvem. Ele está disponível em vários idiomas; e a versão em C# pode ser visualizada em <https://docs.microsoft.com/azure/iot-hub/iot-hub-csharp-csharp-c2d>.

### Serviço de mensagens da nuvem para o dispositivo

O Hub IoT é um serviço gerenciado que fornece uma infraestrutura de serviço de mensagens bidirecional. As mensagens podem ser enviadas da nuvem para dispositivos e com base na mensagem sobre a qual os dispositivos podem agir.

Existem três tipos de padrões de serviço de mensagens da nuvem para o dispositivo:

- Os métodos diretos exigem confirmação imediata de resultados. Com frequência, os métodos diretos são usados para o controle interativo de dispositivos, como a abertura e o fechamento de portões de garagem. Eles seguem o padrão de solicitação-resposta.
- A configuração de propriedades do dispositivo usando o IoT do Azure fornece propriedades do **dispositivo gêmeo**. Por exemplo, você pode definir o intervalo de envio de telemetria para 30 minutos. Os dispositivos gêmeos são documentos JSON que armazenam informações do estado do dispositivo (como metadados, configurações e condições). Um hub IoT persiste um dispositivo gêmeo para cada dispositivo no hub IoT.

- Mensagens da nuvem para o dispositivo são usadas para notificações unidirecionais para o aplicativo do dispositivo. Isso segue o padrão disparar e esquecer.

Em todas as organizações, a segurança é uma grande preocupação e, mesmo no caso de dispositivos e dados IoT, essa preocupação ainda existe. Abordaremos a segurança na próxima seção.

## Segurança

A segurança é um aspecto importante em aplicações baseadas em IoT. Os aplicações baseadas em IoT são compostas por dispositivos que usam a Internet pública para conectividade com aplicações de back-end. A proteção de dispositivos, de aplicações de back-end e de conectividade de usuários mal-intencionados e hackers deve ser considerada uma prioridade para o sucesso dessas aplicações.

### Segurança na IoT

As aplicações IoT são criadas principalmente em torno da Internet, e a segurança deve desempenhar um papel importante para garantir que a solução não seja comprometida. Algumas das decisões de segurança mais importantes que afetam a arquitetura de IoT são listadas aqui:

- Sobre os dispositivos que usam HTTP versus pontos de extremidade REST HTTPS, os pontos de extremidade REST protegidos por certificados garantem que as mensagens transferidas de um dispositivo para a nuvem e vice-versa sejam bem criptografadas e assinadas. As mensagens não devem fazer sentido para um intruso e devem ser extremamente difíceis de decifrar.
- Se os dispositivos estiverem conectados a um gateway local, o gateway local deverá se conectar à nuvem usando um protocolo HTTP seguro.
- Os dispositivos devem ser registrados no Hub IoT para que possam enviar mensagens.
- As informações passadas para a nuvem devem ser mantidas no armazenamento, que está bem protegido e seguro. Os tokens SAS apropriados ou as cadeias de conexão armazenados no Azure Key Vault devem ser usados para a conexão.
- O Azure Key Vault deve ser usado para armazenar todos os segredos, as senhas e as credenciais, incluindo os certificados.
- A Central de Segurança do Azure para IoT fornece prevenção e análise de ameaças para todos os dispositivos, IoT Edge e Hub IoT em todos os ativos de IoT. Podemos criar nossos próprios painéis na Central de Segurança do Azure com base nas avaliações de segurança. Alguns dos principais recursos incluem gerenciamento na Central de Segurança do Azure, proteção contra ameaças adaptativas e detecção de ameaças inteligentes. É uma prática recomendada considerar a Central de Segurança do Azure ao implementar soluções IoT seguras.

Em seguida, vamos analisar o aspecto de escalabilidade do Hub IoT.

## Escalabilidade

A escalabilidade para o Hub IoT é um pouco diferente de outros serviços. No Hub IoT, existem dois tipos de mensagens:

- **Entrada:** mensagens do dispositivo para a nuvem
- **Saída:** mensagens da nuvem para o dispositivo

Os dois tipos precisam ser contabilizados em termos de escalabilidade.

O Hub IoT fornece um par de opções de configuração durante o tempo de provisionamento para configurar a escalabilidade. Essas opções também estarão disponíveis após o provisionamento e podem ser atualizadas para se ajustarem melhor aos requisitos da solução em termos de escalabilidade.

As opções de escalabilidade disponíveis para o Hub IoT são:

- A edição de **Unidade de Manutenção de Estoque (SKU)**, que é o tamanho do Hub IoT
- Número de unidades

Primeiro, investigaremos a opção Edição de SKU.

### Edição de SKU

O SKU no Hub IoT determina o número de mensagens que um hub pode manipular por unidade por dia, e isso inclui as mensagens de entrada e de saída. Há quatro níveis:

- **Gratuito:** permite 8.000 mensagens por unidade por dia e permite mensagens de entrada e de saída. No máximo, uma unidade pode ser provisionada. Essa edição é apropriada para se familiarizar e testar os recursos do serviço do Hub IoT.
- **Standard (S1):** permite 400.000 mensagens por unidade por dia e permite mensagens de entrada e de saída. No máximo, 200 unidades podem ser provisionadas. Essa edição é apropriada para um número pequeno de mensagens.
- **Standard (S2):** permite 6 milhões de mensagens por unidade por dia e permite mensagens de entrada e de saída. No máximo, 200 unidades podem ser provisionadas. Essa edição é apropriada para um número grande de mensagens.
- **Standard (S3):** permite 300 milhões de mensagens por unidade por dia e permite mensagens de entrada e de saída. No máximo, 10 unidades podem ser provisionadas. Essa edição é apropriada para um número muito grande de mensagens.

As opções de atualização e escalabilidade estão disponíveis no portal do Azure, na folha **Preços e escala** do Hub IoT. As opções serão apresentadas a você, conforme mostrado na Figura 17.8:

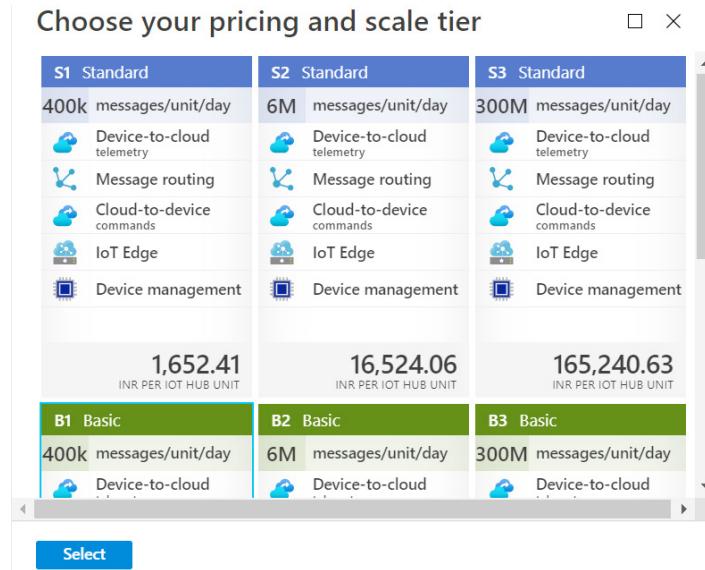


Figura 17.8: escolhendo uma camada de preços e escala

Você pode notar que a camada **Standard S3** permite um máximo de **10 unidades**, só comparado a outras unidades padrão que permitem **200 unidades**. Isso está diretamente relacionado ao tamanho dos recursos de computação provisionados para executar serviços IoT. O tamanho e a capacidade de máquinas virtuais para **Standard S3** são significativamente maior em comparação a outros níveis onde o tamanho permanece o mesmo.

## Unidades

As unidades definem o número de instâncias de cada SKU em execução por trás do serviço. Por exemplo, duas unidades do nível de SKU **Standard S1** significarão que o Hub IoT é capaz de lidar com  $400\text{ K} * 2 = 800\text{ K}$  de mensagens por dia.

Mais unidades aumentarão a escalabilidade da aplicação: A Figura 17.9 vem da folha **Preços e escala** do Hub IoT, onde você pode ver a camada de preços atual e o número de unidades:

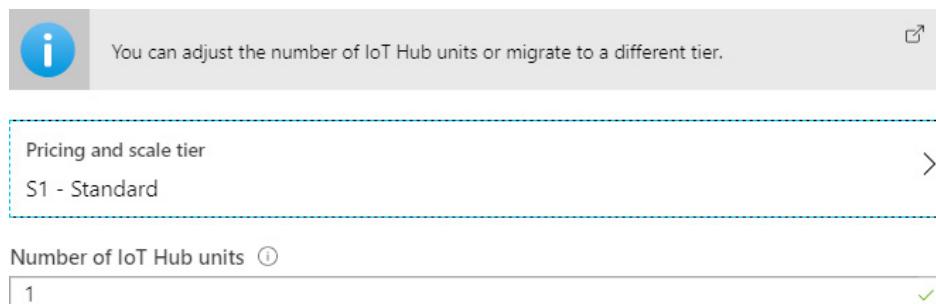


Figura 17.9: opções para ajustar ou migrar unidades do Hub IoT

Um dos serviços em expansão no Hub IoT do Azure agora é o Azure IoT Edge, que é um serviço totalmente gerenciado criado no Hub IoT do Azure. Vamos explorar o Azure IoT Edge na próxima seção.

## Azure IoT Edge

O Microsoft Azure IoT Edge aproveita a computação de borda para implementar soluções IoT. A computação de borda refere-se aos recursos de computação que estão disponíveis em sua rede na infraestrutura local, no final da sua rede, onde a Internet pública é iniciada. Ela pode ser implantada em sua rede principal ou em uma rede de convidados com isolamento de firewall.

O Azure IoT Edge consiste no tempo de execução do IoT Edge, que precisa ser instalado em um computador ou dispositivo. O Docker será instalado no computador. O computador pode ser executado no Windows ou no Linux. A função do Docker é executar os módulos do IoT Edge.

O Azure IoT Edge depende do conceito de nuvem híbrida, em que você pode implantar e gerenciar soluções IoT no hardware na infraestrutura local e integrá-los facilmente ao Microsoft Azure.

A Microsoft fornece uma documentação abrangente para o Azure IoT Edge, com modelos de início rápido e orientações sobre como instalar os módulos. O link para a documentação é <https://docs.microsoft.com/azure/iot-edge>.

Na próxima seção, analisaremos como a infraestrutura é gerenciada no caso do Hub IoT do Azure e como a alta disponibilidade é oferecida aos clientes.

## Alta disponibilidade

O Hub IoT é uma oferta de **Plataforma como Serviço (PaaS)** do Azure. Os clientes e os usuários não interagem diretamente com o número e o tamanho subjacentes de máquinas virtuais nas quais o serviço do Hub IoT é executado. Os usuários decidem a região, a SKU do hub IoT e o número de unidades para sua aplicação. O restante da configuração é determinado e executado pelo Azure nos bastidores. O Azure garante que cada serviço PaaS seja altamente disponível por padrão. Ele faz isso ao garantir que várias máquinas virtuais provisionadas por trás do serviço estejam em racks separados no datacenter. Ele faz isso colocando as máquinas virtuais em um conjunto de disponibilidade e em domínios de falha e de atualização separados. Isso ajuda a garantir a alta disponibilidade para a manutenção planejada e não planejada. Os conjuntos de disponibilidade cuidam da alta disponibilidade no nível do datacenter.

Na próxima seção, abordaremos o Azure IoT Central.

## Azure IoT Central

O Azure IoT Central fornece uma plataforma para criar aplicações IoT de nível empresarial para atender aos seus requisitos de negócios de forma segura, confiável e escalável. O IoT Central elimina o custo de desenvolvimento, manutenção e gerenciamento de soluções IoT.

O IoT Central fornece gerenciamento centralizado por meio do qual você pode gerenciar e monitorar dispositivos, condições do dispositivo, criação de regras e dados de dispositivos. Na Figura 17.10, você pode ver alguns dos modelos que estão disponíveis no portal do Azure durante a criação de aplicações do IoT Central:

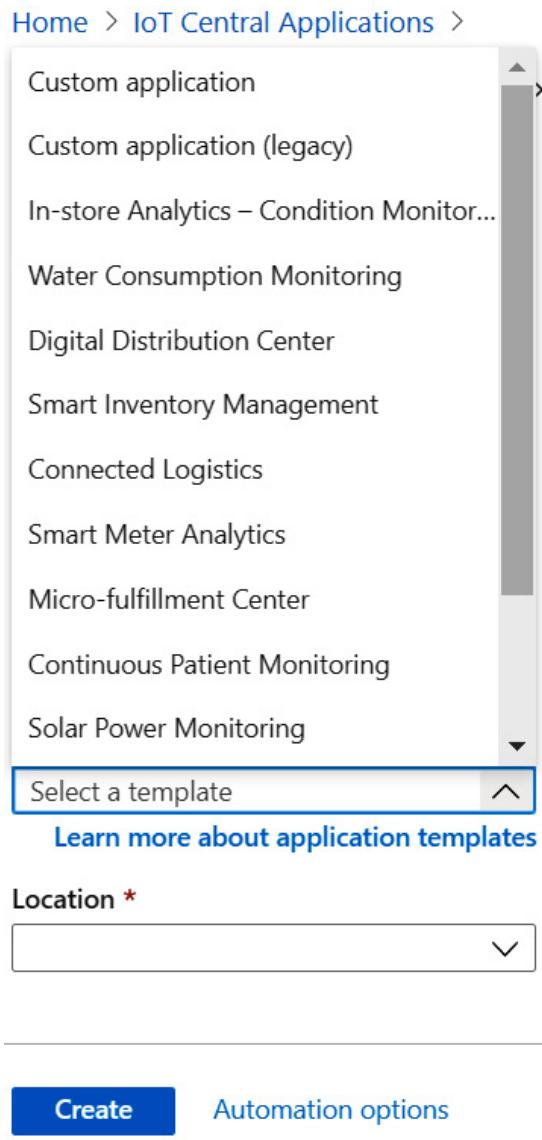


Figura 17.10: criando uma aplicação do Azure IoT Central

Os modelos oferecerão uma vantagem, e você poderá personalizá-los de acordo com seus requisitos. Isso economizará muito tempo durante a fase de desenvolvimento.

No momento da redação, o IoT Central oferece um trial de sete dias, e você pode ver os preços deste serviço aqui: <https://azure.microsoft.com/pricing/details/iot-central/?rtc=1>.

O Azure IoT Central é um benefício para todas as organizações que estão desenvolvendo aplicações IoT.

## Resumo

A IoT é uma das maiores tecnologias emergentes desta década e já está transformando as indústrias. Coisas que pareciam impossíveis antes agora são repentinamente possíveis.

Neste capítulo, exploramos o Hub IoT e abordamos a entrega de soluções IoT para o cliente de maneira mais rápida, melhor e mais barata do que as soluções alternativas. Também abordamos como a IoT pode rastrear o ciclo de vida de desenvolvimento inteiro e ajudar a agilizar o tempo de lançamento no mercado para as empresas. Por fim, você aprendeu sobre o Azure IoT Edge e o Azure IoT Central.

Para ajudá-lo a analisar os crescentes volumes de dados com eficácia, abordaremos o Azure Synapse Analytics no próximo capítulo.





# 18

## Azure Synapse Analytics para arquitetos

O Azure Synapse Analytics é uma evolução inovadora do SQL Data Warehouse do Azure. O Azure Synapse é um serviço integrado de análise de dados totalmente gerenciado que combina data warehouse, integração de dados e processamento de big data com tempo acelerado para insights para formar um único serviço.

Neste capítulo, exploraremos Azure Synapse Analytics cobrindo os seguintes tópicos:

- Uma visão geral do Azure Synapse Analytics
- Introdução aos espaços de trabalho do Synapse e ao Synapse Studio
- Migrar de sistemas herdados existentes para o Azure Synapse Analytics
- Migrar dados e esquemas de data warehouse existentes para o Azure Synapse Analytics
- Redesenvolver processos de ETL escaláveis usando o Azure Data Factory
- Problemas comuns de migração e resoluções
- Considerações sobre segurança
- Ferramentas para ajudar a migrar para o Azure Synapse Analytics

## Azure Synapse Analytics

Atualmente, com armazenamento acessível e capacidade de armazenamento elástico alta, as organizações estão acumulando mais dados do que nunca. Arquitetar uma solução para analisar esses grandes volumes de dados para fornecer insights significativos sobre uma empresa pode ser um desafio. Um obstáculo que muitas empresas enfrentam é a necessidade de gerenciar e manter dois tipos de sistemas de análise:

- **Data warehouses:** fornecem insights críticos sobre o negócio.
- **Data lakes:** fornecem insights significativos sobre clientes, produtos, funcionários e processos por meio de várias metodologias de análise.

Ambos os sistemas de análise são fundamentais para as empresas e, ainda assim, operam de forma independente uns dos outros. Ao mesmo tempo, as empresas precisam obter insights de todos os seus dados organizacionais para se manterem competitivas e inovar em processos para obter melhores resultados.

Os arquitetos que precisam criar seus próprios pipelines de dados completo devem seguir estas etapas:

1. Ingerir dados de várias fontes de dados.
2. Carregar todas essas fontes de dados em um data lake para processamento adicional.
3. Executar a limpeza de dados em diferentes estruturas e tipos de dados.
4. Preparar, transformar e modelar os dados.
5. Fornecer os dados limpos a milhares de usuários por meio de ferramentas e aplicações de BI.

Até agora, cada uma dessas etapas exigia uma ferramenta diferente. Evidentemente, com tantos serviços, aplicações e ferramentas diferentes disponíveis no mercado, escolher os mais adequados pode ser uma tarefa assustadora.

Há inúmeros serviços disponíveis para ingerir, carregar, preparar e fornecer dados. Também há inúmeros serviços para limpeza de dados com base na linguagem escolhida pelo desenvolvedor. Além disso, alguns desenvolvedores podem preferir usar o SQL, alguns podem querer usar o Spark, enquanto outros podem querer usar ambientes sem código para transformar os dados.

Mesmo depois que a coleção aparentemente adequada de ferramentas for selecionada, muitas vezes há uma curva de aprendizado íngreme para essas ferramentas. Além disso, os arquitetos podem encontrar desafios logísticos inesperados na manutenção de um pipeline de dados em plataformas e linguagens diferentes devido a incompatibilidades. Com essa gama de problemas, implementar e manter soluções analíticas baseadas em nuvem pode ser uma tarefa difícil.

O Azure Synapse Analytics resolve esses problemas e muito mais. Ele simplifica todo o padrão de data warehouse moderno, permitindo que os arquitetos se concentrem em criar soluções de análise completas em um ambiente unificado.

## Um cenário comum para arquitetos

Um dos cenários mais comuns que um arquiteto enfrenta é ter que fazer um plano para migrar as soluções herdadas existentes de data warehouse para uma solução moderna de análise empresarial. Com sua escalabilidade ilimitada e experiência unificada, o Azure Synapse tornou-se uma das principais opções para muitos arquitetos considerarem. Mais adiante neste capítulo, também abordaremos considerações arquitetônicas comuns para migrar de uma solução herdada existente de data warehouse para o Azure Synapse Analytics.

Na próxima seção, forneceremos uma visão geral técnica dos principais recursos do Azure Synapse Analytics. Os arquitetos que são novos no ecossistema do Azure Synapse terão o conhecimento necessário sobre o Synapse depois de ler este capítulo.

## Uma visão geral do Azure Synapse Analytics

O Azure Synapse Analytics permite que os profissionais de dados criem soluções de análise completas, aproveitando uma experiência unificada. Ele oferece funcionalidades sofisticadas para desenvolvedores SQL, consulta sob demanda sem servidor, suporte de aprendizado de máquina, a capacidade de incorporar o Spark nativamente, blocos de anotações colaborativos e integração de dados em um único serviço. Os desenvolvedores podem escolher entre uma variedade de linguagens compatíveis (por exemplo, C#, SQL, Scala e Python) por meio de diferentes mecanismos.

Alguns dos principais recursos do Azure Synapse Analytics incluem:

- Análise de SQL com pools (totalmente provisionados) e sob demanda (sem servidor).
- Spark com suporte total para Python, Scala, C# e SQL.
- Fluxo de dados com experiência de transformação de big data sem código.
- Integração e orquestração de dados para integrar os dados e operacionalizar o desenvolvimento de código.
- Uma versão nativa de nuvem do **Processamento analítico/transacional híbrido (HTAP)**, fornecida pelo Azure Synapse Link.

Para acessar todas as funcionalidades mencionadas anteriormente, o Azure Synapse Studio fornece uma única interface do usuário unificada da Web.

Este único serviço de dados integrado é vantajoso para as empresas, pois acelera a entrega de BI, IA, aprendizado de máquina, Internet das Coisas e aplicações inteligentes.

O Azure Synapse Analytics pode obter e fornecer insights de todos os seus dados no data warehouse e nos sistemas de análise de big data a velocidades muito rápidas. Ele permite que os profissionais de dados usem linguagens familiares, como SQL, para consultar bancos de dados relacionais e não relacionais em escala de petabytes. Além disso, recursos avançados, como simultaneidade ilimitada, gerenciamento inteligente de workloads e isolamento de workloads, ajudam a otimizar a performance de todas as consultas para workloads de missão crítica.

## O que é isolamento de workloads?

Um dos principais recursos da execução de data warehouses corporativos em grande escala é o isolamento de workloads. Essa é a capacidade de garantir reservas de recursos em um cluster de computação para que várias equipes possam trabalhar nos dados sem se atrapalhar, conforme ilustrado na Figura 18.1:

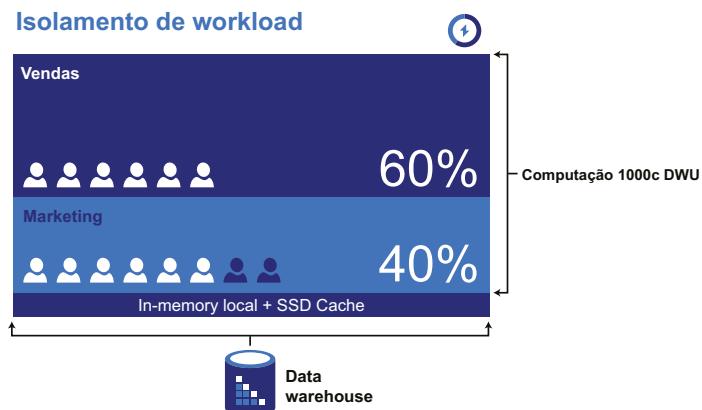


Figura 18.1: exemplo de isolamento de workloads

Você pode criar grupos de workloads em um cluster definindo alguns limites simples. Eles são ajustados automaticamente dependendo do workload e do cluster, mas sempre garantem uma experiência de qualidade para os usuários que executam os workloads. Consulte <https://techcommunity.microsoft.com/t5/data-architecture-blog/configuring-workload-isolation-in-azure-synapse-analytics/ba-p/1201739> para ler mais sobre como configurar o isolamento de workloads no Azure Synapse Analytics.

Para aproveitar ao máximo os benefícios do Azure Synapse, primeiro vamos ver os espaços de trabalho do Synapse e o Synapse Studio.

## Introdução aos espaços de trabalho do Synapse e ao Synapse Studio

No coração do Azure Synapse, está o espaço de trabalho. O espaço de trabalho é um recurso de alto nível que compõe sua solução de análise em um data warehouse. O espaço de trabalho do Synapse tem suporte para o processamento relacional e de big data.

O Azure Synapse fornece uma interface de usuário Web unificada para preparação, gerenciamento e armazenamento de dados, além de análise de big data, BI e tarefas de IA conhecidas como Synapse Studio. Juntamente com espaços de trabalho do Synapse, o Synapse Studio é um ambiente ideal para engenheiros de dados e cientistas de dados compartilharem e colaborarem com as soluções de análise, conforme mostrado na Figura 18.2:

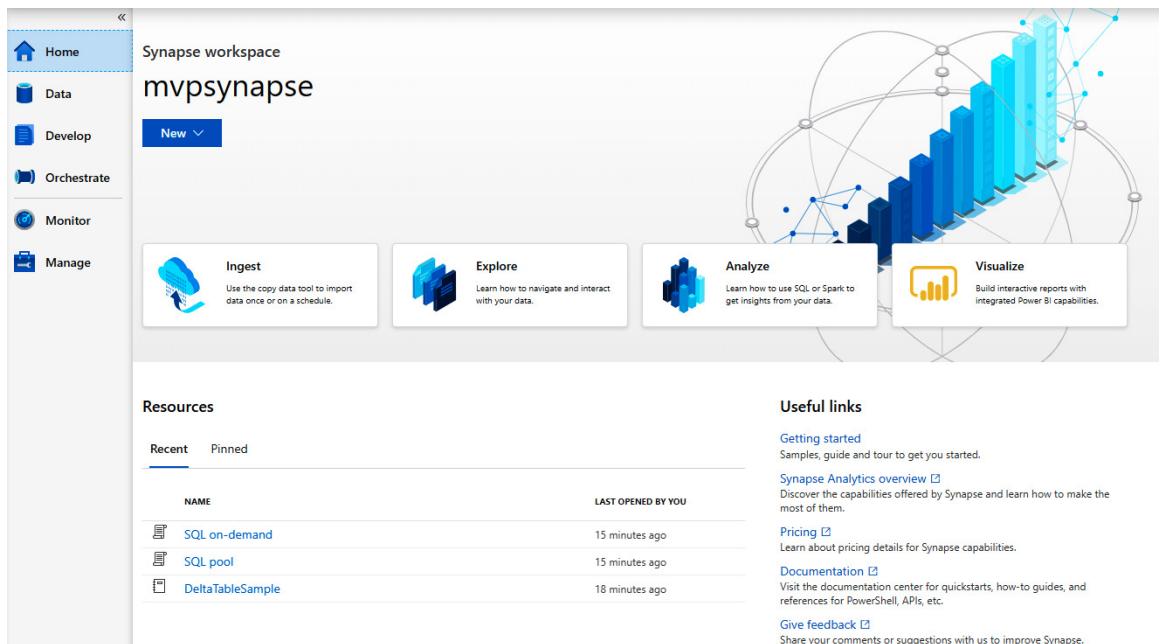


Figure 18.2: um espaço de trabalho do Synapse no Azure Synapse Studio

As seções a seguir destacam as funcionalidades, os principais recursos, os detalhes da plataforma e os serviços do usuário final de espaços de trabalho do Synapse e do Synapse Studio:

**Recursos:**

- Um data warehouse rápido, altamente elástico e seguro com performance e segurança líderes da indústria
- A capacidade de explorar o Azure Data Lake Storage e data warehouses usando a sintaxe T-SQL familiar em consultas SQL e SQL sob demanda (sem servidor)
- Apache Spark integrado ao Azure Machine Learning
- Integração de dados híbridos para acelerar a ingestão de dados e operacionalização do processo de análise (ingerir, preparar, transformar e fornecer)
- Geração de relatório de negócios e veiculação com integração do Power BI

**Principais recursos:**

- Criar e operacionalizar pipelines para ingestão e orquestração de dados.
- Explorar diretamente os dados em seu Azure Data Lake Storage ou data warehouse, bem como quaisquer conexões externas ao espaço de trabalho, usando o Synapse Studio.
- Escrever código usando blocos de anotações e editores de consultas T-SQL.
- Ferramenta de transformação de dados sem código, se preferir não escrever seu próprio código.
- Monitorar, proteger e gerenciar seus espaços de trabalho sem sair do ambiente.
- Experiência de desenvolvimento baseada na Web para as soluções de análise inteiras.
- O recurso de backup e restauração no pool de SQL do Azure Synapse permite que pontos de restauração sejam criados para facilitar a recuperação ou a cópia de um data warehouse para um estado anterior.
- A capacidade de executar consultas simultâneas de T-SQL em pools de SQL em petabytes de dados para fornecer ferramentas e aplicações de BI.
- O SQL sob demanda fornece consultas SQL sem servidor para facilitar a exploração e a análise de dados no Azure Data Lake Storage sem nenhuma configuração ou manutenção de infraestrutura.
- Atende a toda a gama de necessidades de análise, desde engenharia de dados até ciência de dados, usando uma variedade de linguagens, como Python, Scala, C# e Spark SQL.
- Pools Spark, que aliviam a configuração e manutenção complexas de clusters e simplificam o desenvolvimento de aplicações Spark e o uso de blocos de anotações do Spark.

- Oferece uma integração profunda entre o Spark e o SQL, permitindo que os engenheiros de dados preparam dados no Spark, escrevam os resultados processados no pool de SQL e usem qualquer combinação do Spark com o SQL para engenharia e análise de dados, com suporte interno para Azure Machine Learning.
- Capacidade de integração de dados altamente escalável e híbrida que acelera a ingestão e a operacionalização de dados por meio de pipelines de dados automatizados.
- Fornece um serviço integrado sem atritos com segurança, implantação, monitoramento e faturamento unificados.

## Plataforma

- Oferece suporte a computação provisionada e sem servidor. Exemplos de computação provisionada incluem computação SQL e computação Spark.
- A computação provisionada permite que as equipes segmentem os recursos de computação para que possam controlar o custo e o uso para se alinhar melhor com a estrutura organizacional.
- Por outro lado, a computação sem servidor permite que as equipes usem o serviço sob demanda sem provisionamento ou gerenciamento de infraestrutura subjacente.
- Integração profunda entre os mecanismos Spark e SQL.

Na seção a seguir, abordaremos os outros recursos do Azure Synapse, incluindo o Apache Spark para Synapse, o Synapse SQL, o SQL sob demanda, os pipelines do Synapse e o Azure Synapse Link para Cosmos DB.

## Apache Spark para Synapse

Para os clientes que querem o Apache Spark, o Azure Synapse tem suporte nativo pelo Azure Databricks e é totalmente gerenciado pelo Azure. A versão mais recente do Apache Spark será automaticamente disponibilizada aos usuários, juntamente com todos os patches de segurança. Você pode criar blocos de anotações rapidamente com sua linguagem escolhida, como Python, Scala, Spark SQL e .NET para Spark.

Se você usar o Spark no Azure Synapse Analytics, ele será fornecido como a oferta de Software como Serviço. Por exemplo, você pode usar o Spark sem configurar ou gerenciar seus próprios serviços, como uma rede virtual. O Azure Synapse Analytics cuidará da infraestrutura subjacente para você. Isso permite que você use o Spark imediatamente em seu ambiente do Azure Synapse Analytics.

Na próxima seção, vamos explorar o Synapse SQL.

## Synapse SQL

O Synapse SQL permite o uso do T-SQL para consultar e analisar dados. Há dois modelos para escolher:

1. Modelo totalmente provisionado
2. Modelo SQL sob demanda sem servidores

## SQL sob demanda

O SQL sob demanda fornece consultas SQL sem servidor. Isso permite mais facilidade de exploração e análise de dados no Azure Data Lake Storage sem nenhuma configuração ou manutenção de infraestrutura:

<b>TI tradicional</b>	<b>IaaS</b>	<b>PaaS</b>	<b>Sem servidor</b>	<b>SaaS</b>
Aplicação	Aplicação	Aplicação	Aplicação	Aplicação
Dados	Dados	Dados	Dados	Dados
Tempo de execução	Tempo de execução	Tempo de execução	Tempo de execução	Tempo de execução
Middleware	Middleware	Middleware	Middleware	Middleware
Sistema operacional	Sistema operacional	Sistema operacional	Sistema operacional	Sistema operacional
Virtualização	Virtualização	Virtualização	Virtualização	Virtualização
Servidores	Servidores	Servidores	Servidores	Servidores
Armazenamento	Armazenamento	Armazenamento	Armazenamento	Armazenamento
Rede	Rede	Rede	Rede	Rede



### Você gerencia

Tabela 18.1: comparação entre diferentes infraestruturas

## Principais recursos:

- Os analistas podem se concentrar em analisar os dados sem se preocupar em gerenciar nenhuma infraestrutura.
- Os clientes podem se beneficiar de um modelo de preços simples e flexível, pois eles só pagam pelo que usam.

- Ele usa a sintaxe familiar da linguagem T-SQL e o melhor otimizador de consultas SQL no mercado. O otimizador de consultas SQL é o cérebro por trás do mecanismo de consulta.
- Você pode facilmente escalar sua computação e armazenamento, independentemente uns dos outros, à medida que suas necessidades aumentam.
- Integre-se perfeitamente ao SQL Analytics Pool e ao Spark por meio de sincronização de metadados e conectores nativos.

## Pipelines do Synapse

Os pipelines do Synapse permitem que os desenvolvedores criem fluxos de trabalho completos para cenários de movimentação e processamento de dados. O Azure Synapse Analytics usa a tecnologia do **Azure Data Factory (ADF)** para fornecer recursos de integração de dados. Os principais recursos do ADF que são essenciais para o pipeline de data warehouse moderno estão disponíveis no Azure Synapse Analytics. Todos esses recursos são embrulhados com um modelo de segurança comum, **Controle de acesso baseado em função (RBAC)**, no espaço de trabalho do Azure Synapse Analytics.

A Figura 18.3 a seguir mostra um exemplo de um pipeline de dados e das atividades do ADF que estão diretamente integrados dentro do ambiente do Azure Synapse Analytics:



Figura 18.3: pipelines de dados no Azure Synapse Analytics

## Principais recursos:

- Serviços integrados de plataforma para gerenciamento, segurança, monitoramento e gerenciamento de metadados.
- Integração nativa entre Spark e SQL. Use uma única linha de código para ler e escrever com Spark de/no SQL analytics.
- A capacidade de criar uma tabela do Spark e consultá-la instantaneamente com o SQL Analytics sem definir um esquema.
- Ambiente “livre de chaves”. Com o logon único e a passagem do Azure Active Directory, nenhuma chave ou login é necessário para interagir com o **Azure Data Lake Storage (ADLS)**/bancos de dados.

Na próxima seção, abordaremos o Azure Synapse Link para Cosmos DB.

## Azure Synapse Link para Cosmos DB

O Azure Synapse Link é uma versão nativa da nuvem do HTAP. É uma extensão do Azure Synapse. Como aprendemos anteriormente, o Azure Synapse é um único serviço gerenciado para a execução de análises em data lakes e data warehouses, usando a computação sem servidor e provisionada. Com o Azure Synapse Link, esse alcance também pode ser estendido para fontes de dados operacionais.

O Azure Synapse Link elimina o gargalo que é encontrado em sistemas operacionais e analíticos tradicionais. O Azure Synapse viabiliza isso separando a computação do armazenamento em todos os seus serviços de dados. No lado transacional, o Cosmos DB é um serviço de banco de dados de alta performance, com replicação geográfica e multimodelo. No lado da análise, o Azure Synapse fornece escalabilidade ilimitada. Você pode escalar os recursos para transações e para análise de forma independente. Juntos, isso torna o HTAP nativo da nuvem uma realidade. Assim que o usuário indicar quais dados no Cosmos DB deseja disponibilizar para análise, os dados estarão disponíveis no Synapse. Ele extrai os dados operacionais que você deseja analisar e mantém automaticamente uma versão colunar orientada à análise deles. Como resultado, quaisquer alterações nos dados operacionais no Cosmos DB são continuamente atualizadas para os dados do Link e do Synapse.

O maior benefício do uso do Azure Synapse Link é que ele alivia a necessidade de processamento em lote agendado ou ter que criar e manter pipelines operacionais.

Como mencionado anteriormente, o Azure Synapse é a plataforma mais escolhida pelos arquitetos para migrar as soluções existentes do data warehouse para uma solução moderna de análise empresarial. Na próxima seção, também abordaremos considerações arquitetônicas comuns para migrar de uma solução herdada existente de data warehouse para o Azure Synapse Analytics.

## Migrar de sistemas herdados existentes para o Azure Synapse Analytics

Atualmente, muitas organizações estão migrando suas soluções herdadas de data warehouse para o Azure Synapse Analytics para obter os benefícios de alta disponibilidade, segurança, velocidade, escalabilidade, economia de custos e performance do Azure Synapse.

Para empresas que executam sistemas herdados de data warehouse, como o Netezza, a situação é ainda mais grave porque a IBM anunciou o fim do suporte ao Netezza (<https://www.ibm.com/support/pages/end-support-dates-netezza-5200-netezza-8x50z-series-and-netezza-10000-series-appliances>).

Há muitas décadas, algumas empresas escolheram o Netezza para gerenciar e analisar grandes volumes de dados. Atualmente, à medida que as tecnologias evoluem, os benefícios de ter uma solução de data warehouse baseada em nuvem superam os benefícios na infraestrutura local. O Azure Synapse é um serviço de análise ilimitado baseado em nuvem com um tempo incomparável de insight que acelera a entrega de BI, IA e aplicações inteligentes para empresas. Com sua arquitetura separada e multicloud de computação e armazenamento, o Azure Synapse pode ser escalado instantaneamente de maneiras impossíveis com sistemas herdados, como o Netezza.

Esta seção aborda as considerações arquitetônicas e a metodologia de alto nível para planejar, preparar e executar uma migração bem-sucedida de um sistema herdado existente de data warehouse para o Azure Synapse Analytics. Sempre que apropriado, serão indicados exemplos específicos e referências ao Netezza. Este capítulo não se destina a ser um manual passo a passo completo de migração, mas sim uma visão geral prática para ajudar no planejamento da migração e no escopo do projeto.

Ele também identifica alguns dos problemas comuns de migração e possíveis resoluções. Ele também fornece detalhes técnicos sobre as diferenças entre o Netezza e o Azure Synapse Analytics. Eles devem ser levados em consideração como parte de seu plano de migração.

### Por que você deve migrar seu data warehouse herdado para o Azure Synapse Analytics?

Ao migrar para o Azure Synapse Analytics, as empresas com sistemas herdados de data warehouse podem aproveitar as inovações mais recentes em tecnologias de nuvem e delegar tarefas, como manutenção de infraestrutura e atualização de plataforma para o Azure.

Os clientes que migraram para o Azure Synapse já estão colhendo muitos benefícios, incluindo o seguinte.

## Performance

O Azure Synapse Analytics oferece a melhor performance de banco de dados relacional usando técnicas como **processamento massivo paralelo (MPP)** e o armazenamento em cache automático in-memory. Para obter mais informações, consulte a arquitetura do Azure Synapse Analytics (<https://docs.microsoft.com/azure/synapse-analytics/sql-data-warehouse/massively-parallel-processing-mpp-architecture>).

## Velocidade

O data warehouse usa muitos processos. Envolve a ingestão de dados, a transformação de dados, a limpeza de dados, a agregação de dados, a integração de dados e a produção de relatórios e visualização de dados. Os muitos processos envolvidos na movimentação de dados de fontes originais para um data warehouse são complexos e interdependentes. Um único gargalo pode retardar todo o pipeline e um pico inesperado no volume de dados amplia a necessidade de velocidade. Quando a pontualidade dos dados é importante, o Azure Synapse Analytics atende à demanda por processamento rápido.

## Maior segurança e conformidade

O Azure é uma plataforma de nuvem disponível no mundo todo, altamente escalável e segura. Ele oferece muitos recursos de segurança, incluindo Azure Active Directory, RBAC, identidades gerenciadas e pontos de extremidade privados gerenciados. O Azure Synapse Analytics, que reside dentro do ecossistema do Azure, herda todos os benefícios mencionados anteriormente.

## Elasticidade e eficiência de custos

Em um data warehouse, as demandas de processamento de workloads podem oscilar. Às vezes, essas oscilações podem variar drasticamente entre picos e vales. Por exemplo, picos repentinos nos volumes de dados de vendas podem ocorrer durante as temporadas de férias. A elasticidade da nuvem permite que o Azure Synapse aumente e reduza rapidamente sua capacidade de acordo com a demanda, sem causar impacto na disponibilidade, estabilidade, performance e segurança da infraestrutura. O melhor de tudo é que você só paga o que usar.

## Infraestrutura gerenciada

Eliminar a sobrecarga de gerenciamento e operações do datacenter para o data warehouse permite que as empresas realoquem valiosos recursos para onde o valor for produzido e se concentrem em usar o data warehouse para fornecer as melhores informações e insights. Isso reduz o custo total de propriedade e oferece melhor controle de custos sobre suas despesas operacionais.

## Escalabilidade

O volume de dados em um data warehouse normalmente aumenta com o passar do tempo e conforme o histórico é coletado. O Azure Synapse Analytics pode ser escalado para corresponder a esse crescimento adicionando incrementalmente recursos à medida que os dados e workloads aumentam.

## Redução de custos

A operação de um datacenter herdado na infraestrutura local é dispendiosa (considerando os custos de servidores e hardware, rede, espaço físico, eletricidade, resfriamento e pessoal). Essas despesas podem ser minimizadas substancialmente com o Azure Synapse Analytics. Com a separação das camadas de computação e armazenamento, o Azure Synapse oferece uma relação de preço-performance muito lucrativa.

O Azure Synapse Analytics fornece a você uma verdadeira escalabilidade de nuvem pré-paga, sem a necessidade de reconfiguração complicada à medida que seus dados ou workloads aumentam.

Agora que você aprendeu por que é vantajoso migrar para o Azure Synapse Analytics, começaremos a abordar o processo de migração.

## O processo de migração em três etapas

Um projeto de migração de dados bem-sucedido começa com um plano bem projetado. Um plano eficaz é responsável pelos muitos componentes que precisam ser considerados, prestando atenção especial à arquitetura e à preparação de dados. Veja a seguir o plano de processo de migração em três etapas.

### Preparação

- Defina o escopo do que deve ser migrado.
- Crie um inventário de dados e processos para migração.
- Defina as alterações no modelo de dados (se houver).
- Defina o mecanismo de extração de dados de origem.
- Identifique ferramentas e serviços adequados do Azure (e de terceiros) a serem usados.
- Treine a equipe no início na nova plataforma.
- Configure a plataforma de destino do Azure.

### Migração

- Comece pequeno e simples.
- Automatize sempre que possível.
- Aproveite as ferramentas e recursos incorporados do Azure para reduzir o esforço de migração.
- Migre metadados para tabelas e visualizações.
- Migre dados históricos a serem mantidos.
- Migre ou refatore procedimentos e processos de negócios armazenados.
- Migre ou refatore processos de carga incremental de ETL/ELT.

## Pós-migração

- Monitore e documente todas as etapas do processo.
- Use a experiência adquirida para criar um modelo para futuras migrações.
- Recrie o modelo de dados, se necessário.
- Teste aplicações e ferramentas de consulta.
- Compare e otimize a performance da consulta.

Em seguida, vamos falar sobre os dois tipos de estratégias de migração.

## Os dois tipos de estratégias de migração

Os arquitetos devem começar o planejamento de migração avaliando o data warehouse existente para determinar qual estratégia de migração funciona melhor para sua situação. Existem dois tipos de estratégia de migração para considerar.

### Estratégia lift and shift

Para a estratégia lift and shift, o modelo de dados existente é migrado inalterado para a nova plataforma do Azure Synapse Analytics. Isso é feito para minimizar o risco e o Lift and shift é uma boa estratégia para ambientes herdados de data warehouse, como o Netezza, onde qualquer uma das condições a seguir é aplicável:

- Um único data mart deve ser migrado.
- Os dados já estão em um esquema de estrela ou floco de neve bem projetado.
- Há pressões imediatas de tempo e custos para migrar para um ambiente de nuvem moderno.

### Estratégia Reprojetar

Em cenários em que o data warehouse herdado evoluiu com o passar do tempo, pode ser essencial reprojetá-lo para manter os melhores níveis de performance ou oferecer suporte a novos tipos de dados. Isso pode incluir uma alteração no modelo de dados subjacente.

Para minimizar o risco, é recomendável migrar primeiro usando a estratégia lift and shift e, em seguida, modernizar gradualmente o modelo de dados do data warehouse no Azure Synapse Analytics usando a estratégia Reprojetar. Uma mudança completa no modelo de dados aumentará os riscos porque afetará a origem dos trabalhos de ETL do data warehouse e os data marts downstream.

Na próxima seção, ofereceremos algumas recomendações sobre como reduzir a complexidade de seu data warehouse herdado existente antes de migrar.

## Reducir a complexidade do seu data warehouse herdado existente antes de migrar

Na seção anterior, apresentamos as duas estratégias de migração. Como prática recomendada, durante a etapa de avaliação inicial, esteja ciente de formas de simplificar seu data warehouse existente e documentá-la. O objetivo é reduzir a complexidade de seu sistema herdado de data warehouse existente antes da migração para facilitar o processo de migração.

Veja a seguir algumas recomendações sobre como reduzir a complexidade de seu data warehouse herdado existente:

- **Remova e arquive tabelas não usadas antes de migrar:** evite migrar dados que não estão mais em uso. Isso ajudará a reduzir o volume geral de dados para migrar.
- **Converta data marts físicos em data marts virtuais:** minimize o que você tem para migrar, reduza o custo total de propriedade e melhore a agilidade.

Na próxima seção, vamos ver melhor por que você deve considerar a conversão de um data mart físico em um data mart virtual.

### Converter data marts físicos em data marts virtuais

Antes de migrar seu data warehouse herdado, considere converter seus data marts físicos atuais em data marts virtuais. Ao usar data marts virtuais, você pode eliminar os armazenamentos de dados físicos e os trabalhos de ETL para data marts sem perder nenhuma funcionalidade antes da migração. O objetivo aqui é reduzir o número de armazenamentos de dados para migrar, reduzir cópias de dados, reduzir o custo total de propriedade e melhorar a agilidade. Para conseguir isso, você precisará alternar de data marts físicos para virtuais antes de migrar seu data warehouse. Podemos considerar isso como uma etapa de modernização do data warehouse antes da migração.

### Desvantagens de data marts físicos

- Várias cópias dos mesmos dados
- Maior custo total de propriedade
- Difícil de mudar à medida que os trabalhos de ETL são afetados

## Vantagens de data marts virtuais

- Simplifica a arquitetura de data warehouse
- Não há necessidade de armazenar cópias de dados
- Mais agilidade
- Menor custo total de propriedade
- Usa otimização pushdown para aproveitar o poder do Azure Synapse Analytics
- Fácil de alterar
- Fácil de ocultar dados confidenciais

Na próxima seção, falaremos sobre como migrar os esquemas de data warehouse existentes para o Azure Synapse Analytics.

## Migrar esquemas de data warehouse existentes para o Azure Synapse Analytics

A migração dos esquemas de um data warehouse herdado existente envolve a migração de tabelas de preparação existentes, data warehouse herdado e esquemas de data mart dependentes.

Para ajudar a entender a magnitude e o escopo de sua migração de esquema, recomendamos que você crie um inventário de seu data warehouse e data mart existentes.

Aqui está uma checklist para você coletar as informações necessárias:

- Contagens de linhas
- Tamanho de dados de preparação, data warehouse e data mart: tabelas e índices
- Proporções de compactação de dados
- Configuração de hardware atual
- Tabelas (incluindo partições): identifique tabelas de pequena dimensão
- Tipos de dados
- Modos de exibição
- Índices
- Dependências de objeto
- Uso de objeto

- Funções: funções prontas para uso e **funções definidas pelo usuário (UDFs)**
- Procedimentos armazenados
- Requisitos de escalabilidade
- Projeções de crescimento
- Requisitos de workloads: usuários simultâneos

Com o inventário concluído, agora você pode tomar decisões sobre como escolher o esquema que deseja migrar. Basicamente, há quatro opções para o escopo da migração de esquema de data warehouse herdado:

1. Migrando um data mart por vez:

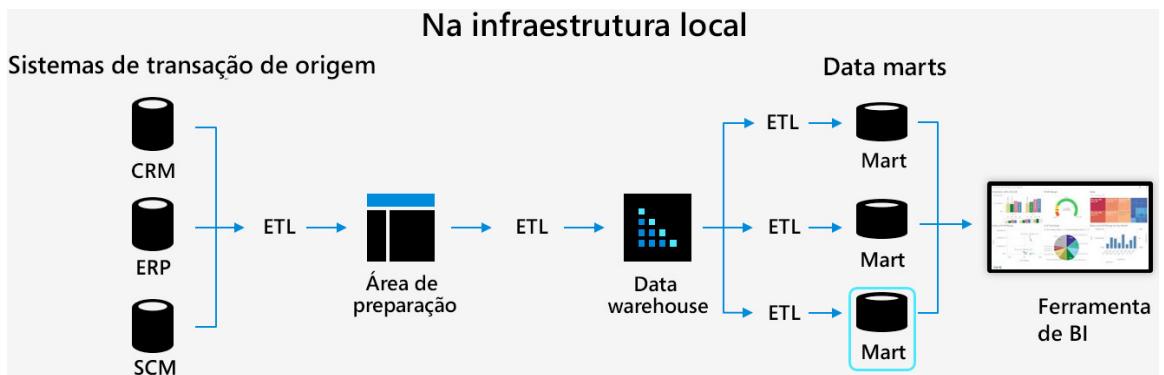


Figura 18.4: migrando um data mart por vez

2. Migrar todos os data marts de uma só vez, depois o data warehouse:

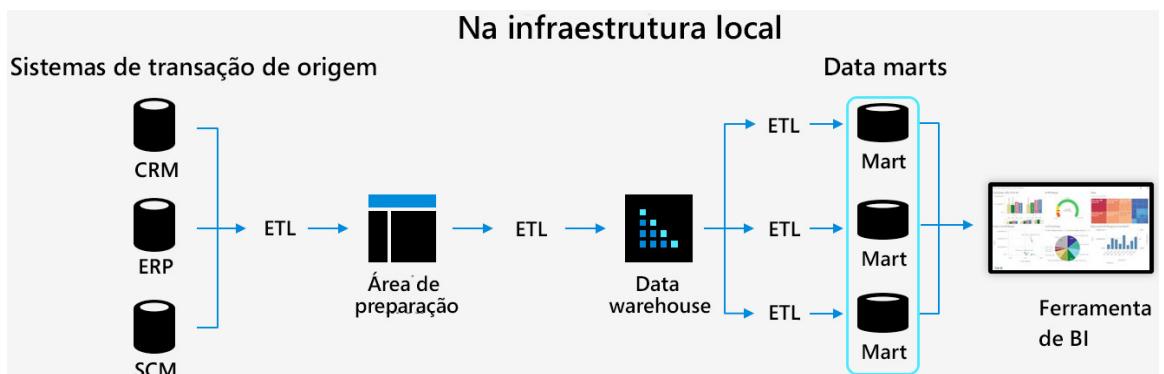


Figura 18.5: migrando todos os data marts de uma só vez, depois o data warehouse

### 3. Migrar o data warehouse e a área de preparação:

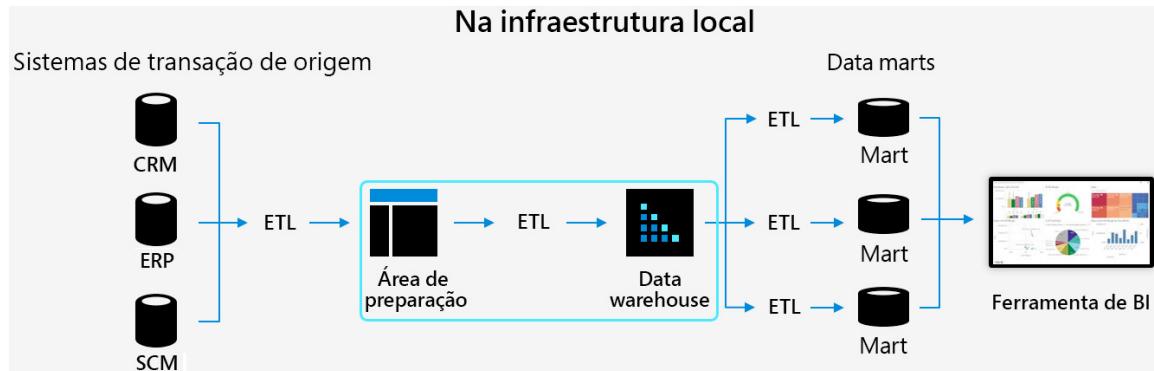


Figura 18.6: migrando o data warehouse e a área de preparação

### 4. Migrar tudo de uma vez:

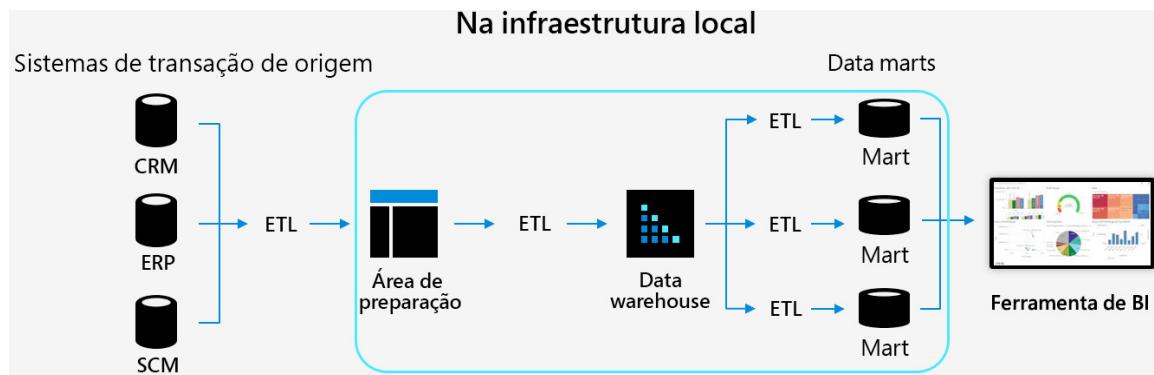


Figura 18.7: migrando tudo de uma vez

Tenha em mente ao escolher sua opção que o objetivo é alcançar um design de banco de dados físico que corresponderá ou excederá seu sistema de data warehouse herdado atual em performance e, preferencialmente, a um custo menor.

Para recapitular, veja algumas recomendações para a migração de esquema:

- Evite migrar objetos ou processos desnecessários.
- Considere o uso de data marts virtuais para reduzir ou eliminar o número de data marts físicos.
- Automatize sempre que possível. A implementação do DataOps deve ser considerada juntamente com a migração para o Azure Synapse.
- Use metadados de tabelas de catálogo do sistema de data warehouse herdado para gerar **linguagem de definição de dados (DDL)** para o Azure Synapse Analytics.
- Realize alterações necessárias no modelo de dados ou otimizações de mapeamento de dados no Azure Synapse Analytics.

Na próxima seção, falaremos sobre como migrar dados históricos de um data warehouse herdado para o Azure Synapse Analytics.

## Migrar dados históricos e processos de ETL de seu data warehouse herdado para o Azure Synapse Analytics

Depois que o escopo de migração do esquema for determinado, estaremos prontos para tomar decisões sobre como migrar os dados históricos.

As etapas para migrar dados históricos são as seguintes:

1. Crie tabelas de destino no Azure Synapse Analytics.
2. Migre dados históricos existentes.
3. Migre as funções e procedimentos armazenados, conforme necessário.
4. Migre a preparação e os processos de carga incremental (ETL/ELT) para dados de entrada.
5. Aplique qualquer opção de ajuste de performance que for necessária.

A Tabela 18.2 descreve as quatro opções de migração de dados e seus prós e contras:

Opção de migração de dados	Prós	Contras
Migrar dados do data mart primeiro, seguido por dados do data warehouse	<ul style="list-style-type: none"> <li>A migração de dados de um data mart por vez é uma abordagem incremental de baixo risco. Ela também fornece uma prova mais rápida do caso de negócios para os usuários finais do departamento de análise.</li> <li>A migração de ETL subsequente é limitada apenas aos dados nos data marts dependentes migrados.</li> </ul>	<ul style="list-style-type: none"> <li>Até que sua migração esteja concluída, você terá alguns dados que existem na infraestrutura local e no Azure.</li> <li>O processamento de ETL do data warehouse para data marts precisaria preencher o firewall e ser alterado para segmentar o Azure Synapse.</li> </ul>
Migrar dados do data warehouse primeiro, seguido por data marts.	<ul style="list-style-type: none"> <li>Todos os dados históricos do data warehouse são migrados.</li> </ul>	<ul style="list-style-type: none"> <li>Deixar data marts dependentes na infraestrutura local não é ideal, pois os ETLs precisam fluir dados de volta para o datacenter.</li> <li>Nenhuma oportunidade real para a migração de dados incremental.</li> </ul>
Migrar o data warehouse e os data marts juntos	<ul style="list-style-type: none"> <li>Todos os dados são migrados de uma só vez.</li> </ul>	<ul style="list-style-type: none"> <li>Risco possivelmente maior.</li> <li>Os ETLs provavelmente precisarão ser migrados juntos.</li> </ul>
Converter marts físicos em marts virtuais e migrar somente o data warehouse	<ul style="list-style-type: none"> <li>Não há armazenamentos de dados de data mart para migrar.</li> <li>Sem ETLs do data warehouse para os marts para migrar.</li> <li>Apenas os dados do data warehouse para migrar.</li> <li>Menos cópias de dados.</li> <li>Nenhuma perda na funcionalidade.</li> <li>Menor custo total de propriedade.</li> <li>Mais agilidade.</li> <li>Arquitetura de dados geral mais simples.</li> <li>Pode ser possível com visualizações no Azure Synapse.</li> </ul>	<ul style="list-style-type: none"> <li>Se as visualizações aninhadas não forem capazes de oferecer suporte a data marts virtuais, um software de virtualização de dados de terceiros no Azure provavelmente será necessário.</li> <li>Todos os marts precisariam ser convertidos antes que os dados do data warehouse fossem migrados.</li> <li>Os mapeamentos de marts virtuais e data warehouse para mart virtual precisarão ser transportados para o servidor de virtualização de dados no Azure e redirecionados para o Azure Synapse.</li> </ul>

Tabela 18.2: opções de migração de dados com seus prós e contras

Na próxima seção, falaremos sobre como migrar os processos de ETL para o Azure Synapse Analytics.

## Migrar os processos de ETL existentes para o Azure Synapse Analytics

Há várias opções disponíveis para migrar seus processos de ETL existentes para o Azure Synapse Analytics. A Tabela 18.3 descreve algumas das opções de migração de ETL com base em como os trabalhos de ETL existentes foram criados:

Como os trabalhos de ETL existentes são criados?	Opções de migração	Por que migrar e o que procurar
Código 3GL e scripts personalizados	<ul style="list-style-type: none"> <li>Planeje desenvolver novamente usando o ADF.</li> </ul>	<ul style="list-style-type: none"> <li>O código não fornece linhagem de metadados.</li> <li>Difícil de manter se os autores tiverem saído da empresa.</li> <li>Se as tabelas de preparo estiverem no data warehouse legado e o SQL for usado para transformar dados, resolva as diferenças com o T-SQL.</li> </ul>
Procedimentos armazenados que são executados em seu DBMS legado de data warehouse	<ul style="list-style-type: none"> <li>Planeje desenvolver novamente usando o ADF.</li> </ul>	<ul style="list-style-type: none"> <li>Prováveis diferenças significativas entre o data warehouse legado e o Azure Synapse.</li> <li>Sem linhagem de metadados.</li> <li>Isso precisa de uma avaliação cuidadosa, mas a principal vantagem pode ser o pipeline como abordagem de código, o que é possível com o ADF.</li> </ul>
Ferramenta de ETL gráfica (como Informatica ou Talend)	<ul style="list-style-type: none"> <li>Continue usando sua ferramenta de ETL existente e mude o destino para o Azure Synapse.</li> <li>Possivelmente mude para uma versão do Azure da sua ferramenta de ETL existente e transporte os metadados para executar trabalhos de ELT no Azure certificando-se de habilitar o acesso a fontes de dados na infraestrutura local.</li> <li>Controle a execução de serviços ETL usando o ADF.</li> </ul>	<ul style="list-style-type: none"> <li>Evita um novo desenvolvimento.</li> <li>Minimiza o risco e oferece migração mais rápida.</li> </ul>
Software de automação de data warehouse	<ul style="list-style-type: none"> <li>Continue usando sua ferramenta de ETL existente, mudando o destino e preparando para o Azure Synapse.</li> </ul>	<ul style="list-style-type: none"> <li>Evita um novo desenvolvimento.</li> <li>Minimiza o risco e oferece migração mais rápida.</li> </ul>

Tabela 18.3: opções de migração de ETL

Na próxima seção, falaremos sobre como redesenvolver processos de ETL escaláveis usando o ADF.

## Redesenvolver processos de ETL escaláveis usando o ADF

Outra opção para lidar com seus processos herdados existentes de ETL é redesenvolvê-los usando o ADF. O ADF é um serviço de integração de dados do Azure para criar fluxos de trabalho orientados por dados (conhecidos como pipelines) para orquestrar e automatizar o movimento e a transformação de dados. Você pode usar o ADF para criar e programar pipelines para ingerir dados de diferentes armazenamentos de dados. O ADF pode processar e transformar dados usando serviços de computação, como Spark, Azure Machine Learning, Azure HDInsight, Hadoop e Azure Data Lake Analytics:

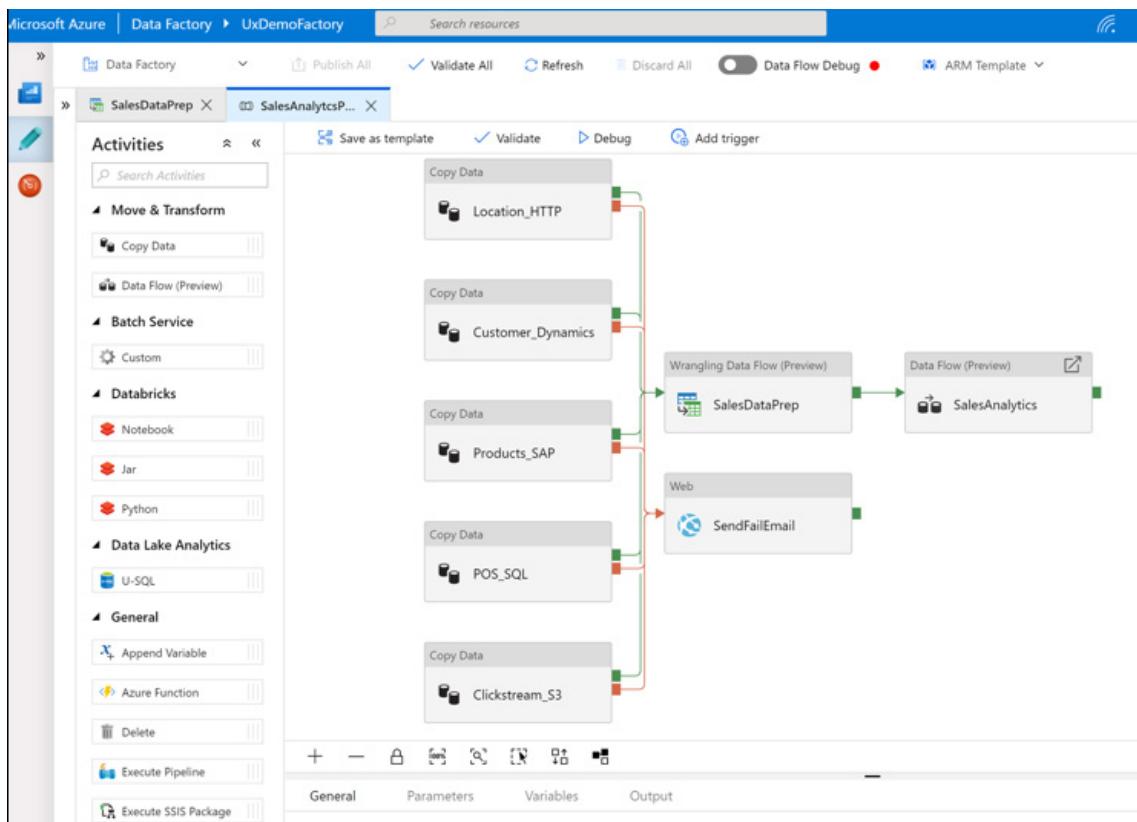


Figura 18.8: redesenvolvendo processos de ETL escaláveis usando o ADF

A próxima seção oferecerá algumas recomendações para migração de consultas, relatórios de BI, painéis e outras visualizações.

## Recomendações para migração de consultas, relatórios de BI, painéis e outras visualizações

A migração de consultas, relatórios de BI, painéis e outras visualizações do seu data warehouse herdado para o Azure Synapse Analytics é simples se o sistema herdado usar o SQL padrão.

No entanto, muitas vezes, esse não é o caso. Nessa situação, uma estratégia diferente deve ser tomada:

- Identifique os relatórios de alta prioridade para migrar primeiro.
- Use as estatísticas de uso para identificar os relatórios que nunca são usados.
- Evite migrar qualquer coisa que não esteja mais em uso.
- Depois de produzir a lista de relatórios para migrar, suas prioridades e os relatórios não usados a serem ignorados, confirme essa lista com os stakeholders.
- Para os relatórios que você está migrando, identifique precocemente as incompatibilidades para avaliar o esforço de migração.
- Considere a virtualização de dados para proteger as ferramentas de BI e aplicações contra alterações estruturais no modelo de dados de data warehouse e/ou data mart que pode ocorrer durante a migração.

## Problemas comuns de migração e resoluções

Durante o processo de migração, você pode encontrar certos problemas que precisa superar. Nesta seção, destacaremos alguns dos problemas comuns e forneceremos resoluções que você pode implementar.

### Problema 1: tipos de dados incompatíveis e soluções alternativas

A Tabela 18.4 mostra os tipos de dados de sistemas herdados de data warehouse que não têm suporte, bem como a solução adequada para o Azure Synapse Analytics.

Tipo de dados incompatível	Solução alternativa para o Azure Synapse Analytics
geometry	varbinary
geography	varbinary
hierarchyid	nvarchar(4000)
image	varbinary
text	varchar
ntext	nvarchar
sql_variant	Divida a coluna em várias colunas fortemente tipificadas
tabela	Converta em tabelas temporárias
timestamp	Código de retrabalho para usar datetime2 e a função CURRENT_TIMESTAMP
xml	varchar
user-defined type	Converta de volta para o tipo de dados nativo quando possível

Tabela 18.4: tipos de dados sem suporte e soluções alternativas apropriadas no Azure Synapse Analytics

## Problema 2: Diferenças de tipo de dados entre o Netezza e o Azure Synapse

A Tabela 18.5 mapeia os tipos de dados do Netezza para seus tipos de dados equivalentes do Azure Synapse:

Tipo de dados Netezza	Tipo de dados do Azure Synapse
BIGINT	BIGINT
BINARY VARYING(n)	VARBINARY(n)
BOOLEAN	BIT
BYTEINT	TINYINT
CHARACTER VARYING(n)	VARCHAR(n)
CHARACTER(n)	CHAR(n)
DATE	DATE
DECIMAL(p,s)	DECIMAL(p,s)
DOUBLE PRECISION	FLOAT
FLOAT(n)	FLOAT(n)
INTEGER	INT
INTERVAL	Atualmente, os tipos de dados INTERVAL não são diretamente compatíveis no Azure Synapse, mas podem ser calculados usando funções temporais, como DATEDIFF
MONEY	MONEY
NATIONAL CHARACTER VARYING(n)	NVARCHAR(n)
NATIONAL CHARACTER(n)	NCHAR(n)
NUMERIC(p,s)	NUMERIC(p,s)
REAL	REAL
SMALLINT	SMALLINT
ST_GEOGRAPHY(n)	Tipos de dados espaciais, como ST_GEOGRAPHY não são atualmente compatíveis com o Azure Synapse, mas os dados podem ser armazenados como VARCHAR ou VARBINARY
TIME	TIME
TIME WITH TIME ZONE	DATETIMEOFFSET
TIMESTAMP	DATETIME

Tabela 18.5: tipos de dados do Netezza e seus equivalentes do Azure Synapse

### Problema 3: Diferenças de restrição de integridade

Preste muita atenção às diferenças de restrição de integridade entre seu data warehouse ou data mart herdado e o Azure Synapse Analytics. Na Figura 18.9, o lado esquerdo representa o antigo sistema herdado de data warehouse e restrições de chave primária, e o lado direito mostra o novo ambiente do Azure Synapse Analytics:

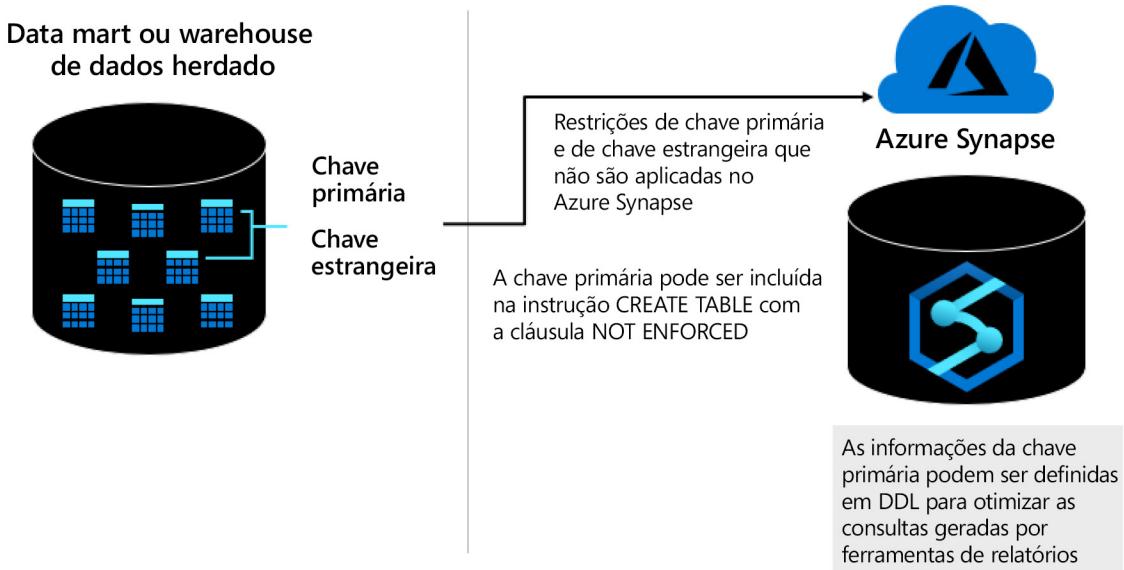


Figura 18.9: diferenças de restrição de integridade

Nas próximas seções, forneceremos uma cobertura abrangente sobre como resolver outras incompatibilidades do SQL comuns durante a migração de um data warehouse herdado para o Azure Synapse Analytics.

## Incompatibilidades do SQL comuns e resoluções

Esta seção fornecerá detalhes técnicos sobre incompatibilidades de SQL e resoluções comuns entre os sistemas herdados de data warehouse e o Azure Synapse Analytics. A seção explicará e comparará as diferenças e fornecerá resoluções usando uma tabela de referência rápida que você pode consultar posteriormente ao iniciar seu projeto de migração.

Os tópicos que abordaremos são os seguintes:

- Diferenças e resoluções de **linguagem de definição de dados (DDL)** SQL
- Diferenças e resoluções de **linguagem de manipulação de dados (DML)** SQL
- Diferenças e resoluções de **linguagem de controle de dados (DCL)** SQL
- Diferenças do SQL estendido e soluções alternativas

## Diferenças e resoluções de DDL SQL

Nesta seção, abordaremos as diferenças e as resoluções de DDL SQL entre os sistemas herdados de data warehouse e o Azure Synapse Analytics.

Problemas	Sistema de data warehouse legado	Resoluções
Tipos de tabelas proprietárias	<ul style="list-style-type: none"> <li>No sistema legado, identifique qualquer uso de tipos de tabelas proprietárias.</li> </ul>	<ul style="list-style-type: none"> <li>Migre para tabelas padrão no Azure Synapse Analytics.</li> <li>Para séries temporais, índice ou partição na coluna de data/hora.</li> <li>A filtragem adicional precisará ser adicionada às consultas temporais relevantes.</li> </ul>
Modos de exibição	<ul style="list-style-type: none"> <li>Identifique visualizações de tabelas de catálogo e scripts DDL.</li> </ul>	<ul style="list-style-type: none"> <li>Exibições com extensões ou funções proprietárias do SQL terão que ser reescritas.</li> <li>O Azure Synapse Analytics também oferece suporte a visualizações materializadas. Ele as mantém e as atualiza automaticamente.</li> </ul>
Nulos	<ul style="list-style-type: none"> <li>Os valores NULL podem ser gerenciados de forma diferente nos bancos de dados SQL legados. Por exemplo, na Oracle, uma string vazia é equivalente a um valor NULL.</li> <li>Alguns DBMSes possuem funções SQL proprietárias para lidar com NULLs; por exemplo, NVL na Oracle.</li> </ul>	<ul style="list-style-type: none"> <li>Gere consultas SQL para testar valores NULL.</li> <li>Teste relatórios que incluem colunas anuláveis.</li> </ul>

Tabela 18.6: diferenças de DDL SQL entre sistemas herdados e o Azure Synapse

## Diferenças e resoluções de DML SQL

Nesta seção, abordaremos as diferenças e as resoluções de DML SQL entre os sistemas herdados de data warehouse e o Azure Synapse Analytics:

Função	Netezza	Equivalente ao Azure Synapse
STRPOS	SELECT STRPOS('ABCDEFG', 'BCD') ...	SELECT CHARINDEX('BCD', 'ABCDEFG') ...
AGE	SELECT AGE('25-12-1940', '25-12-2020') FROM ...	SELECT DATEDIFF(dia, '1940-12-25', '2020-12-25') FROM...
NOW()	NOW()	CURRENT_TIMESTAMP
SEQUENCE	CREATE SEQUENCE...	Reescreva usando colunas de identidade no Azure Synapse Analytics.
UDF	Os UDFs da Netezza são escritos em nzLua ou C++.	Reescreva em T-SQL no Azure Synapse Analytics.
Procedimentos armazenados	Os procedimentos armazenados da Netezza são escritos em NZPLSQL (com base em Postgres PL/pgSQL).	Reescreva em T-SQL no Azure Synapse Analytics.

Tabela 18.7: diferenças de DML SQL entre o Netezza e o Azure Synapse

Em seguida, falaremos sobre as diferenças e as resoluções de DCL SQL entre os sistemas herdados de data warehouse e o Azure Synapse Analytics.

## Diferenças e resoluções de DCL SQL

Nesta seção, abordaremos as diferenças e as resoluções de DCL SQL entre os sistemas herdados de data warehouse e o Azure Synapse Analytics. O Netezza oferece suporte a duas classes de direitos de acesso: administrador e objeto. A Tabela 18.8 mapeia os direitos de acesso do Netezza e seus equivalentes do Azure Synapse para referência rápida.

## Mapear privilégios de administrador do Netezza para os equivalentes do Azure Synapse

Tabela 18.8 mapeia os privilégios de administrador do Netezza para os equivalentes do Azure Synapse:

Privilégio de administrador	Descrição	Equivalente ao Azure Synapse
Backup	Permite que os usuários criem backups e executem o comando nzbackup.	Recurso de backup e restauração no pool de SQL do Azure Synapse
[Criar] agregado	Permite que o usuário crie agregados definidos pelo usuário (UDAs). A permissão para operar em UDAs existentes é controlada por privilégios de objeto.	O recurso CREATE FUNCTION do Azure Synapse incorpora a funcionalidade agregada de Netezza
[Criar] banco de dados	Permite que o usuário crie bancos de dados. A permissão para operar em bancos de dados existentes é controlada por privilégios de objeto.	CREATE DATABASE
[Criar] tabela externa	Permite que o usuário crie tabelas externas. A permissão para operar em tabelas existentes é controlada por privilégios de objeto.	CREATE TABLE
[Criar] função	Permite que o usuário crie UDFs. A permissão para operar em UDFs existentes é controlada por privilégios de objeto.	CREATE FUNCTION
[Criar] grupo	Permite que o usuário crie grupos. A permissão para operar em grupos existentes é controlada por privilégios de objeto.	CREATE ROLE
[Criar] índice	Apenas para uso do sistema. Os usuários não podem criar índices.	CREATE INDEX
[Criar] biblioteca	Permite que o usuário crie bibliotecas compartilhadas. A permissão para operar em bibliotecas compartilhadas existentes é controlada por privilégios de objeto.	N/A
[Criar] visualização materializada	Permite que o usuário crie visualizações materializadas.	CREATE VIEW
[Criar] procedimento	Permite que o usuário crie procedimentos armazenados. A permissão para operar em procedimentos armazenados existentes é controlada por privilégios de objeto.	CREATE PROCEDURE
[Criar] esquema	Permite que o usuário crie esquemas. A permissão para operar em esquemas existentes é controlada por privilégios de objeto.	CREATE SCHEMA

Privilégio de administrador	Descrição	Equivalente ao Azure Synapse
[Criar] sequência	Permite que o usuário crie sequências de bancos de dados.	N/A
[Criar] sinônimo	Permite que o usuário crie sinônimos.	CREATE SYNONYM
[Criar] tabela	Permite que o usuário crie tabelas. A permissão para operar em tabelas existentes é controlada por privilégios de objeto.	CREATE TABLE
[Criar] tabela temporária	Permite que o usuário crie tabelas temporárias. A permissão para operar em tabelas existentes é controlada por privilégios de objeto.	CREATE TABLE
[Criar] usuário	Permite que o usuário crie usuários. A permissão para operar em usuários existentes é controlada por privilégios de objeto.	CREATE USER
[Criar] visualização	Permite que o usuário crie visualizações. A permissão para operar em visualizações existentes é controlada por privilégios de objeto.	CREATE VIEW
[Gerenciar] hardware	Permite que o usuário visualize o status do hardware, gerencie SPUs, gerencie a topologia e o espelhamento e execute testes de diagnóstico.	Gerenciado automaticamente por meio do portal do Azure no Azure Synapse
[Gerenciar] segurança	Permite que o usuário execute comandos e operações relacionadas ao gerenciamento e à configuração de bancos de dados históricos; gerenciamento de objetos de segurança multinível, incluindo especificação de segurança para usuários e grupos; e gerenciamento de chaves de bancos de dados para a assinatura digital de dados de auditoria.	Gerenciado automaticamente por meio do portal do Azure no Azure Synapse
[Gerenciar] Sistema	Permite que o usuário faça as seguintes operações de gerenciamento: iniciar/parar/pausar/retomar o sistema, abortar sessões e exibir o mapa de distribuição, as estatísticas do sistema e os logs. O usuário pode emitir estes comandos: nzsystem, nzstate, nzstats e nzsession.	Gerenciado automaticamente por meio do portal do Azure no Azure Synapse
Restauração	Permite que o usuário restaure o sistema. O usuário pode executar o comando nzrestore.	Gerenciado automaticamente no Azure Synapse
Sem vedação	Permite que o usuário crie ou altere um UDF ou agregado para ser executado no modo sem vedação.	N/A

Tabela 18.8: privilégios de administrador do Netezza e seus equivalentes do Azure Synapse

## Mapear privilégios de objeto do Netezza para seus equivalentes do Azure Synapse

A Tabela 18.9 mapeia os privilégios de objeto do Netezza para os equivalentes do Azure Synapse para referência rápida:

Privilégio de objeto	Descrição	Equivalente ao Azure Synapse
Abortar	Permite que o usuário aborte sessões. Aplica-se a grupos e usuários.	KILL DATABASE CONNECTION
Alterar	Permite que o usuário modifique os atributos de objeto. Aplica-se a todos os objetos.	ALTER
Excluir	Permite que o usuário exclua as linhas da tabela. Aplica-se apenas a tabelas.	DELETE
Soltar	Permite que o usuário solte objetos. Aplica-se a todos os tipos de objeto.	DROP
Executar	Permite que o usuário execute UDFs, UDA's ou procedimentos armazenados.	EXECUTE
GenStats	Permite que o usuário gere estatísticas em tabelas ou bancos de dados. O usuário pode executar o comando GENERATE STATISTICS.	Gerenciado automaticamente no Azure Synapse
Preparar	Permite que o usuário recupere o espaço em disco para linhas excluídas ou desatualizadas e reorganize uma tabela pelas chaves de organização ou migre dados para tabelas que têm várias versões armazenadas.	Gerenciado automaticamente no Azure Synapse
Inserir	Permite que o usuário insira linhas em uma tabela. Aplica-se apenas a tabelas.	INSERT
Listar	Permite que o usuário exiba um nome de objeto, seja em uma lista ou de outra maneira. Aplica-se a todos os objetos.	LIST
Selecionar	Permite que o usuário selecione (ou consulte) linhas dentro de uma tabela.	SELECT
Truncar	Permite que o usuário exclua todas as linhas de uma tabela. Aplica-se apenas a tabelas.	TRUNCATE
Atualizar	Permite que o usuário modifique as linhas da tabela. Aplica-se apenas a tabelas.	UPDATE

Tabela 18.9: privilégios de objeto do Netezza e seus equivalentes do Azure Synapse

## Diferenças do SQL estendido e soluções alternativas

A Tabela 18.10 descreve as diferenças do SQL estendido e possíveis soluções alternativas ao migrar para o Azure Synapse Analytics:

SQL extensão	Descrição	Como migrar
UDFs	<ul style="list-style-type: none"> <li>Pode conter código arbitrário</li> <li>Pode ser codificado em várias linguagens (como Lua e Java)</li> <li>Pode ser chamado dentro de uma instrução SELECT do SQL da mesma forma que funções incorporadas como SUM() e AVG() são usadas</li> </ul>	<ul style="list-style-type: none"> <li>Use CREATE FUNCTION e recodifique no T-SQL.</li> </ul>
Armazenado procedimentos	<ul style="list-style-type: none"> <li>Pode conter uma ou mais instruções SQL, bem como a lógica processual em torno dessas instruções SQL</li> <li>Implementado em uma linguagem padrão (como Lua) ou em uma linguagem proprietária (como Oracle PL/SQL)</li> </ul>	<ul style="list-style-type: none"> <li>Recodifique em T-SQL.</li> <li>Algumas ferramentas de terceiros que podem ajudar na migração:</li> <li>Datometry</li> <li>WhereScape</li> </ul>
Gatilhos	<ul style="list-style-type: none"> <li>Sem suporte do Azure Synapse</li> </ul>	<ul style="list-style-type: none"> <li>A funcionalidade equivalente pode ser alcançada usando outras partes do ecossistema do Azure. Por exemplo, para streaming de dados de entrada, utilize o Azure Stream Analytics.</li> </ul>
Análise no banco de dados	<ul style="list-style-type: none"> <li>Sem suporte do Azure Synapse</li> </ul>	<ul style="list-style-type: none"> <li>Executar Advanced Analytics, como modelos de aprendizado de máquina em grande escala, para usar o Azure Databricks.</li> <li>O Azure Synapse abre a possibilidade de realizar funções de Machine Learning com Spark MLlib.</li> <li>Como alternativa, migre para o banco de dados SQL do Azure e use a função PREDICT.</li> </ul>
Geoespacial tipos de dados	<ul style="list-style-type: none"> <li>Sem suporte do Azure Synapse</li> </ul>	<ul style="list-style-type: none"> <li>Armazenar dados geoespaciais, como latitude/longitude, e formatos populares, como Well-KnownText (WKT) e WKB (binário bem conhecido), em colunas VARCHAR ou VARBINARY e acessá-los diretamente usando ferramentas de cliente geoespaciais.</li> </ul>

Tabela 18.10: diferenças do SQL estendido e soluções alternativas

Nesta seção, falamos sobre problemas comuns de migração que os arquitetos podem encontrar durante um projeto de migração e possíveis soluções. Na próxima seção, vamos ver considerações de segurança às quais um arquiteto deve estar atento.

## Considerações sobre segurança

Proteger seus ativos de dados é fundamental em qualquer sistema de data warehouse. Ao planejar um projeto de migração de data warehouse, a segurança, o gerenciamento de acesso do usuário, o backup e a restauração também devem ser levados em consideração. Por exemplo, a criptografia de dados pode ser obrigatória para regulamentos da indústria e do governo, como HIPAA, PCI e FedRAMP, bem como em indústrias não regulamentadas.

O Azure inclui muitos recursos e funções como padrão que, tradicionalmente, precisam ser personalizados em produtos herdados de data warehouse. O Azure Synapse oferece suporte à criptografia de dados em repouso e em movimento como padrão.

### Criptografia de dados em repouso

- A **criptografia de dados transparente (TDE)** pode ser habilitada para criptografar e descriptografar dinamicamente dados do Azure Synapse, logs e backups associados.
- O armazenamento de dados do Azure também pode criptografar automaticamente dados que não são de banco de dados.

### Dados em movimento

Todas as conexões com o Azure Synapse Analytics são criptografadas por padrão, usando protocolos padrão da indústria, como TLS e SSH.

Além disso, a **máscara de dados dinâmicos (DDM)** pode ser usada para ofuscar dados de determinadas classes de usuários com base em regras de mascaramento de dados.

Como prática recomendada, se o data warehouse herdado tiver uma hierarquia complexa de permissões, usuários e funções, considere o uso de técnicas de automação em seu processo de migração. Você pode usar metadados existentes do seu sistema herdado para gerar o SQL necessário para migrar usuários, grupos e privilégios no Azure Synapse Analytics.

Na seção final deste capítulo, revisaremos algumas das ferramentas que os arquitetos podem escolher para ajudar a migrar de sistemas herdados de data warehouse para o Azure Synapse Analytics.

## Ferramentas para ajudar a migrar para o Azure Synapse Analytics

Agora que abordamos o planejamento e a preparação e uma visão geral do processo de migração, vamos ver as ferramentas que você pode usar para migrar seu data warehouse herdado para o Azure Synapse Analytics. As ferramentas que abordaremos são:

- ADF
- Azure Data Warehouse Migration Utility
- Serviços da Microsoft para transferência de dados físicos
- Serviços da Microsoft para ingestão de dados

Vamos começar.

### ADF

O ADF é um serviço de integração de dados totalmente gerenciado, pré-pago e híbrido para processamento de ETL em escala de nuvem. Ele oferece os seguintes recursos:

- Processa e analisa dados in-memory e em paralelo para escalar e maximizar a taxa de transferência
- Cria pipelines de migração de data warehouse que orquestram e automatizam o movimento de dados, a transformação de dados e o carregamento de dados no Azure Synapse Analytics
- Também pode ser usado para modernizar seu data warehouse, ingerindo dados no Azure Data Lake Storage, processando e analisando dados em escala e carregando dados em um data warehouse
- Oferece suporte a interfaces de usuário baseadas em função para mapear fluxos de dados para profissionais de TI e processar dados self-service para usuários corporativos
- Pode se conectar a vários armazenamentos de dados que abrangem aplicações de datacenter, nuvens e SaaS
- Mais de 90 conectores nativos e livres de manutenção disponíveis (<https://azure.microsoft.com/services/data-factory>)
- Pode misturar e associar processamento e mapeamento de fluxos de dados no mesmo pipeline para preparar dados em escala
- A orquestração de ADF pode controlar a migração de data warehouse para o Azure Synapse Analytics
- Pode executar pacotes SSIS de ETL

## Azure Data Warehouse Migration Utility

O Azure Data Warehouse Migration Utility pode migrar dados de um data warehouse baseado no SQL Server na infraestrutura local para o Azure Synapse. Ele oferece os seguintes recursos:

- Usa uma abordagem semelhante a um assistente para realizar uma migração lift and shift de esquema e dados de um data warehouse baseado no SQL Server na infraestrutura local.
- Você pode selecionar o banco de dados na infraestrutura local que contém as tabelas que deseja exportar para o Azure Synapse. Em seguida, você pode selecionar as tabelas que deseja migrar e migrar o esquema.
- Gera automaticamente o código T-SQL necessário para criar um banco de dados vazio equivalente e tabelas no Azure Synapse. Depois de fornecer detalhes de conexão para o Azure Synapse, você pode executar o T-SQL gerado para migrar o esquema.
- Após a criação do esquema, você pode usar o utilitário para migrar os dados. Isso exporta os dados do data warehouse baseado no SQL Server na infraestrutura local e gera comandos **programa de cópia em massa (BCP)** para carregar esses dados no Azure Synapse.

## Serviços da Microsoft para transferência de dados físicos

Nesta seção, veremos os serviços comuns da Microsoft que podem ser usados para transferência física de dados, incluindo Azure ExpressRoute, AzCopy e Azure Databox.

### Azure ExpressRoute

O Azure ExpressRoute permite fazer conexões privadas entre seus datacenters e o Azure sem passar pela Internet pública. Ele oferece os seguintes recursos:

- Largura de banda de até 100 Gbps
- Baixa latência
- Conecta-se diretamente à sua **rede de longa distância (WAN)**
- Conexões privadas ao Azure
- Maior velocidade e confiabilidade

### AzCopy

O AzCopy é uma ferramenta de linha de comando para copiar arquivos e blobs de/para contas de armazenamento. Ele oferece os seguintes recursos:

- Capacidade de copiar dados de/para o Azure por meio da Internet.
- Uma combinação de AzCopy com a largura de banda do ExpressRoute necessária pode ser uma solução ideal para transferência de dados para o Azure Synapse.

## Azure Data Box

O Azure Data Box permite que você transfira grandes volumes de dados para o Azure de forma rápida, confiável e econômica. Ele oferece os seguintes recursos:

- Capaz de transferir grandes volumes de dados (dezenas de terabytes a centenas de terabytes)
- Sem restrições de conectividade de rede
- Ótimo para a migração única e a transferência inicial em massa

## Serviços da Microsoft para ingestão de dados

Nesta seção, veremos os serviços comuns da Microsoft que podem ser usados para a ingestão de dados, incluindo:

- PolyBase
- BCP
- API SqlBulkCopy
- SQL Standard

### PolyBase (método recomendado)

O PolyBase fornece o carregamento de dados em massa no Azure Synapse Analytics mais rápido e escalável. Ele oferece os seguintes recursos:

- Usa o carregamento paralelo para fornecer a taxa de transferência mais rápida
- Pode ler de arquivos simples no Armazenamento de Blobs do Azure ou de fontes de dados externas por meio de conectores
- Totalmente integrado ao ADF
- CREATE TABLE AS ou INSERT ... SELECT
- Pode definir uma tabela de preparo como tipo HEAP para carga rápida
- Suporte para linhas com até 1 MB de comprimento

## BCP

O BCP pode ser usado para importar e exportar dados de qualquer ambiente do SQL Server, incluindo o Azure Synapse Analytics. Ele oferece os seguintes recursos:

- Oferece suporte a linhas com mais de 1 MB de comprimento
- Originalmente desenvolvido para versões anteriores do Microsoft SQL Server

Consulte <https://docs.microsoft.com/sql/tools/bcp-utility> ler mais sobre o utilitário BCP.

## API SqlBulkCopy

A API SqlBulkCopy é a API equivalente à funcionalidade do BCP. Ele oferece os seguintes recursos:

- Permite a implementação de processos de carga de forma programática
- Capacidade de carregar tabelas do SQL Server em massa com dados de fontes selecionadas

Consulte <https://docs.microsoft.com/dotnet/api/system.data.sqlclient.sqlbulkcopy> ler mais sobre esta API.

## Supporte do SQL Standard

O Azure Synapse Analytics oferece suporte ao SQL Standard, incluindo a capacidade de:

- Carregar linhas individuais ou resultados de instruções **SELECT** em tabelas de data warehouse.
- Insira dados em massa de dados extraídos de fontes de dados externas em tabelas usando as instruções **INSERT ... SELECT** no PolyBase.

Esta seção forneceu as considerações arquitetônicas e a metodologia de alto nível para planejar, preparar e executar uma migração bem-sucedida de um sistema herdado existente de data warehouse para o Azure Synapse Analytics. Ele contém muitas informações que você pode consultar posteriormente ao iniciar seu projeto de migração para o Azure Synapse Analytics.

## Resumo

O Azure Synapse Analytics é um serviço de análise ilimitado com um tempo incomparável de insight que acelera a entrega de BI, IA e aplicações inteligentes para empresas. Você obterá muitos benefícios ao migrar o data warehouse herdado para o Azure Synapse Analytics, incluindo performance, velocidade, maior segurança e conformidade, elasticidade, infraestrutura gerenciada, escalabilidade e redução de custos.

Com o Azure Synapse, profissionais de dados com diversas habilidades podem colaborar, gerenciar e analisar seus dados mais importantes com facilidade, tudo dentro do mesmo serviço. Desde a integração do Apache Spark com o poderoso e confiável mecanismo SQL até a integração e gerenciamento de dados sem código, o Azure Synapse é criado para todos os profissionais de dados.

Este capítulo forneceu as considerações arquitetônicas e a metodologia de alto nível necessária para preparar e executar a migração de um sistema herdado existente de data warehouse para o Azure Synapse Analytics.

Projetos de migração de dados bem-sucedidos começam com um plano bem projetado. Um plano eficaz é responsável pelos muitos componentes que precisam ser considerados, prestando atenção especial à arquitetura e à preparação de dados.

Depois de migrar para o Azure Synapse com êxito, você pode explorar outras tecnologias da Microsoft no rico ecossistema analítico do Azure para modernizar ainda mais sua arquitetura de data warehouse.

Aqui estão algumas ideias para considerar:

- Descarregue suas áreas de preparação e processamento de ELT para o Azure Data Lake Storage e o ADF.
- Crie produtos de dados confiáveis uma vez no formato comum de modelo de dados e consuma em todos os lugares, não apenas em seu data warehouse.
- Habilite o desenvolvimento colaborativo de pipelines de preparação de dados por empresas e TI usando fluxos de dados de mapeamento e processamento do ADF.
- Crie pipelines analíticos no ADF para analisar dados em lote e em tempo real.
- Crie e implante modelos de aprendizado de máquina para adicionar outros insights ao que você já sabe.
- Integre seu data warehouse com dados de streaming em tempo real.
- Simplifique o acesso a dados e insights em vários armazenamentos de dados analíticos do Azure criando um data warehouse lógico usando o PolyBase.

No próximo capítulo, você aprenderá em detalhes sobre os Serviços Cognitivos do Azure, com foco em arquitetar soluções que incluem inteligência como seu mecanismo principal.



# 19

## Arquitetar soluções inteligentes

A tecnologia de nuvem mudou muitas coisas, incluindo a criação de aplicações inteligentes de forma ágil, escalável e paga conforme o uso. As aplicações antes do surgimento da tecnologia de nuvem geralmente não incorporavam inteligência, principalmente porque:

- Era demorado e propenso a erros.
- Era difícil escrever, testar e experimentar algoritmos continuamente.
- Não havia dados suficientes.
- Era extremamente caro.

Na última década, duas coisas mudaram que levaram à criação de aplicações significativamente mais inteligentes do que no passado. Essas duas coisas são a escalabilidade ilimitada, sob demanda e econômica da nuvem, juntamente com a disponibilidade de dados em termos de volume, variedade e velocidade.

Neste capítulo, analisaremos arquiteturas que podem ajudar a criar aplicações inteligentes com o Azure. Alguns dos tópicos abordados neste capítulo são:

- A evolução da IA
- Processos de IA do Azure
- Serviços Cognitivos do Azure
- Criar um serviço de reconhecimento óptico de caracteres
- Criar um serviço de recursos visuais usando o SDK .NET de Pesquisa Cognitiva

## A evolução da IA

A IA não é um novo campo de conhecimento. Na verdade, a tecnologia é o resultado de décadas de inovação e pesquisa. No entanto, sua implementação nas décadas anteriores foi um desafio pelas seguintes razões:

1. **Custo:** experimentos de IA eram caros por natureza, e não havia tecnologia de nuvem. Toda a infraestrutura era comprada ou contratada por terceiros. Experimentos também eram demorados para configurar, e grandes habilidades eram necessárias para começar. Um grande poder de armazenamento e computação também era necessário, o que geralmente faltava na comunidade em geral e estava disponível apenas para alguns.
2. **Falta de dados:** quase não havia dispositivos portáteis inteligentes e sensores disponíveis gerando dados. Os dados eram limitados por natureza e precisavam ser adquiridos, o que novamente tornava as aplicações de IA dispendiosas. Os dados também eram menos confiáveis, e havia uma falta de confiança geral nos dados em si.
3. **Dificuldade:** os algoritmos de IA não eram documentados o suficiente e estavam principalmente no domínio de matemáticos e estatísticos. Eles eram difíceis de criar e utilizar nas aplicações. Basta imaginar a criação de um **sistema de reconhecimento óptico de caracteres (OCR)** há 15 anos. Não havia quase nenhuma biblioteca, dados, poder de processamento ou as habilidades necessárias para desenvolver aplicações usando OCR.

Embora o fluxo de dados tenha aumentado com o tempo, ainda havia falta de ferramentas para entender os dados de forma que agregasse valor comercial. Além disso, bons modelos de IA são baseados em dados suficientemente precisos e treinados com algoritmos para serem capazes de resolver problemas reais. A tecnologia de nuvem e o grande número de sensores e dispositivos portáteis redefiniram esse cenário.

Com a tecnologia de nuvem, é possível provisionar recursos de armazenamento e computação sob demanda para aplicações baseadas em IA. A infraestrutura de nuvem fornece muitos recursos para migração de dados, armazenamento, processamento e computação, além de gerar insights e, eventualmente, fornecer relatórios e painéis. Ela faz tudo isso com um custo mínimo de forma mais rápida, pois não há nada físico envolvido. Vamos nos aprofundar na compreensão do que acontece por trás da criação de uma aplicação baseada em IA.

## Processos de IA do Azure

Cada projeto baseado em IA deve passar por um determinado conjunto de etapas para que seja operacional. Vamos explorar estas sete fases:

### Ingestão de dados

Nesta fase, os dados são capturados de várias fontes e armazenados de forma que possam ser consumidos na próxima fase. Os dados são limpos antes de serem armazenados, e os desvios da norma são ignorados. Isso faz parte da preparação de dados. Os dados podem ter diferentes velocidade, variedade e volume. Eles podem ser estruturados de forma semelhante a bancos de dados relacionais, semiestruturados como documentos JSON ou não estruturados, como imagens, documentos do Word e assim por diante.

### Transformação de dados

Os dados ingeridos são transformados em outro formato, pois podem não ser consumíveis em seu formato atual. A transformação de dados normalmente inclui a limpeza e a filtragem de dados, removendo os vieses dos dados, aumentando os dados ao uni-los a outros conjuntos de dados, criando dados adicionais com base em dados existentes e muito mais. Isso também faz parte da preparação dos dados.

### Análise

Os dados da última fase são reutilizados para análise. A fase de análise contém atividades relacionadas à descoberta de padrões nos dados, realização de análises de dados exploratórios e geração de insights adicionais com base neles. Esses insights são armazenados juntamente com os dados existentes para consumo na próxima fase. Isso faz parte do processo de empacotamento do modelo.

## Modelagem de dados

Depois que os dados são aumentados e limpos, os dados apropriados e necessários são disponibilizados para os algoritmos de IA a fim de gerar um modelo propício para atingir o objetivo geral. É um processo iterativo conhecido como experimentação usando várias combinações de dados (engenharia de recursos) para garantir que o modelo de dados seja robusto. Isso também faz parte do processo de empacotamento do modelo.

Os dados são alimentados em algoritmos de aprendizado para identificar padrões. Este processo é normalmente conhecido como treinamento do modelo. Mais tarde, os dados de teste são usados no modelo para verificar sua eficácia e eficiência.

## Validar o modelo

Depois que o modelo é criado, um conjunto de dados de teste é usado para encontrar sua eficácia. Se a análise obtida dos dados de teste reflete a realidade, o modelo é sólido e utilizável. O teste é um aspecto importante do processo de IA.

## Implantação

O modelo é implantado em produção para que os dados em tempo real possam ser alimentados nele para obter a saída prevista. Essa saída pode ser usada nas aplicações.

## Monitoramento

O modelo implantado em produção é monitorado continuamente para a análise futura de todos os dados de entrada e para treinar novamente e melhorar os modelos de eficácia.

Os estágios e processos de IA, por natureza, são demorados e iterativos. Assim, as aplicações baseadas neles têm um risco inerente de serem de longa duração, experimentais e de uso intensivo de recursos, juntamente com atrasos com custos excedentes e com poucas chances de sucesso.

Mantendo essas coisas em mente, deve haver soluções de IA prontas para uso que os desenvolvedores possam usar nas aplicações para torná-las inteligentes. Essas soluções de IA devem ser facilmente consumíveis de aplicações e devem ter os seguintes recursos:

- **Capacidade entre plataformas:** os desenvolvedores que usam qualquer plataforma devem ser capazes de consumir esses serviços. Elas devem ser implantadas e consumidas em Linux, Windows ou Mac sem problemas de compatibilidade.
- **Capacidade entre linguagens:** os desenvolvedores devem ser capazes de usar qualquer linguagem para consumir essas soluções. Os desenvolvedores não só encontrarão uma curva de aprendizado mais curta, como também não precisarão alterar sua escolha preferencial de linguagem para consumir essas soluções.

Essas soluções devem ser implantadas como serviços usando padrões e protocolos da indústria. Geralmente, esses serviços estão disponíveis como pontos de extremidade REST HTTP que podem ser invocados usando qualquer linguagem de programação e plataforma.

Há muitos tipos de serviço que podem ser modelados e implantados para o consumo do desenvolvedor. Alguns exemplos incluem:

- **Conversão de linguagem:** nesses serviços, o usuário fornece texto em uma linguagem e obtém o texto correspondente em uma linguagem diferente como saída.
- **Reconhecimento de caracteres:** esses serviços aceitam imagens e retornam o texto presente neles.
- **Conversão de fala em texto:** esses serviços podem converter a fala de entrada em texto.

Agora que examinamos os detalhes da criação de um projeto baseado em IA/ML, vamos nos aprofundar nas aplicações de vários serviços cognitivos oferecidos pelo Azure.

## Serviços Cognitivos do Azure

O Azure fornece um serviço abrangente conhecido como Serviços Cognitivos do Azure. Os Serviços Cognitivos do Azure é um conjunto de serviços que os desenvolvedores podem consumir em suas aplicações para transformá-las em aplicações inteligentes.

Visão	Pesquisa na Web	Linguagem	Fala	Decisão
<ul style="list-style-type: none"> <li>• Visão Computacional</li> <li>• Detecção Facial</li> <li>• Video Indexer</li> <li>• Visão Personalizada</li> <li>• Reconhecedor de formulário (visualização)</li> <li>• Reconhecedor de tinta (visualização)</li> </ul>	<ul style="list-style-type: none"> <li>• Sugestão Automática do Bing</li> <li>• Pesquisa Personalizada do Bing</li> <li>• Pesquisa de Entidade do Bing</li> <li>• Pesquisa de Imagens do Bing</li> <li>• Pesquisa do Bing News</li> <li>• Verificação Ortográfica do Bing</li> <li>• Pesquisa de Vídeo do Bing</li> <li>• Pesquisa na Web do Bing</li> </ul>	<ul style="list-style-type: none"> <li>• Leitor imersivo (visualização)</li> <li>• Reconhecimento de linguagem</li> <li>• QnA Maker</li> <li>• Análise de Texto</li> <li>• Tradutor</li> </ul>	<ul style="list-style-type: none"> <li>• Fala em Texto</li> <li>• Texto em Fala</li> <li>• Tradução de Fala</li> <li>• Reconhecimento de palestrantes (visualização)</li> </ul>	<ul style="list-style-type: none"> <li>• Detecção de Anomalias (visualização)</li> <li>• Moderador de Conteúdo</li> <li>• Personalizador</li> </ul>

Tabela 19.1: serviços Cognitivos do Azure

Os serviços foram divididos em cinco categorias principais, dependendo de sua natureza. Estas cinco categorias são as seguintes:

## Visão

Esta API fornece algoritmos para classificação de imagens e ajuda no processamento de imagens, fornecendo informações significativas. A visão computacional pode fornecer uma variedade de informações com base nas imagens em diferentes objetos, pessoas, personagens, emoções e muito mais.

## Pesquisa

Essas APIs ajudam em aplicações relacionadas a pesquisas. Elas ajudam na pesquisa com base em texto, imagens, vídeo e fornecem opções de pesquisa personalizadas.

## Linguagem

Estas APIs são baseadas no processamento de linguagem natural e ajudam a extrair informações sobre a intenção do texto enviado pelo usuário juntamente com a detecção de entidades. Elas também ajudam na análise e conversão de texto para diferentes linguagens.

## Fala

Estas APIs ajudam a converter a fala em texto, texto em fala e tradução de fala. Elas podem ser usadas para ingerir arquivos de áudio e realizar ações com base no conteúdo em nome dos usuários. A Cortana é um exemplo que usa serviços semelhantes para tomar medidas para os usuários com base em fala.

## Decisão

Estas APIs ajudam na detecção de anomalias e na moderação de conteúdo. Elas podem verificar se há conteúdo em imagens, vídeos e texto e descobrir padrões que devem ser destacados. Um exemplo dessa aplicação é exibir um aviso sobre conteúdo adulto.

Agora que você tem uma compreensão dos conceitos básicos dos Serviços Cognitivos, vamos abordar como eles funcionam em detalhes.

## Entender os Serviços Cognitivos

Os Serviços Cognitivos do Azure consistem em pontos de extremidade HTTP que aceitam solicitações e enviam respostas de volta para o chamador. Quase todas as solicitações são solicitações POST HTTP e consistem em um cabeçalho e um corpo.

O provisionamento de Serviços Cognitivos gera dois artefatos importantes que ajudam um chamador a invocar um ponto de extremidade com êxito. Ele gera uma URL de ponto de extremidade e uma chave única.

O formato da URL é `https://[azure location].api.cognitive.microsoft.com/{cognitive type}/{version}/{sub type of service}?{query parameters}`. Um exemplo de URL é:

`https://eastus.api.cognitive.microsoft.com/vision/v2.0/  
ocr?language=en&detectOrientation=true`

O Serviço Cognitivo é provisionado na região Leste dos EUA do Azure. O tipo de serviço é a visão computacional usando a versão 2 e o subtipo é OCR. Geralmente, há alguns subtipos para cada categoria de nível superior. Por fim, há alguns parâmetros de cadeia de consulta, como `language` e `detectOrientation`. Esses parâmetros de consulta são diferentes para cada categoria e subcategoria de serviço.

O cabeçalho ou os parâmetros de consulta devem fornecer o valor de chave para que a invocação de ponto de extremidade seja bem-sucedida.

O valor de chave deve ser atribuído à chave de cabeçalho `Ocp-Apim-Subscription-Key` com a solicitação.

O conteúdo do corpo da solicitação pode ser uma cadeia simples, um binário ou uma combinação de ambos. Dependendo do valor, o cabeçalho do tipo de conteúdo apropriado deve ser definido na solicitação.

Os valores de cabeçalho possíveis são:

- `Application/octet-stream`
- `multipart/form-data`
- `application/json`

Use `octet-stream` ao enviar dados binários e `json` para enviar valores de cadeia. `form-data` pode ser usado para enviar vários valores de combinação de binário e texto.

A chave é uma cadeia única usada para validar se o chamador recebeu permissão para invocar a URL. Essa chave deve ser protegida para que outras pessoas que não devem ser capazes de invocar os pontos de extremidade não tenham acesso a ela. Mais adiante no capítulo, você verá maneiras de proteger essas chaves.

## Consumir Serviços Cognitivos

Há duas maneiras de consumir Serviços Cognitivos:

- **Usando um ponto de extremidade HTTP diretamente:** neste caso, o ponto de extremidade é invocado diretamente ao criar o cabeçalho e o corpo com valores apropriados. Em seguida, o valor de retorno é analisado, e os dados são extraídos dele. Todos os serviços de IA nos Serviços Cognitivos são APIs REST. Elas aceitam solicitações HTTP em JSON, bem como outros formatos e respostas no formato JSON.
- **Usar um SDK:** o Azure fornece vários **kits de desenvolvimento de software (SDKs)**. Existem SDKs disponíveis para as linguagens .NET, Python, Node.js, Java e Go.

Na seção a seguir, veremos a utilização de um dos Serviços Cognitivos usando ambos os aspectos. Vamos explorar isso criando alguns serviços de IA usando pontos de extremidade HTTP.

## Criar um serviço de OCR

Nesta seção, usaremos alguns dos serviços de IA usando C#, bem como o PowerShell para mostrar seu uso usando o ponto de extremidade HTTP diretamente. A próxima seção se concentrará em fazer o mesmo usando um SDK .NET.

Antes de começar a criar um projeto usando Serviços Cognitivos, a primeira etapa é provisionar a API em si.

O reconhecimento óptico de caracteres está disponível como uma API de visão e pode ser provisionado usando o portal do Azure, conforme mostrado a seguir. Crie uma API de visão navegando pelos **Serviços Cognitivos > Visão computacional > Criar**, conforme mostrado na Figura 19.1:

Home > New > Computer Vision > Create

**Create**

Computer Vision

Name \*

Subscription \*

Location \*

Pricing tier [\(View full pricing details\)](#) \*

Resource group \*

[Create new](#)

Figura 19.1: criando uma API de visão

Depois que a API é provisionada, a página de visão geral fornece todos os detalhes para consumir a API. Ela fornece a URL base e as informações importantes. Anote a chave, pois será usada posteriormente:

- 1** Get the API Key and endpoint to authenticate your applications and start sending calls to the service.
 

Key1 ⓘ  
ff0cd61f27d8452bbadad36942570c48

Endpoint ⓘ  
https://azureforarchitects.cognitiveservices.azure.com/

All Computer Vision calls and Docker container activations require a key. Specify the key either in the request header (Web API), the Computer Vision client (SDK) or through the command-line (Docker container).
- 2** Try the service in the API console - requires an API Key and selecting your location: eastus
 

Use the API Console to quickly try the API without writing code. Be sure to select the correct location for this resource, as listed above.

[API Console](#)
- 3** Make a web API call - requires your API Key and endpoint
 

Use the sample code in these quickstarts to begin integrating Computer Vision into your applications to analyze images. You can unlock insights such as detecting objects, brands, faces and more.

[C# Quickstart](#)  
[Python Quickstart](#)  
[Java Quickstart](#)
- 4** Try the Computer Vision Containers
 

The Recognize Text portion of the Computer Vision Cognitive Service is also available as a Docker container. This enables you to run the API on-premises if you don't want your data to leave your machine or environment. These containers have the same interfaces and capabilities as the operation in the hosted API.

[Installing the Computer Vision Containers](#)  
[Container Support in Azure Cognitive Services](#)  
[Container Samples](#)

**Figura 19.2:** página de visão geral

Ela também fornece um console de API para testar rapidamente a API. Ao clicar nela, é aberta uma nova janela que tem todos os pontos de extremidade relacionados a este serviço disponíveis. Ao clicar em **OCR**, você verá um formulário que pode ser preenchido com os dados apropriados e executar os pontos de extremidade de serviço. Você também verá uma resposta completa. Isso é mostrado na Figura 19.3. A URL está disponível como uma URL de solicitação, e a solicitação é uma solicitação HTTP típica com um método **POST**. A URL aponta para o ponto de extremidade na região Leste dos EUA do Azure. Ela também está relacionada ao grupo de visão de APIs, versão 2 e ao ponto de extremidade de OCR.

A chave de assinatura é passada no cabeçalho com o nome **ocp-apim-subscription-key**. O cabeçalho também contém a chave do tipo de conteúdo com **application/json** como valor. Isso ocorre porque o corpo da solicitação contém uma cadeia JSON. O corpo está na forma de JSON com a URL da imagem da qual o texto deve ser extraído:

#### Request URL

```
https://eastus.api.cognitive.microsoft.com/vision/v2.0/ocr?language=en&detectOrientation=true
```

#### HTTP request

```
POST https://eastus.api.cognitive.microsoft.com/vision/v2.0/ocr?language=en&detectOrientation=true HTTP/1.1
Host: eastus.api.cognitive.microsoft.com
Content-Type: application/json
Ocp-Apim-Subscription-Key: *****
{"url": "https://ichef.bbci.co.uk/news/320/cpsprodpb/F944/production/_109321836_oomzonlz.jpg"}
```

**Send**

Figura 19.3: URL de solicitação

A solicitação pode ser enviada para o ponto de extremidade, clicando no botão **Enviar**. Isso resultará em uma resposta HTTP 200 OK, conforme mostrado a seguir, se tudo der certo. Se houver um erro nos valores da solicitação, a resposta será um código HTTP de erro:

#### Response content

```
csp-billing-usage: CognitiveServices.ComputerVision.OCR=1
apim-request-id: 6b08bdc8-edac-41e0-9361-f9679373b7e1
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
x-content-type-options: nosniff
Date: Sat, 04 Apr 2020 14:02:35 GMT
Content-Length: 257
Content-Type: application/json; charset=utf-8

{
  "language": "en",
  "textAngle": -0.029670597283902981,
  "orientation": "Up",
  "regions": [
    {
      "boundingBox": "159,136,80,18",
      "lines": [
        {
          "boundingBox": "159,136,80,18",
          "words": [
            {
              "boundingBox": "159,137,44,17",
              "text": "MY02"
            },
            {
              "boundingBox": "210,136,29,15",
              "text": "ZRO"
            }
          ]
        }
      ]
    }
}
```

Figura 19.4: resposta HTTP 200 OK

A resposta consiste em detalhes relacionados ao uso de cobrança, uma ID de solicitação interna gerada pelo ponto de extremidade, o tamanho do conteúdo, o tipo de conteúdo de resposta (sendo JSON) e os dados e o tempo da resposta. O conteúdo da resposta consiste em uma carga JSON com as coordenadas do texto e o texto em si.

## Usar o Powershell

A mesma solicitação pode ser criada usando o PowerShell. O seguinte código do PowerShell pode ser executado usando o PowerShell ISE.

O código usa o cmdlet **Invoke-WebRequest** para invocar o ponto de extremidade de Serviços Cognitivos, passando a URL para o parâmetro **Uri** usando o método **POST** e adiciona os cabeçalhos apropriados conforme abordado na última seção e, por fim, o corpo consistindo em dados no formato JSON. Os dados são convertidos em JSON usando o cmdlet **ConvertTo-Json**:

```
$ret = Invoke-WebRequest -Uri "https://eastus.api.cognitive.microsoft.com/vision/v2.0/ocr?language=en&detectOrientation=true" -Method Post  
-Headers @{"Ocp-Apim-Subscription-Key"="ff0cd61f27d8452bbadad36942570c48"; "Content-type"="application/json"} -Body $(ConvertTo-Json -InputObject @>{"url"="https://ichef.bbci.co.uk/news/320/cpsprodpb/F944/production/_109321836_oomzonlz.jpg"})  
  
$val = Convertfrom-Json $ret.content  
  
foreach ($region in $val.regions) {  
    foreach($line in $region.lines) {  
        foreach($word in $line.words) {  
            $word.text  
        }  
    }  
}
```

A resposta do cmdlet é salva em uma variável que também consiste em dados no formato JSON. Os dados são convertidos em um objeto do PowerShell usando o cmdlet **Convertfrom-Json** e colocados em loop para encontrar as palavras no texto.

## Usar C#

Nesta seção, vamos criar um serviço que deve aceitar solicitações de usuários, extrair a URL da imagem, criar a solicitação HTTP e enviá-la para o ponto de extremidade de Serviços Cognitivos. O ponto de extremidade de Serviços Cognitivos retorna uma resposta JSON. O conteúdo de texto apropriado é extraído da resposta e retornado ao usuário.

### Arquitetura e design

Uma aplicação inteligente é uma aplicação MVC ASP.NET Core. Uma aplicação MVC é criada por um desenvolvedor em uma máquina de desenvolvedor, gera uma imagem do Docker e envia a imagem do Docker para o Registro de Contêiner do Azure. Aqui, os principais componentes da aplicação são explicados, juntamente com seu uso:

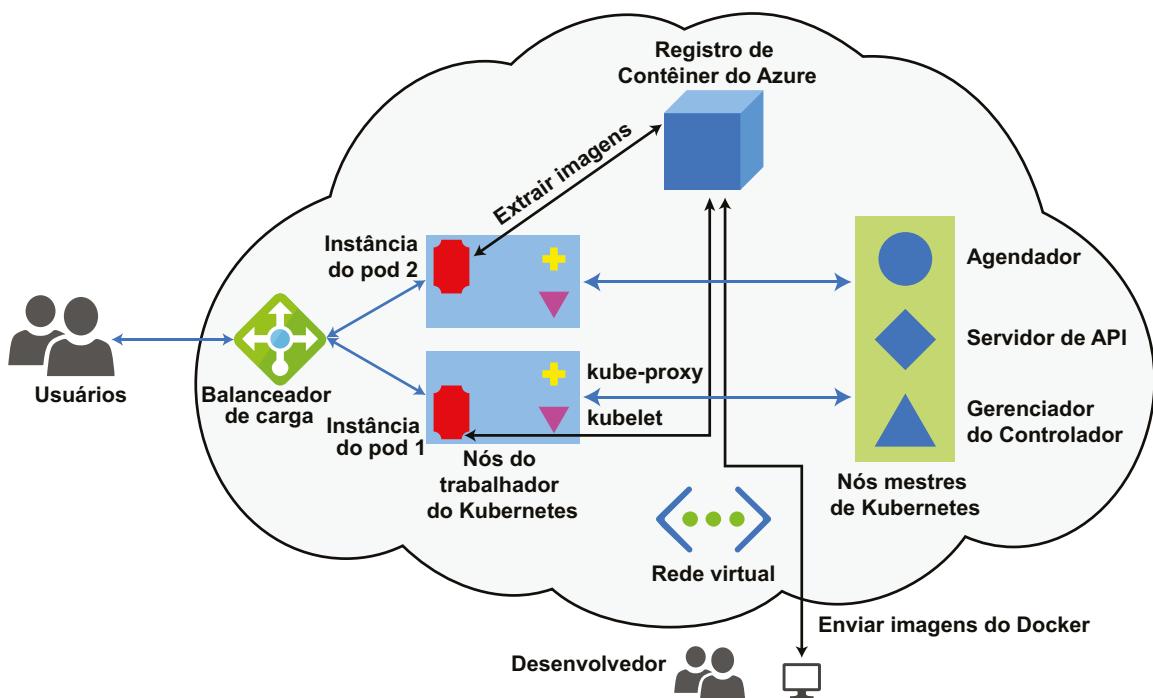


Figura 19.5: fluxo de trabalho de uma aplicação inteligente

## Docker

O Docker é um dos principais representantes em tecnologias de contêiner e está disponível entre plataformas, incluindo Linux, Windows e Mac. O desenvolvimento de aplicações e serviços em contêineres em mente fornece a flexibilidade para implantá-las em nuvens e locais, bem como na infraestrutura local. Ele também remove todas as dependências na plataforma de host, o que novamente permite menos dependência da plataforma como serviço. O Docker ajuda na criação de imagens personalizadas, e os contêineres podem ser criados com base nessas imagens. As imagens contêm todas as dependências, binários e estruturas necessárias para fazer a aplicação ou o serviço funcionar, e eles são completamente autossuficientes. Isso os torna um ótimo destino de implantação para serviços como microsserviços.

## Registro de Contêiner do Azure

O Registro de Contêiner do Azure é um registro semelhante ao Docker Hub para o armazenamento de imagens de contêiner em um repositório. É possível criar vários repositórios e carregar várias imagens neles. Uma imagem tem um nome e um número de versão, formando juntos um nome totalmente qualificado que é usado para consultá-los em uma definição do pod do Kubernetes. Essas imagens podem ser acessadas e baixadas por qualquer ecossistema do Kubernetes. Um pré-requisito disso é que os segredos apropriados para extrair a imagem já devem ser criados antes. Ela não precisa estar na mesma rede que os nós do Kubernetes e, na verdade, não há necessidade de uma rede criar e usar o Registro de Contêiner do Azure.

## Serviço Azure Kubernetes

A aplicação inteligente que aceita a URL de uma imagem para recuperar o texto nela pode ser hospedada em máquinas virtuais simples ou até mesmo no Serviço de Aplicativo do Azure. No entanto, a implantação no Serviço de Kubernetes do Azure oferece muitas vantagens, o que foi abordado no Capítulo 8, Arquitetar soluções do Kubernetes do Azure. Por enquanto, é importante saber que essas aplicações têm recuperação automática por natureza, e um número mínimo de instâncias é automaticamente mantido pelo mestre do Kubernetes, juntamente com o fornecimento da flexibilidade para atualizá-los de várias maneiras, incluindo implementações azul-verde e atualizações canário.

## Pods, conjuntos de réplicas e implantações

O desenvolvedor também cria um arquivo YAML relacionado à implantação do Kubernetes que faz referência às imagens na especificação do pod e também fornece uma especificação para o conjunto de réplicas. Ele fornece sua própria especificação relacionada à estratégia de atualização.

## Design de tempo de execução

A arquitetura e o design permanecem os mesmos que na seção anterior. No entanto, quando a aplicação ou o serviço já está disponível e em operação, ele já fez o download das imagens do Registro de Contêiner do Azure e criou pods que executam contêineres nelas. Quando um usuário fornece uma URL de imagem para decodificar o texto que contém, a aplicação no pod invoca a API de visão computacional dos Serviços Cognitivos do Azure, passa a URL para ela e aguarda uma resposta do serviço:

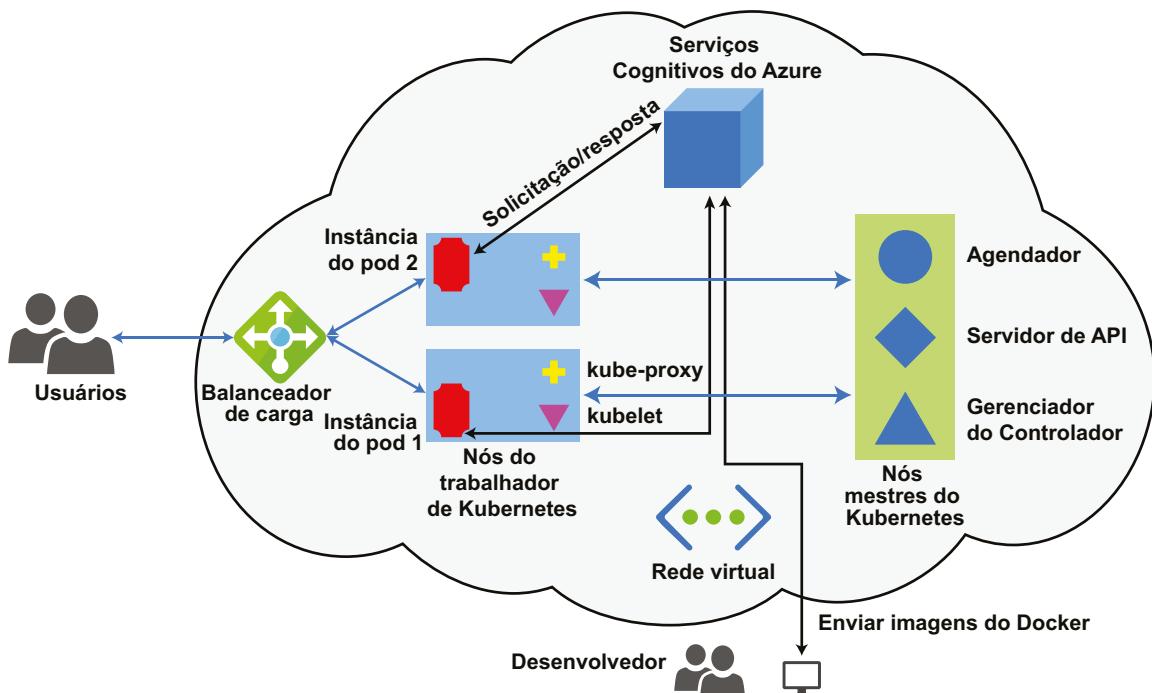


Figura 19.6: fluxo de trabalho de uma aplicação inteligente

Depois que recebe a resposta JSON dos serviços, ele pode recuperar as informações e retorná-las ao usuário.

## O processo de desenvolvimento

O ambiente de desenvolvimento pode ser Windows ou Linux. Ele funcionará com o Windows 10 e o Windows 2016/19 Server. Ao usar o Windows, é útil implantar o Docker para Windows para que ele crie um ambiente do Linux e um ambiente Docker do Windows.

Ao criar um projeto de aplicação Web ASP.NET Core usando o Visual Studio 2019, a opção **Supporte do Docker** deve ser selecionada com **Windows** ou **Linux** como valores. Dependendo do valor escolhido, o conteúdo apropriado será gerado no **Dockerfile**. A principal diferença no **Dockerfile** é os nomes de imagem base. Ele usa imagens diferentes para Linux em comparação com o Windows.

Ao instalar o Docker para Windows, ele também instala uma máquina virtual Linux, e, por isso, é importante ativar o hipervisor Hyper-V.

Neste exemplo, em vez de enviar os dados como uma cadeia JSON, a imagem é obtida por download, e os dados binários são enviados para o ponto de extremidade de Serviços Cognitivos.

Ele tem uma função que aceita uma entrada de cadeia para valores de URL. Em seguida, ele invoca os Serviços Cognitivos com valores de cabeçalho apropriados e um corpo contendo a URL. Os valores de cabeçalho devem conter a chave fornecida pelos Serviços Cognitivos durante o provisionamento do serviço. O valor no corpo pode conter valores de cadeia simples na forma de JSON ou pode conter dados de imagem binários em si. A propriedade de cabeçalho do tipo de conteúdo deve ser definida em conformidade.

O código declara a URL e a chave relacionadas aos Serviços Cognitivos. Isso é mostrado apenas para fins informativos. A URL e a chave devem ser colocadas em arquivos de configuração.

Usando o objeto **HttpClient** a imagem correspondente à URL fornecida pelo usuário é obtida por download e armazenada na variável **responseMessage**. Outro objeto **HttpClient** é instanciado, e seus cabeçalhos são preenchidos com **Ocp-Apim-Subscription-Key** e **content-type keys**. O valor do cabeçalho do tipo de conteúdo é **application/octet-stream**, pois os dados binários estão sendo passados para o ponto de extremidade.

Uma solicitação POST é feito após a extração do conteúdo da variável **responseMessage** e transmiti-la como corpo da solicitação para o ponto de extremidade do Serviço Cognitivo.

O código da ação do controlador é mostrado a seguir:

```
[HttpPost]
public async Task<string> Post([FromBody] string value)
{
    string myurl = " https://eastus.api.cognitive.microsoft.com/
vision/v2.0/ocr?language=en&detectOrientation=true
    string token = ".....";
    using (HttpClient httpClient = new HttpClient())
    {
        var responseMessage = await httpClient.GetAsync(value);
        using (var httpClient1 = new HttpClient())

```

```
    }

        httpClient1.BaseAddress = new Uri(myurl);
        httpClient1.DefaultRequestHeaders.Add("Ocp-Apim-
Subscription-Key", token);

        HttpResponseMessage content = responseMessage.Content;
        content.Headers.ContentType = new
MediaTypeWithQualityHeaderValue("application/octet-stream");
        var response = await httpClient1.PostAsync(myurl, content);

        var responseContent = await response.Content.
ReadAsByteArrayAsync();
        string ret = Encoding.ASCII.GetString(responseContent, 0,
responseContent.Length);
        dynamic image = JsonConvert.DeserializeObject<object>(ret);
        string temp = "";
        foreach (var regs in image.regions)
        {
            foreach (var lns in regs.lines)
            {
                foreach (var wds in lns.words)
                {
                    temp += wds.text + " ";
                }
            }
        }
        return temp;
    }
}
```

Depois que o ponto de extremidade terminar o processamento, ele retornará a resposta com uma carga JSON. O contexto é extraído e desserializado em objetos .NET. Vários loops são codificados para extrair o texto da resposta.

Nesta seção, criamos uma aplicação simples que usa os Serviços Cognitivos para fornecer extrações de palavras de recursos usando a API de OCR e implantá-la nos pods do Kubernetes. Esse processo e a arquitetura podem ser usados em qualquer aplicação que deseja consumir APIs dos Serviços Cognitivos. Em seguida, vamos ver outra API de Serviços Cognitivos, conhecida como recursos visuais.

## Criar um serviço de recursos visuais usando o SDK .NET de Pesquisa Cognitiva

A última seção foi sobre a criação de um serviço que usa um ponto de extremidade cognitivo de OCR para retornar o texto nas imagens. Nesta seção, um novo serviço será criado que retornará recursos visuais em uma imagem, como descrições, marcas e objetos.

### Usar o Powershell

O código no PowerShell é semelhante ao exemplo de OCR anterior, portanto, não é repetido aqui. A URL é diferente do exemplo de código anterior:

Request URL

```
https://eastus.api.cognitive.microsoft.com/vision/v2.0/analyze?visualFeatures=Description&language=en
```

HTTP request

```
POST https://eastus.api.cognitive.microsoft.com/vision/v2.0/analyze?visualFeatures=Description&language=en HTTP/1.1
1
Host: eastus.api.cognitive.microsoft.com
Content-Type: application/json
Ocp-Apim-Subscription-Key: *****
{"url": "https://thefinanser.com/wp-content/uploads/2016/06/People.jpg"}
```

Send

Figura 19.7: URL de solicitação

A solicitação é feita usando um método **POST**, e a URL aponta para o ponto de extremidade na região Leste dos EUA do Azure. Ela também usa a versão 2 e consome a API de visão.

A chave de acesso aos Serviços Cognitivos faz parte do cabeçalho HTTP chamado **ocp-apim-subscription-key**. O cabeçalho também contém o cabeçalho **content-type** com **application/json** como valor. Isso ocorre porque o corpo da solicitação contém um valor JSON. O corpo tem a URL da imagem da qual o texto deve ser extraído.

A resposta será no formato JSON com o conteúdo da imagem e uma descrição.

## Usar .NET

Este exemplo é novamente uma aplicação MVC ASP.NET Core e tem o pacote NuGet **Microsoft.Azure.CognitiveServices.Vision.ComputerVision** instalado nela:

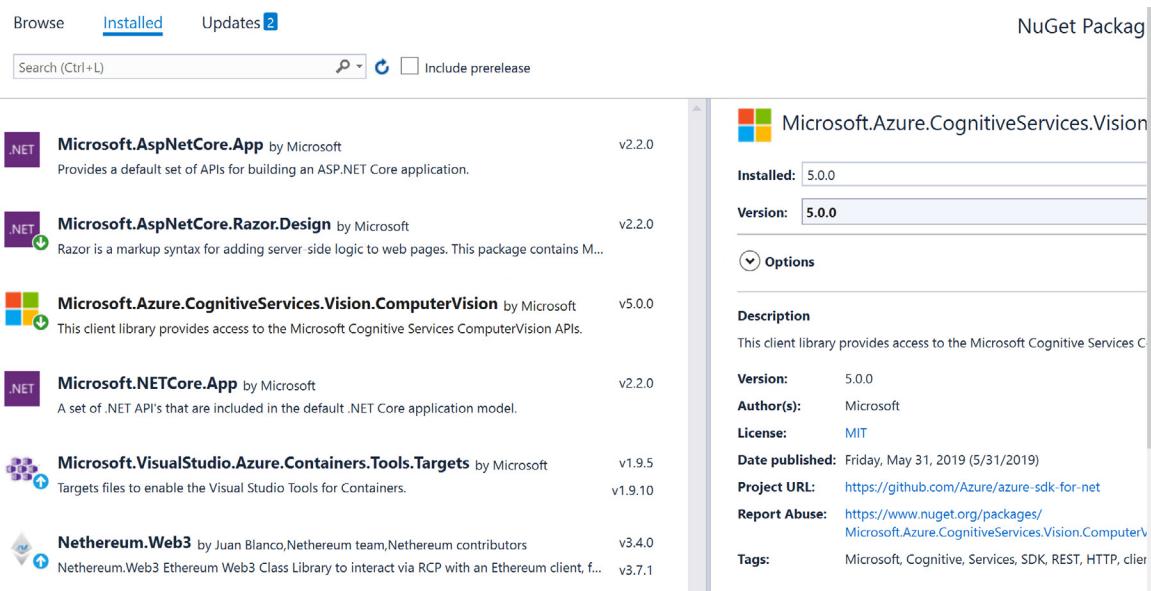


Figura 19.8: aplicação MVC ASP.NET Core com o pacote NuGet Microsoft.Azure.CognitiveServices.Vision.ComputerVision

O código da ação do controlador é mostrado a seguir. Neste código, o serviço cognitivo e a chave são declarados. Ele também declara variáveis para os objetos **ComputerVisionClient** e **VisionType**. Ele cria uma instância do tipo **ComputerVisionClient**, fornecendo a URL e a chave.

A lista **VisionTypes** consiste em vários tipos de dados procurados na imagem: as marcas, as descrições e os objetos são adicionados. Somente esses parâmetros serão extraídos da imagem.

Um objeto **HttpClient** é instanciado para fazer o download da imagem usando a URL disponibilizada pelo usuário e envia esses dados binários para o ponto de extremidade de serviços cognitivos usando a função **AnalyzeImageInStreamAsync** do tipo **ComputerVisionClient**:

```
[HttpPost]
```

```
    public string Post([FromBody] string value)
    {
        private string visionapiurl = " https://
eastus.api.cognitive.microsoft.com/vision/v2.0/
analyze?visualFeature=tags,description,objects&language=en";
        private string apikey = "e55d36ac228f4d718d365f1fcddc0851";
        private ComputerVisionClient client;
        private List<VisualFeatureTypes> visionType = new
List<VisualFeatureTypes>();

client = new ComputerVisionClient(new ApiKeyServiceClientCredentials(apikey))
{
    Endpoint = visionapiurl
};

visionType.Add(VisualFeatureTypes.Description);
visionType.Add(VisualFeatureTypes.Tags);
visionType.Add(VisualFeatureTypes.Objects);

string tags = "";
string descrip = "";
string objprop = "";
using (HttpClient hc = new HttpClient())
{
    var responseMessage = hc.GetAsync(value).GetAwaiter().GetResult();
    Stream streamData = responseMessage.Content.
ReadAsStreamAsync().GetAwaiter().GetResult();
    var result = client.AnalyzeImageInStreamAsync(streamData,
visionType).GetAwaiter().GetResult();
    foreach (var tag in result.Tags) {
        tags += tag.Name + " ";
    }
    foreach (var caption in result.Description.Captions)
```

```
{  
    descrip += caption.Text + " ";  
}  
  
foreach (var obj in result.Objects)  
{  
    objprop += obj.ObjectProperty + " ";  
}  
  
}  
return tags;  
// return descrip or objprop  
  
}
```

Os resultados são colocados em loop, e as marcas são retornadas ao usuário. Da mesma forma, descrições e propriedades de objetos também podem ser retornadas ao usuário. Agora, vamos conferir as maneiras como podemos proteger a exposição de chaves de serviço.

## Proteger a chave dos Serviços Cognitivos

Há várias maneiras de proteger a exposição de chaves a outros atores. Isso pode ser feito usando o recurso de gerenciamento de API no Azure. Também pode ser feito usando Proxies do Azure Functions.

### Usar Proxies do Azure Functions

Os Proxies do Azure Functions podem se referir a qualquer URL, seja ela interna ou externa. Quando uma solicitação chega a Proxies do Azure Functions, ela usará a URL do serviço cognitivo, juntamente com a chave para invocar o ponto de extremidade cognitivo, e também substituirá os parâmetros de solicitação e adicionará a URL de imagem recebida e a adicionará à URL do ponto de extremidade cognitivo como dados POST. Quando uma resposta volta do serviço, ele substituirá a resposta, removerá os cabeçalhos e passará os dados JSON de volta para o usuário.

## Consumir Serviços Cognitivos

O consumo dos Serviços Cognitivos segue um padrão consistente. Cada serviço cognitivo está disponível como uma API REST, com cada API esperando conjuntos diferentes de parâmetros para trabalhar. Os clientes que invocam essas URLs devem conferir a documentação para parâmetros associados e fornecer valores para eles. Consumir URLs é um método relativamente bruto de usar Serviços Cognitivos. O Azure fornece SDKs para cada serviço e para várias linguagens. Os clientes podem usar esses SDKs para trabalhar com Serviços Cognitivos.

The API de criação **LUIS (Serviço Inteligente de Reconhecimento Vocal)** está disponível em [https://\[luis resource name\]-authoring.cognitiveservices.azure.com/](https://[luis resource name]-authoring.cognitiveservices.azure.com/), e a API de produção está disponível em

```
https://[azure region].api.cognitive.microsoft.com/luis/prediction/v3.0/apps/
{application id}/slots/production/predict?subscription-key={cognitive key}
&verbose=true&show-all-intents=true&log=true&query=YOUR_QUERY_HERE.
```

Da mesma forma, a API de detecção facial está disponível em [https://\[endpoint\]/face/v1.0/detect\[?returnFaceId\]\[&returnFaceLandmarks\]\[&returnFaceAttributes\]\[&recognitionModel\]\[&returnRecognitionModel\]\[&detectionModel\]](https://[endpoint]/face/v1.0/detect[?returnFaceId][&returnFaceLandmarks][&returnFaceAttributes][&recognitionModel][&returnRecognitionModel][&detectionModel]).

Há muitas APIs de Serviços Cognitivos, cada uma com vários tipos em termos de URLs, e a melhor maneira de saber sobre essas URLs é usar a documentação do Azure.

## Resumo

Neste capítulo, você adquiriu uma compreensão da arquitetura de implantação e da arquitetura de aplicações para criar aplicações inteligentes no Azure. O Azure fornece Serviços Cognitivos com vários pontos de extremidade: cada ponto de extremidade é responsável por executar um algoritmo relacionado à IA e fornecer saídas. Quase todos os pontos de extremidade de Serviços Cognitivos funcionam de forma semelhante em relação a solicitações e respostas HTTP. Esses pontos de extremidade também podem ser invocados usando SDKs fornecidos pelo Azure para diferentes linguagens, e você viu um exemplo de como obter recursos visuais usando-os. Há mais de 50 pontos de extremidade diferentes, e é recomendável obter uma compreensão da natureza dos pontos de extremidade usando o recurso de console de API oferecido pelo Azure.



Microsoft.Source Newsletter - Inbox

Message

Microsoft

Microsoft.Source Newsletter | Issue 7

You're reading Microsoft.Source, the developer community newsletter featuring ideas and projects from your peers down the street –and around the world. If someone forwarded you this newsletter and you want to receive future editions, [sign up >](#)

Give feedback Get more of what you want in each edition.

**Featured Story**

**Vanilla JS and HTML –No frameworks, no libraries, no problem >**  
Do you know what it takes to render HTML elements without the complexity of AngularJS, React, Svelte, or Vue.js? See how to create a simple web page with pure HTML, CSS, and JS.  
Web, JavaScript, HTML

**What's New**

**Build a web experience to send GIFs to MXChip >**  
IoT, project

**The Making of Azure Mystery Mansion >**  
Game, Twine, PlayFab

**Trying to make FETCH happen >**  
Serverless, IoT, Azure Functions

**Events** [See all events](#)

**Cosmos DB Live Webcast / Online >**  
Expert-led, containers, .Net

**OpenHack Serverless / Los Angeles >**  
In-person event, serverless, hack

**Learning**

**Microsoft Ignite – Watch videos on demand >**  
Watch all keynotes, announcements, and sessions on demand

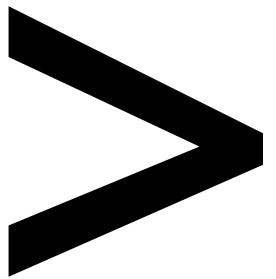
# Por desenvolvedores, para desenvolvedores

Boletim informativo  
Microsoft.Source

Receba artigos técnicos, código de exemplo e informações sobre os próximos eventos no Microsoft.Source, o boletim informativo mensal da comunidade de desenvolvedores.

- Acompanhe as tecnologias mais recentes
- Conecte-se com seus colegas em eventos da comunidade
- Aprenda com recursos práticos





# Índice

## Sobre

Todas as principais palavras-chave usadas neste livro são capturadas em ordem alfabética nesta seção. Cada uma é acompanhada pelo número da página onde aparece.

# A

acesso: 3, 8, 10, 12-13, 19, 71, 73-74, 76, 80, 84-85, 87, 110, 113, 115, 145-148, 156, 158-159, 163-165, 168, 170-171, 177-178, 184, 189-190, 204-206, 208, 210, 213, 221, 226-227, 230-232, 234, 237-243, 245, 247-252, 256, 265, 267, 275, 287, 294, 297-298, 319, 322, 358-359, 361-363, 371, 395-396, 399, 402, 442, 446, 458, 479, 486, 503-504, 517, 522, 545, 553, 565-566, 569, 576, 604, 609, 627, 632, 637, 645, 655 conta: 38, 52, 60, 64, 87, 105-107, 109-111, 113, 116-117, 119, 121, 132, 134, 136-137, 139, 141, 151, 157, 172, 174, 176-177, 179, 184, 191, 194, 220-221, 233, 248-252, 257, 274, 279-284, 286-289, 293, 296-297, 309-312, 318-320, 329, 332, 335-339, 347-348, 358, 361-364, 366, 369, 371-372, 375, 378, 399-400, 405, 408, 410-413, 415, 417, 443, 458, 467-468, 483, 504, 511, 516-519, 522, 524, 526-529, 532, 535, 538, 554, 559, 565-569 adappname: 361 addcontent: 374

endereço: 29, 33, 35, 37, 73, 75, 77, 93, 226, 233, 236, 246, 250, 256, 268, 446, 458-459, 471, 480, 485-486, 501-503, 526, 528-529, 560-561, 570, 572 agregados: 405, 407 alertas: 27, 58, 61, 63-66, 129, 133, 171, 198, 229, 237, 247-248, 254, 266, 333, 578 algoritmo: 50, 659 análise: 8, 50, 61-64, 66-67, 70, 74, 193, 244, 247-248, 265-266, 269, 273, 276-277, 333, 347, 351, 389-392, 403-408, 411, 413-416, 418, 421, 442-443, 462, 472, 534, 578, 580, 582-588, 598, 601-607, 609-614, 616, 619-623, 625-627, 631-633, 635-637, 644 android: 222, 317 ansible: 428, 430, 442 apache: 7, 275-276, 298, 392, 587, 606-607, 637 baseado em API: 116 apikey: 657 apiversion: 151, 157, 485, 488-490, 496, 498-499, 511-512, 514, 520, 524, 527-528, 530, 533-539, 542-544, 551-554, 568, 571-572 appname: 485, 488-491, 496 arquiteto: 20, 23-24, 67, 71, 101, 162, 179, 184, 190-191, 269, 343, 355, 386, 603, 632 artefatos: 106-107, 112, 162-163, 230, 285-286, 395, 429, 432, 434-436, 441, 444, 459-460, 644 assembleias: 265, 432, 448, 460 ataque: 228, 237, 248, 265 auditorias: 187 automatizar: 17-18, 20, 101, 105, 109, 184, 428, 475, 613, 619, 622, 633 autoscale: 48, 53, 315 azcopy: 634 azureappid: 363, 367 azureblob: 536 azurecr: 485, 490-491 azurerm: 119, 122-123 azure-sql: 199, 213, 217

# B

back-end: 163, 310, 315, 321, 505 backup: 46, 201, 606, 632 bacpac: 429 balanceador: 29, 32-35, 38-39, 45, 50, 189, 245, 458-459, 487-488, 498, 500, 526 largura de banda: 42, 70, 73, 76, 79, 87, 215, 634 binário: 272, 406, 439, 645, 653, 656 associação: 307-311, 326, 330-332 blobadded: 335, 340 blobsource: 539 navegador: 17, 263, 295, 470, 500, 576 pacote: 518, 520

# C

armazenamento em cache: 58, 190, 612  
retorno de chamada: 261  
canário: 651  
cassandra: 7, 219-221, 298  
certkey: 115  
cgroups: 476  
cliente: 16-17, 36-37, 57-58, 85, 238, 241, 245, 248, 254, 257, 259-260, 276, 321, 380, 463, 481, 580, 657  
nuvens: 7, 67, 72, 136, 139, 141, 145, 173, 633, 651  
cluster: 29, 200, 275-276, 295, 478, 480-481, 483-484, 486-487, 490, 492-495, 500-506, 604-605  
cmdlet: 18, 112, 116-118, 121, 127, 137-139, 150, 159, 340, 342, 358, 361, 368, 469-470, 516, 519, 649  
cognitivo: 8, 317, 325, 347, 637, 640, 643-646, 649-650, 652-653, 655-659  
colunar: 199, 298, 610  
compilar: 108, 430-431, 470  
simultâneo: 195, 606, 617  
configurar: 12, 19, 27, 40, 62, 66, 130-131, 134, 136-137, 162, 197, 218, 222, 233, 259-261, 266, 292, 307, 313, 333, 346, 348, 350-352, 361, 371, 412, 414-415, 429-430, 433, 446, 457, 460, 467-468, 470, 473, 495, 513, 530, 567, 589, 594

restrição: 79, 198, 502, 625  
contêiner: 4, 10, 15-16, 32, 50, 60, 147, 178, 190, 211, 220, 280, 282, 288-289, 297, 316, 318-321, 330, 340, 395, 413-414, 417, 438, 443, 462-463, 473, 475, 477-479, 484-485, 493, 496, 500-501, 505-507, 522, 558, 565-566, 569, 650-652  
cosmosdb: 330

# D

dacpac: 448  
daemon: 16-17, 463  
painel: 61, 107, 172-173, 176, 180, 247, 373, 443  
banco de dados: 6-7, 16, 39-40, 106, 111, 163, 178, 190, 193-199, 201-202, 204, 206, 208-215, 217-220, 222, 231, 242-243, 252-253, 255-256, 269, 330-331, 405, 432, 442, 445-446, 448, 534-536, 557, 569, 577, 585-586, 610, 612, 618, 634  
databricks: 272, 274, 276, 293-295, 297, 302-303, 510, 534, 586, 607  
datacenter: 7, 26-27, 29, 31, 36, 38, 61, 70, 76, 80-82, 86, 197, 230, 251, 253, 392, 545, 556-557, 565, 596, 613  
dataops: 619

conjunto de dados: 179, 282, 287-289, 292, 298, 406, 536-538, 541, 581  
descriptografia: 33, 227, 239-240  
implantação: 4-8, 10-12, 14, 17, 19-20, 24-25, 39, 42, 45, 49, 58-59, 72-75, 78, 152, 163-164, 179-180, 184, 188, 190, 194-200, 211, 213, 218-219, 222, 225-226, 228, 230-231, 233, 235-236, 239, 244-245, 247, 256, 268, 306, 308, 314-315, 343, 358, 425-427, 429, 431-437, 440, 443, 445-446, 460, 463, 466, 472-473, 475, 478-479, 483, 485, 488-490, 492, 498-500, 506, 509, 512-513, 516-521, 525-526, 534, 545, 547-548, 550-551, 554, 556, 567, 569-570, 572-573, 607, 642, 651, 659  
devops: 6, 11, 226-227, 286, 303, 418, 421-427, 430, 435-444, 457-458, 461-462, 464-466, 471-473, 479, 493  
dnsserver: 544  
downstream: 14, 274, 410, 582, 614

# E

ecossistema: 1, 6-8, 44, 303, 390, 424, 443, 477, 484, 585, 603, 612, 637, 651  
emulador: 327

criptografia: 33, 70, 75, 79, 85, 208-209, 227, 231, 239, 251, 253-254, 267-268, 358, 632  
Pontos de extremidade: 4, 19, 29, 35-38, 40, 125, 256-257, 265, 268, 316, 321, 325, 334, 486-487, 588-589, 591-593, 612, 643-647, 659  
Inscrição: 168  
Entidades: 84, 87, 220, 227, 586  
Eventgrid: 336, 342, 365, 381  
Eventtype: 335, 341, 368  
Extensão: 36, 50, 240, 307, 325, 336, 377-378, 429, 463, 558, 564, 610

## F

Failover: 40, 78, 86, 89, 221  
Nome do arquivo: 115, 137, 297, 497, 537  
Sistema de arquivos: 208, 237, 360-361  
Firewall: 29, 33, 38, 40, 74, 189, 204-205, 233-237, 245-247, 252-253, 265, 269, 446, 458, 483, 493, 557, 570, 596  
Formatos: 256, 272-273, 277, 298, 317, 405-406, 583, 646  
Estrutura: 9, 18, 46, 222, 275, 307-308, 423, 429, 468, 501  
Frequência: 84, 184, 348, 412, 537-538, 540, 581

## G

github: 54, 293, 303, 535, 545, 577  
Gremílio: 219-221

## H

Hadoop: 7, 269, 272, 274-275, 303, 389, 586-587, 622  
Cabeçalho: 125-127, 252, 297-298, 341-342, 368-369, 406, 495, 592, 644-646, 648, 653, 655  
Horizontal: 51, 212  
Hospedagem: 1, 3, 14, 30, 33, 38, 46, 74, 131, 133, 189, 197-198, 226, 229, 238, 285, 315, 343, 407, 414, 445, 476, 493, 503, 506, 526, 532, 584

Nome de host: 485  
HttpClient: 653-654, 656-657  
HttpPost: 653, 657  
HttpStart: 325-326

## I

Identificador: 117, 128, 241, 289, 335, 375, 569  
Imagens: 14, 17, 58, 84, 272, 316, 443, 463, 479, 641, 643-644, 651-653, 655  
Importar: 112, 122-123, 137, 300, 362, 364-365, 469, 636  
Ingestão: 273, 277, 391-392, 399, 403-404, 578, 581, 584-585, 587, 606-607, 612

633, 635, 641

Instalador: 18

Instância: 12, 26, 30, 32-33, 36, 46-47, 49, 54, 56-57, 64, 70, 148, 152, 182, 185, 194, 198-199, 210, 213-214, 222, 245, 272, 277, 279-280, 282, 289, 346, 357-358, 390-391, 393, 487, 492-493, 501-503, 519-520, 522-524, 550, 581, 632, 656

Integridade: 76, 227, 625

Faturamento: 168, 176-177

Isolamento: 4, 10, 15-16, 73, 76, 83, 90, 195, 220, 247, 432, 497, 549, 596, 604-605

## J

JavaScript: 19, 58, 265, 307, 405, 407, 430

Jenkins: 6, 418, 422, 464-466

## K

Kernel: 15-16, 462, 476

KeyVault: 241, 365, 521, 525, 569

Kubectl: 495-497, 499-500, 504

Kubelet: 483, 493, 504-507

## L

Lambda: 307

Bibliotecas: 17, 58, 265, 432, 463, 640

Licença: 188

localhost: 137, 139-140, 471  
locais: 14, 18, 58, 81,  
86, 153, 164, 221,  
519, 556, 651  
registro: 59-60, 74, 115,  
141, 238, 263, 265,  
358, 477, 479, 532  
pesquisa: 405

## M

máquina: 7, 25, 30, 57,  
65, 70-72, 74, 84, 99,  
127, 130, 145, 161, 180,  
185-186, 188-190,  
197-198, 213, 231-233,  
235, 238, 256, 268, 274,  
276, 279, 295, 315, 356,  
360, 389-390, 405,  
407, 429-430, 442-443,  
460, 462, 467, 471, 473,  
478-479, 526, 528-530,  
532, 541-545, 551, 556,  
558-560, 562-565,  
570, 572, 586-587,  
603-604, 606-607,  
622, 637, 650, 653

mainframe: 1

mariadb: 196

mestre: 169, 213, 275-276,  
294, 343, 448, 478-484,  
492, 500, 535, 545,  
555, 557, 565-567,  
571-572, 651

memória: 14, 44, 46-48,  
51, 186, 199, 204,  
217-218, 361, 408, 633

metadados: 12-13, 51, 152,  
167, 309, 485, 488-491,  
496, 498-499, 564,  
591-592, 609-610,  
613, 619, 632

migração: 11, 279-280,  
602, 611, 613-617,  
619-621, 623, 625,  
632-637, 641  
modelagem: 228-229, 642  
monitoramento: 11, 23,  
32, 42, 52, 58-59,  
61-63, 66-67, 74, 187,  
198, 210-211, 222, 229,  
231, 244, 265, 267-269,  
279, 313, 316, 319,  
389, 435, 443, 462,  
471-472, 477, 483, 577,  
588, 607, 610, 642

## N

namespace: 15, 279,  
283, 393-396, 400,  
403, 409-410, 497,  
504, 512, 551

nativo: 178, 213, 257,  
479, 501, 609-610

nginx-lb: 499-500

bloco de anotação:  
295-296

## O

descarregar: 33, 343, 637  
sob demanda: 2-3, 6, 14,  
240, 306, 318, 603-604,  
606-608, 639, 641

openid: 227, 257

oracle: 7

os-level: 30

## P

pacote: 18, 243, 255, 324,  
429, 435, 446, 564, 656

partição: 275, 398,

400-402  
senha: 112-116, 210,  
240-241, 359-361,  
364, 380, 386, 495,  
502, 536, 544  
padrão: 69, 86, 90-101,  
218, 265, 333, 415,  
466, 555, 584,  
592-593, 603, 659  
carga: 35, 66, 184, 227,  
308-309, 335, 341, 368,  
382, 495, 649, 655  
petabytes: 8, 85, 274,  
582, 588, 606  
pipeline: 278, 280,  
282-284, 286-287, 291,  
293, 431, 433-434,  
443, 446, 448, 452,  
457, 459-461, 463-464,  
472, 538, 541, 603,  
609, 612, 633, 650  
guia estratégico: 247  
política: 8, 13, 56,  
145-148, 152-153,  
155-156, 162-164, 174,  
278, 361, 395-396,  
399, 522, 540, 565  
postgresql: 6, 196  
postman: 127, 328  
powershell: 8, 10, 17-20,  
50, 72, 108-109, 111-112,  
114-115, 117-118, 120-121,  
125, 127, 134, 148-150,  
152, 157-159, 163, 237,  
266, 316, 340, 356-357,  
362, 364, 366, 386,  
429, 440, 443, 458,  
466-468, 470, 472,  
493, 510, 512-513, 515,  
519, 521, 543-544, 564,  
567, 646, 649, 655  
premium: 30, 47-48,

84, 87, 190, 200,  
212, 294, 315, 568  
preços: 141, 168, 178,  
184-185, 191, 214-215,  
217-220, 222, 247, 315,  
347, 370, 393-394, 409,  
594-595, 598, 608  
entidade: 59, 107, 111, 113,  
115-118, 121, 161, 210,  
241, 252, 287, 358-362,  
367, 380-381, 516, 569  
protegido: 66, 75,  
230-231, 236, 239, 247,  
251, 256-257, 265, 360,  
522, 553, 593, 645  
protocolo: 29, 33,  
35, 84-85, 88, 227,  
232-233, 251, 257, 341,  
370, 392, 485, 488,  
490-491, 497, 579-580,  
584, 588-589, 593  
psgallery: 564  
publicar: 333-334,  
341-342, 357,  
368-369, 385, 586  
pyspark: 300  
python: 6, 109, 112, 307,  
316, 356, 466, 586,  
603-604, 606-607, 646

## R

raspberry: 589  
rawdata: 282  
legível: 19, 202, 405, 430  
reinicializar: 27, 30, 564  
recriar: 503  
recuperação: 4, 7, 197-198,  
201, 221-222, 513, 545  
redundância: 25-27,  
58, 86, 189, 200

regiões: 7, 12-14, 27-29,  
36-41, 67, 69-73,  
75, 79-80, 101, 106,  
163-164, 189, 191, 197,  
202-203, 219, 221, 230,  
246, 333, 512, 515, 519,  
545, 556, 649, 654  
registro: 15, 237, 316,  
479, 485, 650-652  
confiável: 82, 84, 88-89,  
101, 103, 275-276, 426,  
578, 580, 597, 640  
remoto: 81, 98,  
238, 246, 459  
réplicas: 86, 221, 489,  
491, 498-499  
replicaset: 482-483,  
488, 490-493  
relatar: 176, 180, 182, 184,  
436, 442, 577, 606  
resiliente: 29, 38, 98,  
274, 298, 436  
resposta: 33, 43, 99,  
127-128, 131, 182, 184,  
221-222, 233, 244, 247,  
315, 321, 493, 647-650,  
652, 654-656, 658  
recuperar: 19, 150, 159,  
182, 241, 277, 330, 332,  
341, 368, 651-652  
revogar: 250  
rgname: 569  
baseado em função:  
8, 115, 145, 242, 294,  
504, 609, 633  
runbook: 105-106, 109,  
111, 118-121, 123-129,  
131-132, 134-136, 141,  
333, 356-357, 366-367,  
369, 379, 382, 385, 466

## S

escalabilidade: 3-4, 44-49,  
51, 53-54, 56-57, 197,  
315, 475, 477, 576, 594  
cenários: 29, 53-54, 76,  
78-79, 96, 107, 109,  
136, 180, 184, 222, 277,  
301, 323, 392, 457,  
584-585, 603, 609, 614  
agenda: 53-54, 98, 237,  
276, 307, 346, 369,  
385, 425, 483, 493,  
505, 586, 622  
escopos: 170, 172, 237  
script: 107, 111, 115,  
118-121, 134, 136-138,  
152, 237, 251, 356,  
512, 516-518, 521, 525,  
543, 558, 564-565  
segredo: 239-241, 261-262,  
357, 359-360, 368-369,  
519-521, 523-524, 569  
sendgrid: 355, 357,  
370-373, 375,  
377-379, 386  
sentinel: 243-245, 247  
servicebus: 325, 395, 400  
servicenow: 66  
sessão: 15-16, 29,  
32-33, 35, 38, 40,  
263, 302, 392, 459  
definir: 49, 57, 85, 107,  
131, 136, 161, 211, 217,  
266, 285, 315-316, 366,  
371-373, 375, 378-379,  
522, 566, 592, 605, 607  
sharepoint: 233  
assinatura: 85, 113, 249,  
287, 395, 399, 522, 553  
skuname: 533  
subordinados: 275

socket: 239, 392  
sqlazure: 534  
sqlclient: 636  
sqldataset: 538-540  
sql-query: 586  
autônomo: 236, 242, 244  
sem estado: 57, 142, 392  
estático: 58, 65, 75, 190,  
312, 320, 373, 576  
streaming: 266, 391-392,  
403, 407-408, 414,  
418, 587, 637  
estruturado: 84,  
272-273, 641  
sub-rede: 37, 50, 73,  
75, 207, 232-233,  
235, 238-239,  
501-503, 506-507,  
527-529, 562, 567  
assinar: 333-334, 338, 379  
assinado: 333  
mudar: 27, 172, 262,  
478, 495, 615  
synapse: 405, 586-587,  
598, 601-614,  
616, 619-637  
sysadmin: 567

## T

destino: 29, 33, 87, 113,  
242-243, 277-278, 289,  
292-293, 310, 315, 319,  
390, 407, 426, 443,  
468, 613, 619, 651  
modelo: 19-20, 56, 157,  
336, 348, 358, 446,  
489, 491, 498, 510-519,  
521-522, 524-527, 529,  
532, 534-535, 538,  
541, 547-561, 563,  
565-568, 570-573, 614

locatário: 77, 96, 117,  
257, 358, 361, 363,  
367, 380, 569  
terraform: 428  
tokens: 85, 87, 182, 245,  
249-250, 252, 569, 593  
rastreamento: 63, 76,  
178, 247, 255, 437, 577  
tráfego: 29, 33-38, 40,  
45-47, 72, 74, 77, 79,  
82, 106, 142, 188-189,  
234, 237-238, 245,  
248, 390, 501, 589  
twilio: 355, 357, 371-373,  
375, 378-379, 386  
twitter: 257, 265, 269,  
307, 406, 411-413, 415

## U

ubuntu: 235  
upgrade: 54-56, 489, 594

## V

validação: 277, 431,  
433-434, 437,  
460, 471, 481  
cofres: 241, 268, 358, 366,  
368, 379, 525, 569  
versão: 11-12, 19, 51, 56,  
274, 279, 284-285, 293,  
295, 308, 324, 430,  
438-439, 441, 444, 478,  
488-490, 510-513, 516,  
531, 592, 604, 607, 610,  
645, 647, 651, 655  
virtualizar: 15, 462

## W

warehouse: 277,  
586-587, 601-606,  
609-612, 614-619,  
622-623, 625-627,  
632-634, 636-637  
baseado na web: 33, 606  
webdeploy: 429, 432  
webhook: 66, 119, 125-127,  
129, 131, 333, 342  
webserver: 468, 470, 496  
webserver-: 496  
windows: 1, 6, 14-18, 50,  
63, 108, 188, 210, 234,  
236, 243, 297-298, 316,  
395, 400, 429-430,  
442, 462, 466, 476-477,  
503, 536, 552-553,  
558, 564-565, 596,  
642, 651-653  
workbooks: 247  
trabalhador: 105-106,  
111, 119, 127, 134-136,  
276, 294-295, 478-481,  
483, 492, 500  
fluxos de trabalho:  
108-109, 278, 321-322,  
346, 350, 355, 466,  
586, 609, 622  
workloads: 26, 58, 211,  
226, 233, 235, 237, 317,  
587, 604-605, 613

