AWS 服务与 Azure 服务对照表

以下对照表列出了常见 AWS 服务及其在 Azure 上的对应服务，并提供 Azure 官方迁移指南链接（如有）。

AWS 服务	对应的 Azure 服务	Azure 官方迁移手册链接
K8s Cluster nodes (VM)	Azure Kubernetes Service (AKS)	将 Amazon EKS 迁移到 Azure AKS
Aurora for MySQL	Azure Database for MySQL	使用 DMS 脱机迁移 MySQL 至 Azure Database for MySQL
Aurora for PostgreSQL	Azure Database for PostgreSQL	Amazon Aurora PostgreSQL 脱机迁移至 Azure Database for PostgreSQL
RDS for MySQL	Azure Database for MySQL	使用数据传入复制迁移 RDS MySQL 至 Azure Database for MySQL
RDS for PostgreSQL	Azure Database for PostgreSQL	使用迁移服务离线迁移 RDS PostgreSQL 至 Azure Database for PostgreSQL
Redis (ElastiCache)	Azure Cache for Redis	迁移到 Azure Cache for Redis
SQS	Azure 队列存储 (Queue Storage)	暂无
Memcached (ElastiCache)	Azure Marketplace Memcached *(替代方案)*	暂无
Kafka	Azure Event Hubs (支持 Kafka 协议)	迁移到 Azure 事件中心 (Kafka)生态系统
OpenSearch (Amazon OpenSearch Service)	Azure 认知搜索 (Cognitive Search)	暂无
S3 storage	Azure Blob 存储	使用 AzCopy 将数据从 Amazon S3 复制到 Azure 存储
WAF (Web Application Firewall)	Azure Web Application Firewall (WAF)	暂无
Key Management (AWS KMS)	Azure Key Vault	暂无
EMR (Hadoop)	Azure HDInsight (Hadoop)	暂无

1. K8s Cluster Migration

迁移前需要确认的云内容

1) 服务对应关系

- **计算服务**
  - **AWS EKS**：Amazon Elastic Kubernetes Service 是 AWS 提供的托管 Kubernetes 服务，支持自动化的集群创建、配置和管理。
  - **Azure AKS**：Azure Kubernetes Service 是 Azure 提供的托管 Kubernetes 服务，支持高可用性和自动修复功能。
- **存储服务**

    - **AWS EBS**：Elastic Block Store 提供块存储，通常用于持久化存储。
    - **Azure Managed Disks**：Azure 的托管磁盘服务，提供高性能的块存储。
    - **AWS EFS**：Elastic File System 提供文件存储，支持多个 EC2 实例共享访问。
    - **Azure Files**：Azure 的文件存储服务，支持 SMB 和 NFS 协议，允许多个虚拟机共享访问。
- **网络服务**
    - **AWS VPC**：Virtual Private Cloud 提供隔离的网络环境，支持子网划分、路由表和安全组。
    - **Azure 虚拟网络**：Azure 的虚拟网络服务，支持子网划分、路由表和网络安全组（NSG）。
- **安全服务**
    - **AWS IAM**：Identity and Access Management 提供身份验证和授权管理。
    - **Azure RBAC**：Role-Based Access Control 提供基于角色的访问控制，支持细粒度的权限管理。
- **监控和日志**
    - **AWS CloudWatch**：提供监控和日志记录服务，支持指标和日志的收集、存储和分析。
    - **Azure Monitor**：提供监控和日志记录服务，支持指标和日志的收集、存储和分析。

2) 环境评估和准备

- **计算资源**
    - 评估 EKS 集群的节点类型、数量和资源使用情况。
    - 确认 AKS 支持的节点类型和配置，确保满足应用需求。
- **存储配置**
    - 评估 EKS 集群中使用的存储类型（EBS、EFS）及其配置。
    - 确认 Azure 中的对应存储服务（Managed Disks、Azure Files）及其配置。
- **网络配置**
    - 评估 EKS 集群的网络拓扑，包括子网划分、路由表和安全组规则。
    - 确认 Azure 虚拟网络的网络拓扑，包括子网划分、路由表和网络安全组（NSG）规则。
- **安全配置**
    - 评估 EKS 集群的 IAM 角色和策略设置。
    - 确认 Azure 中的 RBAC 角色和权限设置。
- **监控和日志**
    - 评估 EKS 集群的监控指标和日志设置。
    - 确认 Azure Monitor 的监控指标和日志设置。
- **Kubernetes 版本**
    - 确保目标 Kubernetes 版本在 AKS 支持的窗口内。

迁移工具清单

1) 容器镜像迁移

- **Docker**：用于拉取和推送容器镜像。
- **Azure 容器注册表**：用于存储和管理容器镜像。
- **AWS ECR**：用于导出容器镜像。

2) 数据迁移

- **Velero**：用于备份和恢复 Kubernetes 资源和持久卷。
- **AzCopy**：用于在 AWS S3 和 Azure Blob 存储之间迁移数据。
- **AWS DataSync**：用于将数据从 AWS 存储服务迁移到 Azure。

3) 配置迁移

- **Kubectl**：用于管理 Kubernetes 集群和应用。
- **Helm**：用于管理 Kubernetes 应用的包和配置。
- **Kustomize**：用于自定义 Kubernetes 配置。

4) 自动化工具

- **Terraform**：用于基础设施即代码（IaC）的自动化部署。

- **Ansible**：用于配置管理和自动化部署。
- **Azure CLI**：用于在 Azure 上执行命令行操作。

迁移整体流程概览

1) 环境评估和准备

- 评估当前 AWS EKS 集群的配置和资源使用情况。
- 在 Azure 上创建新的 AKS 集群，并配置网络、存储和安全设置。

2) 数据和配置备份

- 使用 Velero 备份 EKS 集群的 Kubernetes 资源和持久卷。
- 使用 AzCopy 将数据从 AWS S3 迁移到 Azure Blob 存储。

3) 容器镜像迁移

- 从 AWS ECR 导出容器镜像。
- 将容器镜像推送到 Azure 容器注册表。

4) 应用迁移

- 更新 Kubernetes 清单文件，将 AWS 特定配置替换为 Azure 特定配置。
- 使用 Kubectl 或 Helm 将应用部署到 AKS 集群。

5) 验证和优化

- 验证应用在 AKS 上的运行情况。
- 优化性能和安全设置。

迁移步骤

1) 迁移前准备

- **评估和规划**：评估当前 EKS 集群的配置，规划在 Azure 上的部署架构。
- **创建 AKS 集群**：在 Azure 上创建新的 AKS 集群，并配置网络、存储和安全设置。
- **设置工具**：安装和配置 Velero、AzCopy、Kubectl 等工具。

2) 数据和配置备份

- **备份 EKS 集群**：使用 Velero 备份 EKS 集群的 Kubernetes 资源和持久卷。
- **迁移数据**
  - EBS
    - **备份 EBS 卷**：使用 AWS CLI 或 AWS 管理控制台对 EBS 卷进行快照备份。
    - **导出快照**：将 EBS 快照导出为 VHD 或 VMDK 格式。
    - **上传到 Azure**：使用 AzCopy 或 Azure CLI 将快照文件上传到 Azure Blob 存储。
    - **创建 Managed Disk**：在 Azure 中使用上传的快照文件创建新的 Managed Disk。
    - **挂载到 AKS**：将 Managed Disk 挂载到 AKS 集群的节点或 Pod 中。
  - EFS
    - **备份 EFS 文件系统**：使用 AWS CLI 或 AWS 管理控制台对 EFS 文件系统进行备份。
    - **导出数据**：将 EFS 文件系统中的数据导出到 S3 存储桶。
    - **迁移数据**：使用 AzCopy 将数据从 S3 存储桶迁移到 Azure Blob 存储。
    - **创建 Azure Files 共享**：在 Azure 中创建新的 Azure Files 共享。
    - **挂载到 AKS**：将 Azure Files 共享挂载到 AKS 集群的节点或 Pod 中。

3) 容器镜像迁移

- **导出镜像**：从 AWS ECR 导出容器镜像。
- **推送到 Azure 容器注册表**：将容器镜像推送到 Azure 容器注册表。

4) 常用组件迁移

- **网络插件**
  - **Calico**：在 EKS 中使用的 Calico 网络插件也可以在 AKS 中使用。需要检查 Calico 的配置是否与 Azure 网络兼容。
  - **Flannel**：Flannel 是另一种常用的网络插件，也可以在 AKS 中使用。需要确保 Flannel 的配置与 Azure 网络兼容。
- **存储插件**
  - **EBS CSI 驱动程序**：在 EKS 中使用的 EBS CSI 驱动程序需要替换为 Azure 的 CSI 驱动程序，如 Azure Disk CSI 或 Azure File CSI。
  - **EFS CSI 驱动程序**：在 EKS 中使用的 EFS CSI 驱动程序需要替换为 Azure Files CSI 驱动程序。
- **监控和日志**
  - **Prometheus 和 Grafana**：在 EKS 中使用的 Prometheus 和 Grafana 可以在 AKS 中继续使用。需要确保 Prometheus 和 Grafana 的配置与 Azure Monitor 兼容。
  - **EFK Stack**：Elasticsearch、Fluentd 和 Kibana（EFK Stack）在 EKS 中用于日志收集和分析。在 AKS 中可以继续使用 EFK Stack，也可以选择 Azure Monitor 的日志分析功能。
- **CI/CD 工具**
  - **Jenkins**：在 EKS 中使用的 Jenkins 可以在 AKS 中继续使用。需要确保 Jenkins 的配置与 AKS 环境兼容。
  - **Argo CD**：Argo CD 是一种 GitOps 工具，用于 Kubernetes 应用的持续交付。在 EKS 中使用的 Argo CD 可以在 AKS 中继续使用。
    5) 应用迁移
- **无状态负载**
  - **更新清单**：更新 Kubernetes 清单，将 YAML 文件中的镜像引用替换为容器注册表路径。
    - 检查现有的 Kubernetes 资源清单文件，查找 AWS 特定的配置，例如 VPC 和 IAM 角色。
    - 检查与节点、服务账户和其他资源相关的 EKS IAM 角色，并将其映射到对应的 Azure AKS 角色型访问控制 RBAC 角色。可以参考 Kubernetes 工作负载身份识别和访问。
    - 修改清单文件，将 AWS 特定设置替换为 Azure 特定设置，例如注释。
  - **应用清单到 AKS**：
    - 使用 `kubectl apply -f` 将修改后的 Kubernetes 清单文件应用到 AKS 集群。在此之前，需要先 连接到 AKS 集群。
- **有状态负载**
  - 在 AKS 和 EKS 集群中设置 Velero，参考 Velero for AKS 和 Velero for EKS。
  - 执行 EKS 集群的备份，参考 使用 Velero 备份 EKS 集群。
  - 使用 AzCopy 命令，将 Velero 备份从 S3 存储桶复制到 Azure Blob 存储。
  - 如果 AKS 和 EKS 对 PVC 使用不同的 `storageClassNames`，请创建 configMap，将源 `storageClassNames` 转译为 AKS 兼容的类别名称。如果在 EKS 和 AKS Kubernetes 集群上使用相同的存储解决方案，则可以跳过此步骤。
  - 使用 Velero restore 命令 将备份还原至 AKS。
  - 对还原的对象进行必要的更改，例如更新容器镜像引用或访问机密信息。

6) 验证和优化

- **功能验证**：验证应用在 AKS 上的运行情况，确保功能正常。
- **性能优化**：优化资源分配、网络配置等，提升应用性能。
- **安全加固**：配置网络策略、RBAC 权限等，确保应用安全。

验证方式

1) 功能验证

- **冒烟测试**：验证应用的基本功能是否正常。
- **集成测试**：验证应用各模块之间的集成是否正常。
- **用户验收测试（UAT）**：邀请用户进行验收测试，确保应用满足业务需求。

2) 性能验证

- **负载测试**：使用工具如 JMeter 或 Locust 进行负载测试，验证应用在高并发情况下的性能。
- **压力测试**：验证应用在极端负载下的稳定性和响应时间。

3) 安全验证

- **漏洞扫描**：使用工具如 Trivy 或 Clair 扫描容器镜像和应用中的漏洞。
- **渗透测试**：进行渗透测试，验证应用的安全性。

性能优化与安全

1) 性能优化

- **资源分配**：根据应用负载调整 AKS 集群的节点数量和类型。
- **网络优化**：配置 Azure 虚拟网络的加速网络和负载均衡器，提升网络性能。
- **存储优化**：选择合适的存储类别和性能等级，优化数据访问速度。

2) 安全措施

- **网络策略**：配置 Azure 虚拟网络的安全组和网络策略，限制不必要的网络访问。
- **RBAC 权限**：配置 AKS 的 RBAC 权限，确保最小化权限原则。
- **密钥管理**：使用 Azure Key Vault 管理密钥和证书，确保数据安全。
- **安全扫描**：定期进行漏洞扫描和安全评估，及时修复安全问题。

相关资源

- 适用于 Amazon EKS 专业人士的 AKS
- Kubernetes 身份识别与访问管理
- Kubernetes 监控和记录
- 对 Kubernetes 的安全网络访问
- Kubernetes 集群的存储选项
- Kubernetes 的成本管理
- Kubernetes 节点和节点池管理
- 集群治理

2. Aurora for MySQL

本指南详细描述了如何利用 Azure Database Migration Service (Azure DMS) 将 Amazon RDS for Aurora MySQL 数据库迁移至 Azure Database for MySQL。文档覆盖了从前期沟通、环境准备、测试迁移、正式迁移到后续验证等各个阶段，旨在确保迁移过程平稳顺利，同时保障数据完整性和业务连续性。

以下是两者在架构、性能、配置等方面的具体差异对比表：

对比维度	Amazon RDS for Aurora MySQL	Azure Database for MySQL

对比维度	Amazon RDS for Aurora MySQL	Azure Database for MySQL
架构设计	云原生架构，计算与存储分离， 存储层基于分布式设计并集成 S3	基于单节点架构，存储与计算耦合， 采用传统集中式存储设计
数据库引擎兼容性	仅兼容 MySQL 和 PostgreSQL 的特定版本 （与上游版本不完全一致）	完全兼容 MySQL 社区版，支持主流版本（如 MySQL 8.0）
性能优化	单位 CPU 的 Sysbench QPS 更高； 支持自动读写分离，读副本延迟低（毫秒级）	性能依赖实例规格，未提供类似 Aurora 的“计算能力单元”优化指标；读副本需手动配置
存储架构	存储自动扩展（最高 128 TB），IOPS 与存储容量解耦（独立计费）	存储容量与 IOPS 绑定（最高 16 TB）， 需预定义存储大小
扩展性	支持 15 个只读副本，副本自动扩展； 计算节点可独立扩缩容	最多支持 5 个只读副本，需手动调整； 计算与存储需同步扩缩容
高可用性	多可用区部署时故障切换时间 < 30 秒； 存储层自动多副本冗余（6 副本跨 AZ）	高可用模式依赖跨区域复制，故障切换时间约 1- 2 分钟；存储默认 3 副本（同区域）
备份与恢复	自动增量备份（保留 1-35 天）， 支持跨区域备份；恢复速度更快（分钟级）	手动或自动全量备份（保留 7-35 天）， 跨区域备份需额外配置；恢复时间依赖数据量
定价策略	按计算节点规格、存储用量和 I/O 请求单独计费；读副本单独收费	按统一实例规格（vCore + 存储）计费； 读副本按实例规格单独收费
最大连接数	动态调整（最高约 16,000， 取决于实例规格）	固定上限（如 4vCPU 实例约 3,000 连接）
运维复杂度	支持自动故障修复和内核优化； 大表删除操作对性能影响更低	依赖用户自定义维护窗口； 大表操作可能导致性能波动

1) 前提条件与客户确认信息

在正式启动迁移之前，请与客户确认以下关键事项：

- **停机窗口**
  - 确认业务可接受的停机时间或是否要求近零停机。
  - 协商最佳的迁移执行时间（例如低峰期）。
- **源数据库详情（Amazon RDS for Aurora MySQL）**
  - Aurora MySQL 版本及兼容性（确认与目标 Azure Database for MySQL 版本的兼容性）。
  - 数据库大小、表结构、分区、外键、视图、存储过程、触发器及使用的存储引擎。
  - 数据库负载、性能指标和未来增长预估。
- **目标数据库要求（Azure Database for MySQL）**
  - 目标数据库版本（推荐选择与源版本兼容的 MySQL 版本，如 5.7 或 8.0）。
  - 区域选择、计算/存储规格、备份策略和高可用性需求。
  - 是否需要额外的安全配置和性能优化需求。
- **网络与安全配置**
  - 确认 AWS 和 Azure 之间的网络互联方案（VPN、ExpressRoute 或 AWS Direct Connect）。
  - 检查防火墙、安全组和访问控制策略，确保 Azure DMS 可访问 Aurora 实例（默认 MySQL 端口 3306）。

- **访问权限与认证**
  - 在 Aurora 实例上为 Azure DMS 分配所需的权限（例如 SELECT、SHOW VIEW、TRIGGER 权限）。
  - 确认连接字符串、用户名和密码已正确配置，测试远程连接是否正常。
- **迁移模式选择**
  - 确定迁移方式：离线迁移（全量迁移）或在线迁移（初始全量数据迁移后，持续捕获实时变更）。
  - 制定详细的回退计划，确保在迁移失败时可快速切换回原系统。
- **相关团队协调**
  - 明确 DBA、开发、运维及业务团队各自的职责和协调响应机制。
  - 确保测试环境已搭建完成，以便进行干跑测试。

---

2) 架构概述

- **数据源**：Amazon RDS for Aurora MySQL
- **数据目标**：Azure Database for MySQL
- **迁移工具**：Azure Database Migration Service (Azure DMS)
- **迁移模式**：离线迁移或在线迁移（根据业务需求选择）
- **网络连通性**：确保 AWS 与 Azure 之间的网络互联正常
- **监控与验证**：利用 Azure Monitor 和 Azure DMS 控制台实时监控迁移进程和目标数据库的性能

---

3) 迁移前准备

3.1) 环境与数据评估

- **收集数据库信息**
  - 获取 Aurora MySQL 数据库的版本、存储配置、表结构、索引、视图、存储过程及触发器信息。
  - 确认数据库中是否使用特定的 MySQL 引擎或扩展（例如 InnoDB 设置与备份配置）。
- **性能基准与容量规划**
  - 采集现有数据库的性能指标，为目标实例的容量和性能做规划和调整。
  - 根据数据大小确定迁移过程中是否需要分批次处理或并行处理数据。
- **停机与回退策略**
  - 根据业务要求确定允许的停机时间窗口。
  - 制定详细的回退流程，以便在迁移异常时迅速恢复到 Aurora 实例。

3.2) 网络与安全配置

- **网络互联**
  - 配置并测试 AWS 与 Azure 之间的 VPN/ExpressRoute 连接，确保带宽和延迟满足迁移要求。
- **安全策略**
  - 配置 Aurora 实例和 Azure 环境的防火墙/网络安全组，确保 Azure DMS 可以通过 MySQL 默认端口（3306）通信。
  - 为 Azure DMS 分配专用 IP 范围，并在防火墙配置中予以放行。
- **用户权限**

- 在 Aurora 实例上创建用于迁移的专用用户，分配必要的只读权限（SELECT、SHOW VIEW、TRIGGER 等）。

3.3) Azure 资源配置

- **目标数据库配置**
  - 在 Azure 上创建 Azure Database for MySQL 实例（建议使用 Flexible Server 模式），并根据预期工作负载分配合适的计算和存储资源。
  - 配置自动备份、监控和高可用性选项，保障数据安全与业务连续性。
- **Azure DMS 实例**
  - 在 Azure 门户中创建 Azure Database Migration Service 实例，选择支持目标迁移模式（在线迁移可能需要 Premium SKU）。

---

4) 迁移操作步骤

4.1) 创建迁移项目

a. 登录 Azure 门户。
b. 导航至 **Azure Database Migration Service**。
c. 创建新的迁移项目：

- **源数据库类型**：MySQL（适用于 Aurora MySQL）。
- **目标数据库类型**：MySQL。
- 选择迁移模式：离线或在线迁移。

4.2) 配置源与目标端点

4.2.1) 配置源端点（Amazon RDS for Aurora MySQL）

- 输入 Aurora 实例的服务器地址、端口（默认 3306）、数据库名称以及具有迁移权限的用户名/密码。
- 进行端点连接测试，确保 Azure DMS 能成功连接到 Aurora 实例。

4.2.2) 配置目标端点（Azure Database for MySQL）

- 填写 Azure Database for MySQL 实例的服务器地址、端口、数据库名称、用户名和密码。
- 进行端点测试，确保目标数据库的连接配置正确无误。

4.3) 测试迁移（Dry Run）

- 在非生产环境中启动一次测试迁移，验证数据的导出、传输及导入过程是否顺利。
- 检查并记录迁移日志，注意错误提示和数据一致性情况，为正式迁移做充分准备。

4.4) 执行正式迁移

- 启动正式的迁移任务：
  - **离线迁移**：在预定的停机窗口内停止源系统业务，进行全量数据的离线迁移。
  - **在线迁移**：在全量数据迁移完成后，启动实时数据复制，确保捕获所有后续变更。
- 持续监控迁移任务状态和日志，及时应对可能出现的错误或异常情况。

4.5) 数据验证与业务切换

- **数据验证**
  - 检查源和目标数据库间的记录数、表结构、索引、存储过程以及触发器，确保数据完全一致。

    - 如有必要，可使用数据比对工具进行详细校验。

- **业务切换**

    - 与业务团队协商切换时间，将应用连接切换至目标 Azure Database for MySQL 实例。
    - 切换后，密切监控系统性能及日志，确保无异常情况发生。

4.6) 迁移后收尾工作

- 更新系统文档，记录整个迁移过程中的问题、解决方案和经验教训。
- 配置并完善后续监控机制（例如利用 Azure Monitor、Application Insights）。
- 根据需要关闭或退役原 Amazon RDS for Aurora MySQL 实例。

---

5) 监控与回退计划

- **监控策略**

    - 使用 Azure Monitor 和 DMS 控制台实时监控迁移任务进度和目标数据库性能指标。
    - 设置自动告警，及时通知相关人员处理异常情况。

- **回退计划**

    - 根据预定的回退策略，在遇到不可恢复错误时，迅速启动回退流程，将业务切换回原 Aurora 实例。
    - 在迁移前已对回退方案进行充分测试，确保回退过程快速有效。

---

6) 常见问题及解决方案

- **权限问题**

    - 确保 Aurora 实例上为 Azure DMS 分配了足够的权限，必要时调整相关用户权限设置。

- **网络连接问题**

    - 检查 VPN/ExpressRoute 连接和防火墙规则，确认 Azure DMS IP 范围被允许访问 MySQL 端口 3306。

- **数据一致性问题**

    - 进行充分的数据对比和校验，必要时利用数据比对工具保证源和目标数据一致。

- **性能瓶颈**

    - 在测试阶段评估数据传输速度，必要时调整 Azure DMS 或目标数据库资源配置，确保满足性能要求。

---

7) 参考链接

- 从 AWS RDS for MySQL 迁移到 Azure Database for MySQL —— 考虑事项与方法
- Azure Database Migration Service 官方文档
- Azure Database for MySQL 官方文档
- Azure 云采用框架与迁移最佳实践

3. Aurora for PostgreSQL

本指南详细描述了如何利用 Azure Database Migration Service (Azure DMS) 将 Amazon RDS for Aurora PostgreSQL 数据库迁移至 Azure Database for PostgreSQL。文档覆盖了从前期沟通、环境准备、测试迁移、正式迁移到迁移后验证等各个阶段，旨在确保整个过程平稳顺利，同时保障数据完整性和业务连续性。

对比维度	Amazon RDS for Aurora PostgreSQL	Azure Database for PostgreSQL
架构设计	云原生架构，存储与计算分离， 分布式存储层跨3个可用区 （默认6副本冗余）	灵活服务器模式支持多可用区部署， 存储与计算解耦但采用集中式存储设计 （3副本本地冗余）
数据库引擎兼容性	兼容 PostgreSQL 协议 （基于社区版优化），支持特定版本（如 PostgreSQL 15.3）	完全兼容 PostgreSQL 社区版，支持主流版本 （如 PostgreSQL 16）
性能优化	最高提供标准 PostgreSQL 3倍的吞吐量， 支持并行查询和 Aurora 优化型读取功能	支持查询加速（pg_qualstats）和自动索引优化， 性能依赖 vCore 规格
存储架构	存储自动扩展（10GB 增量，最高 128 TB），IOPS 与存储容量解耦（独立计费）	预定义存储容量（最高 16 TB），IOPS 与存储容量线性关联（每 GB 提供 3 IOPS）
扩展性	支持 15 个只读副本（自动负载均衡）， 计算节点可独立扩缩容 （1秒内完成垂直扩展）	最多支持 5 个只读副本（需手动配置）， 计算与存储需同步扩缩容（需停机维护）
高可用性	多可用区故障切换 <30 秒， 存储层自动修复数据损坏（6副本自动校验）	跨可用区部署故障切换约 60-120 秒， 存储默认本地冗余（LRS）或区域冗余（ZRS）
备份与恢复	持续增量备份（保留35天）， 支持跨区域时间点恢复（PITR）， 克隆数据库仅需数分钟	每日全量备份（保留7-35天）， 跨区域备份需手动配置，恢复速度依赖数据量
最大连接数	动态调整（最高约 16,000，例如 db.r6g.16xlarge 实例）	固定上限（如 8vCore 实例约 5,000 连接）:ml- citation{ref="7" data="citationList"}
安全特性	原生集成 IAM 数据库认证，支持 GuardDuty 威胁检测	支持 Microsoft Entra ID 集成认证，提供 Advanced Threat Protection 服务
定价策略	按计算节点规格、存储用量和 I/O 请求独立计费（例如写操作 $0.22/ 百万请求）	按统一实例规格计费（vCore + 存储组合定价），读副本按主实例 50% 费用计费
运维监控	原生集成 Performance Insights 和 DevOps Guru，支持蓝绿无感升级	提供 Query Performance Insight 和智能告警， 维护窗口需手动配置

1) 前提条件与客户确认信息

在正式启动迁移计划之前，请与客户确认以下关键信息：

- **停机窗口**
  - 确认业务可接受的最大停机时间或是否要求近零停机。
  - 协商最佳的迁移执行时间（例如低峰时段）。
- **源数据库详情（Amazon RDS for Aurora PostgreSQL）**
  - 数据库版本及兼容性（确认 Aurora PostgreSQL 版本与目标 PostgreSQL 版本间的兼容性）。
  - 数据库大小、表结构、视图、存储过程、函数、触发器、外键关系和使用的扩展模块（例如 PostGIS、pg_cron 等）。

    - 数据库负载、性能指标和未来增长预估。

- **目标数据库要求（Azure Database for PostgreSQL）**
    - 目标数据库版本（确保选择与源数据库兼容的版本）。
    - 部署区域、计算和存储规格、备份策略及高可用性要求。
    - 是否需要配置额外的安全或性能优化设置。

- **网络与安全配置**
    - 确认 AWS 与 Azure 之间的网络互联方式（例如 VPN、ExpressRoute 或 AWS Direct Connect）。
    - 检查防火墙、安全组和访问控制策略，确保 Azure DMS 能访问 Aurora 实例（默认 PostgreSQL 端口 5432）。

- **访问权限与认证**
    - 在 Aurora PostgreSQL 上为 Azure DMS 配置必要的权限（例如 SELECT、USAGE、EXECUTE 等）。
    - 确认所有连接字符串、用户名和密码均已配置正确，并可成功进行远程连接测试。

- **迁移模式选择**
    - 确定采用离线迁移（全量数据迁移）还是在线迁移（先进行全量数据迁移后捕获实时数据变更）。
    - 制定详细回退方案，确保在迁移失败时能快速恢复到原系统。

- **相关团队协调**
    - 明确 DBA、开发、运维及业务团队的职责和协调响应机制。
    - 确保测试环境已搭建完成，以便提前进行干跑测试。

---

2) 架构概述

- **数据源**：Amazon RDS for Aurora PostgreSQL
- **数据目标**：Azure Database for PostgreSQL
- **迁移工具**：Azure Database Migration Service (Azure DMS)
- **迁移模式**：离线迁移或在线迁移（依据业务需求选择）
- **网络连接**：确保 AWS 与 Azure 之间网络互联稳定
- **监控与验证**：利用 Azure Monitor 及 DMS 控制台监控迁移过程及数据库性能

---

3) 迁移前准备

3.1) 环境与数据评估

- **收集数据库信息**
    - 获取 Aurora PostgreSQL 数据库的版本、存储配置、表结构、索引、视图、存储过程、函数、触发器和外键详情。
    - 检查是否有使用特定扩展或插件，并评估其在目标系统中的支持情况。

- **性能基准与容量规划**
    - 采集当前数据库的关键性能指标，为目标实例的容量规划、资源分配和性能调优提供依据。
    - 根据数据库大小决定是否需要分批次或并行迁移数据。

- **停机与回退策略**
    - 根据业务要求确定迁移允许的停机时间窗口。
    - 制定详细的回退方案，并在测试中验证回退流程的可行性。

3.2) 网络与安全配置

- **网络互联**
  - 配置并测试 AWS 与 Azure 之间的 VPN 或 ExpressRoute 连接，确保满足带宽和延迟要求。
- **防火墙与安全策略**
  - 配置 Aurora 实例和 Azure 环境的防火墙/安全组，允许 Azure DMS 访问 PostgreSQL 默认端口（5432）。
  - 如有必要，为 Azure DMS 分配固定 IP 并在防火墙中放行。
- **权限配置**
  - 在 Aurora PostgreSQL 上创建并配置用于迁移的专用用户，授予必要的只读和元数据访问权限。

3.3) Azure 资源配置

- **目标数据库配置**
  - 在 Azure 上创建 Azure Database for PostgreSQL 实例（可选 Flexible Server 模式），并根据预期负载配置计算和存储资源。
  - 配置自动备份、监控和高可用性选项，确保数据安全和系统稳定。
- **Azure DMS 实例**
  - 在 Azure 门户中创建 Azure Database Migration Service 实例，选择支持所选迁移模式的 SKU（例如在线迁移可能需要 Premium SKU）。

---

4) 迁移操作步骤

4.1) 创建迁移项目

a. 登录 Azure 门户。
b. 导航至 **Azure Database Migration Service**。
c. 创建新的迁移项目：

- **源数据库类型**：PostgreSQL（适用于 Aurora PostgreSQL）。
- **目标数据库类型**：PostgreSQL。
- 选择迁移模式（离线或在线迁移）。

4.2) 配置源与目标端点

4.2.1) 配置源端点（Amazon RDS for Aurora PostgreSQL）

- 输入 Aurora 实例的服务器地址、端口（默认 5432）、数据库名称及具有迁移权限的用户名/密码。
- 进行端点连接测试，确保 Azure DMS 能成功连接到 Aurora 实例。

4.2.2) 配置目标端点（Azure Database for PostgreSQL）

- 填写 Azure Database for PostgreSQL 实例的服务器地址、端口、数据库名称、用户名和密码。
- 测试目标端点连接，确保配置正确无误。

4.3) 执行测试迁移（Dry Run）

- 在非生产环境中启动一次测试迁移，验证数据导出、传输及导入过程是否顺利。
- 检查迁移日志和错误报告，记录迁移速率和潜在问题，为正式迁移做充分准备。

4.4) 执行正式迁移

- 启动正式迁移任务：
  - **离线迁移**：在预定停机窗口内关闭源系统业务，进行全量数据的离线迁移。
  - **在线迁移**：先进行全量数据迁移，然后启动实时数据复制，捕捉所有后续数据变更。
- 持续监控迁移进度，通过 Azure DMS 控制台和日志信息及时处理出现的任何错误。

4.5) 数据验证与业务切换

- **数据验证**
  - 对比源与目标数据库的记录数、表结构、索引、视图、存储过程和触发器，确保数据一致性。
  - 如有需要，借助数据比对工具进行详细校验。
- **业务切换**
  - 与业务团队确认合适的切换时间窗口，将应用连接切换至目标 Azure Database for PostgreSQL。
  - 切换后密切监控系统运行情况，确保服务平稳过渡。

4.6) 迁移后整理工作

- 更新系统文档，记录迁移过程中遇到的问题及解决方案。
- 配置完善的后续监控（如使用 Azure Monitor 和 Application Insights）。
- 根据需要关闭或退役原 Amazon RDS for Aurora PostgreSQL 实例。

---

5) 监控与回退计划

- **监控策略**
  - 利用 Azure Monitor 和 Azure DMS 控制台实时监控迁移任务进度和目标数据库的性能。
  - 配置自动告警，保证在出现异常时能及时通知相关人员处理。
- **回退计划**
  - 在迁移前制定详细的回退方案，并在测试过程中进行验证。
  - 如正式迁移过程中发生不可恢复错误，迅速启动回退流程，将业务切换回原 Aurora 实例。

---

6) 常见问题及解决方案

- **权限不足**
  - 确保在 Aurora 实例中为 Azure DMS 配置了足够的权限，必要时调整相关角色和权限设置。
- **网络连接问题**
  - 检查 VPN/ExpressRoute 配置，验证防火墙规则是否允许 Azure DMS IP 范围访问默认 PostgreSQL 端口 5432。
- **数据不一致**
  - 进行充分的数据对比和校验，如有需要采用数据同步/比对工具确保源和目标数据一致。
- **性能瓶颈**
  - 在测试阶段评估数据传输速率，必要时调整 Azure DMS 或目标数据库资源配置，确保满足业务性能需求。

---

7) 参考链接

- Azure Database Migration Service 官方文档
- Azure Database for PostgreSQL 官方文档
- Azure 迁移最佳实践与云采用框架

4. RDS for MySQL

本指南详细描述了利用 Azure Database Migration Service（Azure DMS）将 Amazon RDS for MySQL 数据库迁移至 Azure Database for MySQL 的整个过程。迁移方案结合了准备、测试、正式迁移及后续验证等步骤，确保数据安全、停机时间最小并满足业务需求。

以下是两者在架构、性能、配置等方面的具体差异对比表：

对比维度	Amazon RDS for MySQL	Azure Database for MySQL
架构设计	支持多可用区部署，存储与计算分离，通过自动故障转移实现高可用性	基于单节点架构，存储与计算耦合，高可用模式依赖跨区域复制实现
数据库引擎兼容性	完全兼容 MySQL 社区版（8.0/5.7），支持自定义参数组调整内核配置	原生支持 MySQL 8.0，提供灵活服务器模式（可自定义维护窗口）
性能优化	单位 CPU 的 Sysbench QPS 可达 10,000（4vCPU 实例），支持自动读写分离	单位 CPU 的 Sysbench QPS 约 3,000（4vCPU 实例），需手动配置读写分离
存储架构	存储自动扩展（最高 64 TB），IOPS 可独立扩展（最高 160,000）	存储固定容量（最高 16 TB），IOPS 与存储容量线性关联（每 GB 3 IOPS）
扩展性	支持 5 个只读副本，垂直扩展（实例规格升级）需停机 5-10 分钟	最多 5 个只读副本，支持在线垂直扩展（存储与计算同步扩容）
高可用性	多可用区故障切换 <60 秒，通过 6 副本存储层实现数据冗余	跨区域故障切换约 2-5 分钟，存储层采用本地冗余（LRS）3 副本
备份与恢复	持续增量备份（保留 35 天），支持跨区域时间点恢复（PITR）	每日全量备份（保留 7-35 天），跨区域备份需手动配置
最大连接数	动态调整（如 db.m5.8xlarge 实例约 16,000 连接）	固定上限（如 4vCPU 实例约 3,000 连接）
安全特性	支持 IAM 数据库认证、VPC 网络隔离和透明数据加密（TDE）	集成 Microsoft Entra ID 认证，提供高级威胁防护（ATP）服务
定价策略	按计算节点规格、存储容量和 I/O 请求单独计费（如写操作 $0.20/百万请求）	统一实例规格计费（vCore + 存储组合定价），读副本按主实例 50% 费用计费
运维复杂度	自动内核优化、自动故障修复，支持蓝绿部署实现零停机升级	需自定义维护窗口，大表删除等操作可能触发性能波动

1) 前提条件与客户确认信息

在正式开始迁移前，请务必与客户确认以下信息：

- **停机窗口**

    - 确认业务可接受的最大停机时间或是否需要近零停机。
    - 规划迁移执行的最佳时间窗（例如低峰时段）。

- **源数据库详情**

    - 数据库版本（确认 Amazon RDS for MySQL 版本）。
    - 数据库大小、表结构、依赖关系及扩展情况。
    - 性能指标、当前负载情况及增长预估。

- **目标数据库要求**

    - Azure Database for MySQL 所需版本（建议选择与源数据库兼容的 MySQL 5.7 或 8.0）。
    - 目标区域、计算和存储规格、备份和高可用性策略。

- **网络与安全配置**

    - 确认 AWS 与 Azure 间的网络互连方案（VPN、ExpressRoute 或 Direct Connect）。
    - 确认相关防火墙、网络安全组以及访问控制配置，确保 Azure DMS 能够访问源数据库（默认端口 3306）。

- **访问权限**

    - 源数据库所需的权限（如 SELECT、SHOW VIEW、TRIGGER 权限）。
    - 客户端和工具所需的连接凭据是否已准备好。

- **迁移方式选择**

    - 确定是采用离线迁移还是在线迁移（在线迁移适用于数据规模较小或需要低停机时间的场景）。
    - 回退计划制定：万一出现问题时，能否快速恢复到原系统。

- **相关团队准备**

    - 确认各相关团队（DBA、开发、运维、业务）的协调与响应机制。
    - 确认测试环境已搭建完成，可用于干跑测试。

---

2) 架构概述

- **数据源**：Amazon RDS for MySQL
- **数据目标**：Azure Database for MySQL（推荐使用 Flexible Server 模式）
- **迁移工具**：Azure Database Migration Service（Azure DMS）
- **迁移方式**：离线迁移或在线迁移（根据业务需求选择）
- **网络连接**：确保 AWS 与 Azure 之间的网络互联正常
- **监控与验证**：使用 Azure Monitor 和 DMS 控制台对迁移过程及后续运行状态进行监控

---

3) 迁移前准备

3.1) 环境与规划

- **数据评估**

    - 收集源数据库的版本、大小、表结构、索引及关联信息。
    - 评估应用和数据库间的依赖关系，确认是否存在特殊扩展或存储过程。

- **性能基准**

    - 在迁移前采集关键性能指标，便于在目标实例中进行容量规划和性能调优。

- **停机与回退规划**
  - 与业务确认允许的停机时间，规划执行窗口。
  - 制定回退方案，确保迁移过程中如遇故障能迅速切回原环境。

3.2) 网络和安全配置

- **网络连通性**
  - 配置 VPN 隧道、ExpressRoute 或其他跨云连接方案，测试 AWS 与 Azure 之间的网络延迟和吞吐量。
- **防火墙及安全组**
  - 配置源数据库及 Azure 环境的安全策略，确保 Azure DMS IP 范围获得访问权限（默认端口 3306）。
- **权限及认证**
  - 在 Amazon RDS for MySQL 上分配相应权限（READ、SELECT、SHOW VIEW、TRIGGER），供数据提取使用。

3.3) Azure 资源配置

- **目标数据库**
  - 在 Azure 上创建 Azure Database for MySQL 实例（建议使用 Flexible Server 模式）。
  - 根据性能基准规划分配计算、存储资源，并设置备份、监控和高可用性策略。
- **Azure DMS 实例**
  - 在 Azure 门户中创建 Azure Database Migration Service 实例，并确保选取适当 SKU（例如在线迁移需要 Premium SKU）。

---

4) 迁移操作步骤

4.1) 创建迁移项目

a. 登录 Azure 门户。
b. 导航至 **Azure Database Migration Service**。
c. 创建新的迁移项目：

- 选择 **源数据库类型：MySQL**。
- 选择 **目标数据库类型：MySQL**。
- 选择迁移模式（离线或在线）。

4.2) 配置源与目标端点

4.2.1) 配置源端点（Amazon RDS for MySQL）

- 输入服务器地址、端口（3306）、数据库名称及具有相应权限的用户名/密码。
- 进行端点连接测试，确保 Azure DMS 能够访问 Amazon RDS。

4.2.2) 配置目标端点（Azure Database for MySQL）

- 填写 Azure Database for MySQL 实例的名称、服务器地址、端口、数据库名、用户名及密码。
- 进行连接测试，确保配置正确无误。

4.3) 执行测试迁移（Dry Run）

- 在非生产环境中启动一次测试迁移，验证数据导出、传输及导入过程。
- 检查迁移日志，确认无错误，记录数据传输速度及迁移耗时，为正式迁移做参考。

4.4) 执行正式迁移

- 启动正式迁移任务：
  - **离线迁移**：在停机窗口内关闭源系统，执行全量数据迁移。
  - **在线迁移**：初始化数据后开启持续数据复制，确保在数据同步阶段捕获所有变更。
- 持续监控迁移进度，关注 Azure DMS 控制台及日志输出。

4.5) 数据验证与切换

- **数据验证**
  - 对比源与目标数据库的数据完整性（例如表记录数、关键数据校验）。
  - 验证数据库结构、索引、存储过程及触发器是否一致。
- **业务切换**
  - 与业务团队协商合适的切换时机，将应用连接切换至 Azure Database for MySQL 实例。
  - 切换后密切监控系统运行状态，确保无异常。

4.6 迁移后收尾工作

- 更新系统文档，记录迁移过程中遇到的问题及解决办法。
- 配置后续监控（使用 Azure Monitor、Application Insights 等工具）。
- 根据需要关闭或退役原有的 Amazon RDS for MySQL 实例。

---

5) 监控与回退计划

- **监控策略**
  - 利用 Azure Monitor 及 DMS 控制台实时监控迁移任务和数据库性能指标。
  - 设置警报，确保在迁移过程或切换后能够迅速响应异常情况。
- **回退计划**
  - 事先制定完善的回退方案，并在测试中验证回退流程。
  - 若正式迁移过程中出现不可恢复错误，立即启动回退流程，将业务切换回 Amazon RDS for MySQL。

---

6) 常见问题及解决方法

- **权限不足**
  - 确保在源数据库中为 Azure DMS 分配足够的读取权限；根据日志提示调整用户权限。
- **网络连接失败**
  - 检查 VPN/ExpressRoute 配置，确认防火墙规则允许相关 IP 范围访问源数据库。
- **数据一致性问题**
  - 执行充分的数据验证工作，必要时使用数据比对工具确保目标数据库与源数据库一致。
- **性能瓶颈**

- 在测试阶段检查数据传输速率，必要时增加 Azure DMS 或目标数据库资源配置。

7.) 参考链接

- 从 AWS RDS for MySQL 迁移到 Azure Database for MySQL —— 考虑事项与方法
- Azure Database Migration Service 官方文档
- Azure Database for MySQL 官方文档

5. RDS for PostgreSQL

本指南详细描述了利用 Azure Database Migration Service (Azure DMS) 将 Amazon RDS for PostgreSQL 数据库迁移至 Azure Database for PostgreSQL 的完整过程。文档覆盖了从前期确认、环境准备、测试迁移、正式迁移到迁移后验证的各个阶段，旨在确保数据安全、停机时间最小，并满足业务需求。

以下是两者在核心特性、架构设计、性能指标等方面的详细差异对比：

对比维度	Amazon RDS for PostgreSQL	Azure Database for PostgreSQL
架构设计	基于传统主从架构，存储与计算耦合； 支持多可用区部署，但需手动配置跨区域复制	灵活服务器模式支持存储与计算解耦，自动跨可用区部署（ZRS冗余），提供原生读写分离功能
存储配置	最大支持 64 TB 存储容量，IOPS 可独立扩展（最高 80,000），支持存储自动扩展	存储容量固定（最高 16 TB），IOPS 与存储容量线性关联（每 GB 3 IOPS），需预定义存储大小
性能指标	单实例最大支持 96 vCPU，提供 T 系列突发性能实例；Sysbench 测试 QPS 可达 50,000+	单实例最高 64 vCPU，支持内存优化型实例（如 E64ds_v5）；Sysbench 测试 QPS 约 35,000
高可用性	多可用区部署故障切换时间 <60 秒，基于物理复制实现；自动备份保留周期 1-35 天	跨可用区故障切换约 2-5 分钟，基于异步逻辑复制；自动备份保留周期 7-35 天
扩展能力	支持 15 个只读副本，支持跨区域只读副本；垂直扩展需停机 5-10 分钟	最多 5 个只读副本，支持在线垂直扩展（计算与存储同步扩容）；横向扩展需手动配置
网络延迟	默认 VPC 部署，同区域实例间延迟 <1 ms；跨区域复制延迟约 100-200 ms	虚拟网络（VNet）集成，同区域延迟 <2 ms；跨区域复制延迟约 150-300 ms
安全特性	原生集成 IAM 角色认证，支持 KMS 加密和 Secrets Manager；审计日志保留最长 7 年	支持 Microsoft Entra ID 认证，提供透明数据加密（TDE）；审计日志保留周期 7-90 天
监控功能	提供 Enhanced Monitoring 细粒度监控（1秒级指标），集成 CloudWatch 告警系统	内置 Query Performance Insight 分析工具，支持 1 分钟级监控粒度，告警系统集成 Azure Monitor
成本结构	按计算节点规格（如 db.m6g.16xlarge）、存储用量（$0.115/GB/月）和 I/O 请求单独计费	统一实例规格计费（如 GP_Gen5_16），存储（$0.12/GB/月）与计算绑定定价；读副本按主实例 50% 收费
维护窗口	支持自定义维护时段（30分钟窗口），自动小版本升级；大版本升级需手动触发	维护时段固定为 1 小时，支持自动补丁安装；大版本升级需创建新实例迁移数据

---

1) 前提条件与客户确认信息

在正式开始迁移前，请务必与客户确认以下信息：

- **停机窗口**
  - 确认业务可接受的最大停机时间（或是否需要近零停机）。
  - 协商迁移执行的最佳时间（例如低峰时段）。
- **源数据库详情**
  - 数据库版本（确认 Amazon RDS for PostgreSQL 版本）。
  - 数据库大小、表结构、外键、存储过程、扩展模块及依赖关系。
  - 性能指标、当前负载情况和未来增长预估。
- **目标数据库要求**
  - Azure Database for PostgreSQL 版本（建议选择与源数据库兼容的版本）。
  - 目标区域、计算与存储规格、备份及高可用性要求。
  - 是否需要多节点或高可用性配置。
- **网络与安全配置**
  - 确认 AWS 与 Azure 之间的网络互联方案（例如 VPN、ExpressRoute 或 AWS Direct Connect）。
  - 检查防火墙、安全组与访问控制策略，确保 Azure DMS 能访问源数据库（默认 PostgreSQL 端口 5432）。
- **访问权限与认证**
  - 在 Amazon RDS for PostgreSQL 上为 Azure DMS 分配所需权限（如 SELECT、USAGE、EXECUTE 等）。
  - 确认源端与目标端的用户名、密码及连接字符串已正确配置。
- **迁移方式选择**
  - 确定采用离线迁移（全量数据迁移）还是在线迁移（初始数据迁移后持续数据复制，适用于需要最小停机的场景）。
  - 制定回退方案，确保在出现问题时可以快速恢复到原系统。
- **相关团队协调**
  - 确认 DBA、开发、运维及业务团队的职责与响应机制。
  - 确保测试环境已搭建并提前进行干跑测试。

---

2) 架构概述

- **数据源**：Amazon RDS for PostgreSQL
- **数据目标**：Azure Database for PostgreSQL
- **迁移工具**：Azure Database Migration Service (Azure DMS)
- **迁移方式**：离线迁移或在线迁移（依据业务需求选择）
- **网络连通性**：确保 AWS 与 Azure 之间的网络互联畅通
- **监控与验证**：利用 Azure Monitor 及 Azure DMS 控制台实时跟踪迁移进程与系统性能

---

3) 迁移前准备

3.1) 环境与规划

- **数据评估**
  - 收集源数据库的版本、大小、表结构、索引、外键关系及扩展模块使用情况。
  - 确认是否有依赖特定 PostgreSQL 扩展（如 PostGIS、pg_cron 等）。

- **性能基准**
  - 采集当前数据库的关键性能指标，便于目标系统容量规划与性能调优。
- **停机与回退规划**
  - 确定可接受的停机时间，规划合理的迁移窗口。
  - 制定回退方案，在迁移过程中如遇问题可迅速切换回原系统。

3.2) 网络与安全配置

- **网络连通性**
  - 配置并测试 AWS 与 Azure 之间的 VPN/ExpressRoute 连接，确保延迟与带宽满足迁移要求。
- **防火墙与安全组**
  - 配置 Amazon RDS 以及 Azure 环境的防火墙策略，允许 Azure DMS 访问 PostgreSQL 默认端口 5432。
- **权限配置**
  - 在 Amazon RDS for PostgreSQL 上确保为 Azure DMS 分配足够的权限（例如 SELECT、USAGE 和 EXECUTE 权限）。

3.3) Azure 资源配置

- **目标数据库**
  - 在 Azure 上创建 Azure Database for PostgreSQL 实例，根据预期负载配置合适的计算、存储以及高可用性选项。
  - 配置备份、监控和安全策略以保障数据安全。
- **Azure DMS 实例**
  - 在 Azure 门户中创建 Azure Database Migration Service 实例，选择合适的 SKU 以支持所选的迁移方式（在线迁移可能需要 Premium SKU）。

---

4). 迁移操作步骤

4.1) 创建迁移项目

a. 登录 Azure 门户。
b. 导航至 **Azure Database Migration Service**。
c. 创建新的迁移项目：

- 选择 **源数据库类型：PostgreSQL**；
- 选择 **目标数据库类型：PostgreSQL**；
- 选择迁移模式（离线或在线）。

4.2) 配置源与目标端点

4.2.1) 配置源端点（Amazon RDS for PostgreSQL）

- 输入源数据库的服务器地址、端口（通常为 5432）、数据库名称及具有相应权限的用户名/密码。
- 执行端点连接测试，确保 Azure DMS 能成功连接至 Amazon RDS。

4.2.2) 配置目标端点（Azure Database for PostgreSQL）

- 填写 Azure Database for PostgreSQL 实例的服务器地址、端口、数据库名称、用户名和密码。
- 测试连接，确保端点配置正确。

4.3) 执行测试迁移（Dry Run）

- 在非生产环境中启动一次测试迁移，验证数据导出、传输和导入过程是否正常。
- 检查迁移日志，记录迁移速度、潜在错误和数据一致性情况，为正式迁移做充分准备。

4.4) 执行正式迁移

- 启动正式迁移任务：
  - **离线迁移**：在预定停机窗口内关闭源系统业务，执行全量数据的离线迁移。
  - **在线迁移**：初始化全量数据迁移后，启动持续数据复制，确保捕获所有实时数据变更。
- 持续监控迁移进度，注意 Azure DMS 控制台和日志中的错误提示。

4.5) 数据验证与业务切换

- **数据验证**
  - 对比源数据库和目标数据库的记录数、表结构、索引、外键和视图等，确保数据一致性。
  - 如有必要，使用数据比对工具进行详细校验。
- **业务切换**
  - 与业务团队确认切换时间，将应用连接切换至 Azure Database for PostgreSQL 实例。
  - 切换后实时监控系统性能，确保无异常现象。

4.6) 迁移后整理工作

- 更新相关文档，记录迁移过程中的问题及解决方法。
- 配置和调整后续监控机制（例如使用 Azure Monitor、Application Insights）。
- 根据需要停用或退役原 Amazon RDS for PostgreSQL 实例。

---

5) 监控与回退计划

- **监控策略**
  - 利用 Azure Monitor 和 DMS 控制台实时监控迁移任务进度和目标数据库性能。
  - 设置告警，当出现异常时能及时通知运维人员进行处理。
- **回退计划**
  - 在迁移前已制定好完善的回退方案，并通过测试验证。
  - 如果正式迁移过程中发生不可恢复错误，迅速启动回退流程，将业务切换回原系统。

---

6) 常见问题及解决方案

- **权限不足**
  - 确认在源数据库上为 Azure DMS 配置了足够的权限，必要时调整相应用户的权限设置。
- **网络连接问题**

    - 检查 VPN/ExpressRoute 配置，验证防火墙或网络安全组规则，确保允许 Azure DMS IP 范围访问 PostgreSQL 端口 5432。

- **数据不一致**
    - 迁移前后进行详细的数据对比，必要时使用数据同步或比对工具，确保目标数据库与源数据库数据一致。

- **性能瓶颈**
    - 在测试阶段评估数据传输速率，必要时调整 Azure DMS 或目标数据库的资源配置，确保满足性能需求。

---

7) 参考链接

- Azure Database Migration Service 官方文档
- Azure Database for PostgreSQL 官方文档
- Azure 迁移指南与最佳实践

6. Redis (ElastiCache)

AWS Redis 向 Azure Cache for Redis 在线迁移手册（使用 redis-shake)

本文档详细说明如何在不中断业务的情况下，将 AWS 上运行的 Redis 服务（ElastiCache for Redis）在线迁移至 Azure Cache for Redis，主要采用阿里开源的 redis-shake 工具。内容涵盖迁移背景、前期准备、数据同步流程、流量切换、SKU 选择及后续维护策略。

迁移步骤大纲

- **部署工具**
    - 在能同时访问 AWS Redis 和 Azure Cache for Redis 的服务器上部署 redis-shake 工具

- **数据同步**
    - 配置 redis-shake 进行全量数据同步和实时增量同步
    - 启动工具并监控同步进度

- **流量切换与验证**
    - 验证数据一致性
    - 渐进式将业务流量切换到 Azure Cache for Redis

- **后续维护**
    - 持续监控性能指标与系统资源
    - 定期备份、日志审计以及版本更新

---

1) 迁移背景与目标

- **背景**：当前 AWS 环境中使用 ElastiCache for Redis 提供缓存服务，计划迁移至 Azure Cache for Redis，以实现平台整合、降低运维成本并利用 Azure 生态优势。
- **目标**：实现数据在线实时同步、确保数据一致性和平稳过渡，尽量减少停机时间，同时优化性能配置以满足生产需求。

---

2) 迁移前准备工作

2.1) 数据评估与备份

- **评估现状**：确认 AWS Redis 实例的版本、数据规模、持久化模式（RDB/AOF）及相关配置参数。
- **数据备份**：建议提前使用 `BGSAVE` 生成 RDB 备份，确保在紧急情况下可恢复数据。

2.2) 环境规划与网络配置

- **迁移窗口**：选取业务低峰期进行迁移，便于监控和调整。
- **网络与权限配置**：
  - 确保部署工具的服务器能同时访问 AWS Redis（源）和 Azure Cache for Redis（目标）。
  - **重点考虑 VPN 或内网打通**：若源目标位于不同网络环境中，通过 VPN 或内网互联构建稳定、安全的数据通道。
  - 配置 Azure 虚拟网络（VNet）、防火墙和子网，确保连接权限正确，并根据实际需求配置网络带宽与容错机制。

---

3) 在线迁移方案（使用 redis-shake）

3.1) 工具介绍

- **redis-shake**
  阿里开源的 redis-shake 支持全量数据导入与实时增量同步，可在不中断业务的情况下完成数据在线迁移。更多详情请参考 RedisShake GitHub 项目。

3.2) 部署与环境准备

a. **安装工具**

- 在一台能同时访问 AWS 与 Azure 的服务器上下载并编译（或直接使用预编译版本） redis-shake，确保版本与当前 Redis 环境兼容。

b. **网络与权限验证**

- 配置服务器网络，确保稳定访问 AWS Redis 和 Azure Cache for Redis 的各项端口。
- 如前所述，建议通过 VPN 或内网打通实现跨环境稳定连接。
- 完善所有认证信息（例如密码、TLS 证书等）以避免同步过程中出现权限问题。

3.3 配置与启动数据同步

a. **调整配置文件**

- 修改 redis-shake 配置文件（如 `redis-shake.conf` 或命令行参数方式），设置关键参数：
  - **源端**：AWS Redis 的 IP、端口、认证信息及集群参数（如适用）。
  - **目标端**：Azure Cache for Redis 的连接地址、端口、认证信息等。
  - **同步模式**：选择全量数据同步+增量实时同步，保证全量数据导入后捕获业务写入的所有更新。
  - **并发与超时**：根据数据量调整批处理大小、同步并发数和超时设置，以确保传输高效稳定。

b. **启动同步进程**

- 启动 redis-shake，并实时监控日志和同步进度，确认全量数据完成后增量同步正常进行。

c. **监控与调优**

- 实时查看 AWS 和 Azure 两端的 Redis 性能指标（内存使用、连接数、命中率等）以及网络延迟情况。
- 根据实际情况调整 redis-shake 的并发数、批次大小和超时参数，达到最佳同步效果。

4) 流量切换与数据验证

- **流量切换**
  - 当数据同步稳定且经过校验后，逐步将业务流量从 AWS Redis 切换至 Azure Cache for Redis。建议采用渐进式切换策略，由部分流量先行验证新实例，再全面切换。
- **数据一致性验证**
  - 结合应用日志、 Redis 内置工具（如 `INFO` 命令）及监控数据，对比检查数据一致性与延迟，确保切换过程中数据完整无缺。

---

5) Azure Cache for Redis 部署与 SKU 选择

5.1) SKU 概览

Azure Cache for Redis 提供以下三种主要 SKU：

- **Basic**：适用于开发、测试环境，单节点，不支持高可用。
- **Standard**：适用于生产环境，支持主从复制及自动故障转移，实现基本的高可用。
- **Premium**：适用于高性能、高可用要求，支持数据持久化、VNet 集成、Redis 集群模式及地理复制。

5.2) SKU 选择建议

- 根据 AWS Redis 的当前内存使用量、连接数以及业务流量需求选择合适 SKU；
- 对于高并发或写入频繁的场景，建议选择 Premium SKU 以获得更低延迟和更高吞吐量；
- 生产环境建议至少使用 Standard 版本，并可根据数据持久化及高可用需求升级至 Premium。

---

6) 性能调优与后续维护

6.1) 性能调优

- **关键指标**：重点监控缓存命中率、响应延迟和内存使用率，目标命中率在 90% 以上。
- **参数调整**：根据业务负载合理设置 Redis 最大连接数、请求超时和批处理参数；必要时采用 Redis 集群分片策略以实现负载均衡。
- **监控工具**：利用 Azure Monitor 和 Redis 自身的 `INFO` 命令，定期查看同步状态和性能指标，并根据数据调整配置。

6.2) 后续维护

- **定期备份与审计**：配置自动备份策略和日志审计，确保数据安全；
- **版本升级**：关注 Redis 官方更新，及时应用补丁和版本升级；
- **持续调优**：根据监控数据动态调整实例配置，如有需要可升级 SKU 或实施 Redis 分片部署。

---

7) 常见问题与故障排除

- **数据同步延迟或丢失**：检查 redis-shake 日志与配置，确认全量及增量同步设置正确，同时核实网络和认证信息；
- **网络连接问题**：确保 VPN 或内网打通配置正常，必要时调整网络带宽及防火墙策略；
- **系统负载过高**：及时调整 Azure Redis 实例的 SKU 与资源配额，避免因资源瓶颈导致性能下降。

---

8) 参考文档

- Azure Cache for Redis 官方文档
- Azure 定价计算器
- 阿里云 RedisShake GitHub 项目
- Redis 性能优化最佳实践

注：本手册为在线迁移的基础指导，实际实施时请根据具体业务场景、数据规模和系统架构进行详细规划及充分测试，确保迁移平稳且数据一致。

7. SQS

1. 环境准备

a. **梳理当前使用 SQS 的系统架构及消息队列使用场景**

- 分析现有系统中 SQS 的使用情况，包括消息的生产者、消费者、消息类型、处理流程等。
- 确定消息的流向、处理逻辑，以及与其他系统组件的交互方式。

b. **确定关键指标**

- **消息量**：估算系统的日常消息吞吐量，包括峰值和平均值。
- **消息大小**：确定单条消息的平均大小和最大大小。
- **延迟要求**：明确系统对消息处理延迟的容忍度。
- **消息持久性要求**：确定消息在队列中的保留时间，以及是否需要持久化存储。
- **可靠性要求**：定义消息的可靠性需求，如至少一次、至多一次或恰好一次的处理语义。
- **消息保留期**：设定消息在队列中的最大保留时间。

c. **明确现有 SQS 使用的队列类型**

- 确定当前使用的是标准队列还是 FIFO 队列，以便在 Azure Service Bus 中选择相应的队列类型。

d. **在 Azure Service Bus 中创建相应的队列类型**

- 根据上述需求，使用 Azure 门户、CLI 或 ARM 模板创建适当类型的队列或主题订阅。例如：
  - **标准队列**：适用于大多数场景，提供至少一次的消息传递保证。
  - **FIFO 队列**：适用于需要严格消息顺序和精确一次处理的场景，通过使用消息会话实现FIFO。
  - **主题和订阅**：适用于发布/订阅模式的消息传递。

Feature	Amazon SQS	Azure Service Bus
Queue Type	Standard & FIFO Queue	Queue & Topic/Subscription (Standard/Premium)
Message Size Limit	256 KB	256 KB (Standard), 1 MB (Premium)
Message Retention	Up to 14 days	Up to 14 days
Ordering	FIFO Queue supports strict ordering	Queues with sessions support ordering
Dead-Letter Queues	Supported	Supported
Authentication & Authorization	AWS IAM	Azure AD & Shared Access Signatures

---

Feature	Amazon SQS	Azure Service Bus
Transaction support	Limited	Supported (Premium)

2) 修改代码

- **集成 Azure Service Bus SDK**：在 Golang 应用中，引入 Azure Service Bus 的 Go SDK，以便与 Azure Service Bus 进行交互。
  - 使用 azservicebus 包来发送和接收消息。
  - 参考 Azure 官方文档，了解如何在 Go 中使用 Azure Service Bus：
- **编写消息发送逻辑**：实现将消息发送到 Azure Service Bus 队列或主题的功能。
  - 创建客户端连接 Azure Service Bus。
  - 编写发送消息的函数，处理消息的构建和发送过程。
- **编写消息消费逻辑**：实现从 Azure Service Bus 队列或订阅中接收和处理消息的功能。
  - 创建接收器连接 Azure Service Bus。
  - 编写接收消息的函数，处理消息的接收和确认过程。

3) 切换写入

- **更新消息生产者**：修改应用程序的消息生产者部分，将消息发送目标从 SQS 更改为 Azure Service Bus。
  - 确保消息格式和内容与之前在 SQS 中使用的一致。
  - 验证消息发送的可靠性和性能。

4) 并行消费

- **并行消费逻辑**：在迁移过程中，保持从 SQS 和 Azure Service Bus 两个消息源同时消费消息，以确保系统的连续性和消息的完整性。
  - 实现并行消费者，分别从 SQS 和 Azure Service Bus 接收消息。
  - 确保消息处理的幂等性，避免重复处理。
  - 监控消息处理的状态，确保无消息丢失。

5) 切换消费

- **完成迁移**：在确认所有消息生产和消费都已成功切换到 Azure Service Bus，并且系统稳定运行后，停止从 SQS 消费消息。
  - 关闭或删除 SQS 消费者，释放相关资源。
  - 确保 Azure Service Bus 的队列和订阅配置满足系统的性能和可靠性需求。

注意事项：

- **消息顺序和重复性**：如果应用程序对消息的顺序和重复处理有严格要求，建议使用 Azure Service Bus 的 FIFO 队列和消息去重功能。
- **性能优化**：根据系统的消息量和处理需求，调整 Azure Service Bus 的性能设置，如预取数量、消息批处理等。
- **监控和日志**：启用 Azure Monitor 和日志记录功能，实时监控消息传递的健康状况和性能指标。
- **灾难恢复**：考虑在多个区域部署 Azure Service Bus 实例，实现灾难恢复和高可用性。

6) 参考文档

- 使用 SDK for Go V2 的 Amazon SQS 示例
- Amazon SQS code examples for the SDK for Go V2
- 向/从 Azure 服务总线队列发送/接收消息 (Go)
- Azure Service Bus client module for Go

8. Memcached (ElastiCache)

本文档旨在指导客户从 **Amazon ElastiCache for Memcached** 迁移到 **Azure Marketplace Memcached**，确保数据正确切换、业务低影响并满足性能与安全性要求。

1) 客户确认信息

在启动迁移前，请客户确认以下信息：

当前环境与配置

- 使用的 Memcached 版本号
- 集群节点数、内存配置、实例类型
- 所在 AWS 区域及网络配置（如 VPC 子网）
- 应用访问方式、端口及认证机制（如有）

业务和系统需求

- 是否需要数据持久化（Memcached 通常为无状态缓存）
- 应用是否能支持新终端节点和配置
- 性能指标和 SLA 要求
- 可接受的停机窗口时间

安全与网络

- Azure 虚拟网络及子网配置
- 网络连通性、端口开放策略
- IP 白名单或其他访问控制手段

备份与回退

- 当前环境的备份方案
- 回退流程和预案
- 监控与日志机制

2) 迁移背景与目标

背景说明

当前缓存服务基于 **Amazon ElastiCache for Memcached**。因架构调整需迁移至 **Azure Marketplace Memcached** 实例。

迁移目标

- 部署 Azure 上的 Memcached 服务
- 平滑切换客户端连接至新服务
- 实现业务不中断的无缝迁移

3) 迁移前准备

3.1) 环境准备

- 创建 Azure 订阅、资源组
- 配置 Azure 虚拟网络与子网
- 在 Azure Marketplace 部署 Memcached 实例

- 配置防火墙及安全组（开放 11211 端口）

3.2) 应用准备

- 检查客户端是否支持新配置
- 在测试环境验证兼容性
- 规划迁移时段（建议业务低峰期）
- 提前通知运维、开发、业务方

---

4) 架构规划与差异分析

4.1) 架构差异

项目	Amazon ElastiCache	Azure Marketplace Memcached
托管类型	全托管	多为自托管或第三方部署
网络集成	AWS VPC 集成	Azure 虚拟网络、自定义配置
高可用	跨 AZ 支持	需自建或脚本实现
监控支持	原生 CloudWatch	需自建或使用 Azure Monitor

4.2) 规划要点

- 确保性能参数与 AWS 相近
- 配置高可用方案（如负载均衡、冗余节点）
- 设置监控与日志收集机制
- 预留充足测试时间

---

5) 迁移步骤

5.1) 部署 Azure Memcached 实例

a. 登录 Azure 门户，搜索并部署 “Memcached”
b. 选择合适规格（CPU、内存、节点）
c. 配置网络及安全组
d. 初始化参数，如 maxmemory、connections 等

5.2) 应用端更新配置

a. 修改客户端配置文件或连接代码
b. 测试与新实例连接，验证写入/读取
c. 压测新集群性能，与 AWS 对比
d. 调整参数优化响应时间

5.3) 切换业务流量

a. 设定停机或窗口时间
b. 更新 DNS 或反向代理配置指向 Azure

c. 保留 AWS 实例短暂在线，缓冲故障回退
d. 观察系统指标，确认切换成功

---

6) 迁移后验证与测试

- 应用连接验证：读写缓存是否正常
- 性能监控：内存占用、连接数、响应时间
- 日志分析：查看错误与命中率
- 关键功能测试：确保业务流程未中断

---

7) 回退计划

1. 在切换前保持 AWS 实例未下线
2. 若异常，快速将 DNS/连接配置切回原环境
3. 记录问题根因与指标差异
4. 修复并重新规划迁移时间

---

8) 运维建议

- 配置 Azure Monitor 或第三方监控
- 设置告警阈值，如命中率 < 70% 等
- 定期演练切换与回退流程
- 审查安全组和访问控制策略

---

9) 参考链接

- Amazon ElastiCache for Memcached
- Azure 虚拟网络文档

---

9. Kafka

迁移前确认云服务内容

在开始 Kafka 迁移前，请确认以下内容，确保 AWS 与 Azure 之间的配置和网络互通：

- **AWS 部分：**
  - 确认 Kafka 集群（例如 Amazon MSK）的版本、主题配置、分区、消费者组和安全策略
  - 检查数据持久化、备份及日志记录配置，确保能顺利导出数据
  - 验证网络带宽、VPN 或内网互联配置，确保数据复制期间的稳定性
- **Azure 部分：**
  - 如果采用 Event Hub，请确认 Azure Event Hub 已创建，并设置正确的配额、策略及对 Kafka 协议的支持
  - 如果采用 Kafka 镜像部署，请规划并验证在 Azure 上部署 Kafka 集群的方案（例如使用 Azure 虚拟机、AKS 或 Marketplace 镜像），并配置安全组和网络策略
  - 配置所需的存储和监控，确保集群部署后的可持续运维

Kafka AWS 到 Azure 迁移方案

本方案提供两种迁移策略，帮助您将 AWS 上的 Kafka 数据和服务迁移到 Azure：
a. 使用 **Azure Event Hub**（内建 Kafka 协议支持）
b. 通过 **Azure 上的 Kafka 镜像部署**（自建 Kafka 集群并使用数据镜像工具实现数据复制）

---

1) 迁移到 Azure Event Hub

前期准备

- **Event Hub 配置：**
  - 在 Azure 门户中创建一个 Event Hub 命名空间，并在其中创建 Event Hub 实例
  - 启用 Kafka 协议支持（通常默认支持，只需确保使用正确的端口 9093）
- **应用准备：**
  - 更新 Kafka 生产者和消费者应用的连接配置，使其指向 Azure Event Hub 的 Endpoint（例如：`<YourEventHubNamespace>.servicebus.windows.net:9093`）
  - 配置认证方式，通常采用 SAS tokens 或 Azure AD 认证

数据迁移步骤与技术细节

a. **修改生产者/消费者配置**

- 在应用配置文件（如 `application.properties`、`config.yml` 或环境变量）中，将 Kafka 的 `bootstrap.servers` 参数更新为 Event Hub 地址。例如：
  `bootstrap.servers=<YourEventHubNamespace>.servicebus.windows.net:9093`
- 更新认证信息：配置 SAS 密钥，如下所示（Java 示例）：
  `java Properties props = new Properties(); props.put("bootstrap.servers", "<YourEventHubNamespace>.servicebus.windows.net:9093"); props.put("security.protocol", "SASL_SSL"); props.put("sasl.mechanism", "PLAIN"); props.put("sasl.jaas.config", "org.apache.kafka.common.security.plain.PlainLoginModule required " + "username=\"$ConnectionString\" password=\"Endpoint=sb://<YourEventHubNamespace>.servicebus.windows.net/;SharedAccessKeyName=<YourKeyName>;SharedAccessKey=<YourAccessKey>\";");`

b. **实时数据切换**

- 对于生产者：直接改写代码更新 broker 地址，发送新消息到 Event Hub
- 对于消费者：更新订阅的 broker 地址，使其从 Event Hub 中读取消息

c. **历史数据迁移**

- 部署 Kafka Connect 集群，使用 Azure Event Hub Sink Connector将历史数据批量导入。
- 示例 Sink Connector 配置（JSON 格式）：
  `json { "name": "eventhub-sink-connector", "config": { "connector.class": "com.microsoft.azure.eventhubs.connect.EventHubsSinkConnector", "tasks.max": "1", "topics": "your-topic", "eventhubs.connection.string": "<Your_EventHub_Connection_String>", "eventhubs.event.hub.name": "<Your_EventHub_Name>", "errors.tolerance": "all", "errors.log.enable": "true" } }`
- 该配置会将指定主题的历史数据导入到 Azure Event Hub 中。

d. **测试与验证**

- 在测试环境中模拟消息生产和消费，使用工具如 Kafka Console Producer 和 Consumer 验证数据传输
- 检查 Azure Monitor 中 Event Hub 的指标（延时、吞吐量），确保符合预期

后续维护

- 配置 Azure Monitor 来跟踪消息延时、吞吐量及错误情况
- 定期更新 SAS 密钥和访问控制策略，保证通信安全

---

2) 迁移到 Azure 上的 Kafka 镜像部署

前期准备

- **部署方案：**
  - 选择在 Azure 部署 Kafka 集群的方式（使用 Azure 虚拟机、AKS 或 Marketplace 镜像）
  - 确定目标集群的配置（主题、分区、复制因子），确保与 AWS Kafka 匹配
- **网络配置：**
  - 建立 AWS Kafka 与 Azure Kafka 集群间的 VPN 或内网互联，确保数据复制安全、低延迟

数据迁移步骤与技术细节

a. **部署目标 Kafka 集群**

- 使用预构建的 Docker 镜像或 Azure Marketplace 镜像在 Azure 上快速部署 Kafka 集群
- 配置相应的 Zookeeper 服务（或使用 Kafka 自带的 KRaft 模式），确保所有节点正确注册
- 示例命令（使用 Docker Compose 部署 Kafka 集群）：
  `yaml version: '3' services: zookeeper: image: confluentinc/cp-zookeeper:latest environment: ZOOKEEPER_CLIENT_PORT: 2181 kafka: image: confluentinc/cp-kafka:latest depends_on: - zookeeper environment: KAFKA_BROKER_ID: 1 KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181 KAFKA_LISTENERS: PLAINTEXT://:9092 KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://<your_azure_kafka_ip>:9092 ports: - "9092:9092"`

b. **配置 MirrorMaker 数据镜像**

- 部署 Kafka MirrorMaker 或 MirrorMaker 2 作为数据复制工具。
- 示例 MirrorMaker 2 配置文件（`mirror-maker.properties`）：

```
clusters = source, target
source.bootstrap.servers = source-broker1:9092,source-broker2:9092
target.bootstrap.servers = target-broker1:9092,target-broker2:9092

source->target.enabled = true
source->target.topics = .*
source->target.checkpoint.interval.seconds = 60
source->target.replication.factor = 3
```

- 启动 MirrorMaker 2 任务：
  `bash ./bin/connect-mirror-maker.sh config/mirror-maker.properties`
- 监控 MirrorMaker 的日志，检查消息复制延迟和错误信息，确保数据在两个集群间一致同步

c. **更新应用配置**

- 修改生产者和消费者的配置，将连接地址更新为 Azure Kafka 集群的 broker 地址
- 如有需要，可采用负载均衡器或 DNS 切换策略以简化应用配置的更新

d. **测试与验证**

- 在测试环境中生产一些测试数据，观察 MirrorMaker 复制情况，验证目标集群中各主题数据与原集群一致
- 使用 Kafka Console Consumer 验证数据读取的正确性，确保延迟和吞吐量满足业务需求

后续维护

- 建立定期快照和备份机制，防止数据丢失
- 采用监控工具（如 Prometheus、Grafana）定期监控 Kafka 集群关键指标（CPU、内存、磁盘 I/O、查询延时）
- 定期维护 MirrorMaker 任务，调整同步策略以适应流量变化

微软迁移参考链接

- Azure Event Hub 官方文档
- 使用 Kafka 协议的 Azure Event Hubs 迁移指南
- 在 Azure 上部署 Apache Kafka 的最佳实践

注：请选择适合您的业务需求和实时性要求的方案，并确保在迁移过程中充分测试所有数据复制和应用切换流程，以保障系统平稳、安全地完成迁移。

10. OpenSearch

迁移前确认云服务内容

在开始迁移 OpenSearch 数据及索引之前，请确认以下各项内容，以确保迁移过程中源与目标平台配置正确，网络连通、访问权限合理：

- **AWS 相关**：
  - 确认 Amazon OpenSearch Service 的集群版本、索引设置、数据量及查询模式
  - 检查集群快照配置及访问权限，确保可以导出数据
  - 确保网络带宽和安全策略配置满足数据导出需求
- **Azure 相关**：
  - 如果采用 Azure Cognitive Search，请确认服务实例的创建、分区与查询配置，并核实所需的资源配额
  - 如果采用 Azure 上的 OpenSearch 镜像部署，请确认目标部署方案（例如使用 Azure Marketplace 镜像、AKS 或虚拟机）以及网络、安全组和访问控制设置
  - 配置 Azure 存储（如 Blob 存储）用于存放快照数据时，需检查访问策略和带宽配置

OpenSearch (Amazon OpenSearch Service) 迁移方案

本方案介绍两种将 Amazon OpenSearch Service 数据迁移到 Azure 的策略：
a. 采用 **Azure Cognitive Search** 作为托管搜索解决方案
b. 通过 **Azure 上的 OpenSearch 镜像部署** 快速构建开源 OpenSearch 集群

迁移整体流程概览

- **前期准备**
  - 评估现有 OpenSearch 集群的数据规模、索引结构和查询模式
  - 确定迁移目标（Azure Cognitive Search 或 Azure 上的 OpenSearch 镜像部署）
  - 配置网络互通（如需要 VPN 或内网互联），准备数据导出和重新索引策略
- **迁移实施**
  - 导出现有数据和索引配置
  - 在目标平台中重建索引并导入数据
  - 更新应用的查询逻辑和终端配置，指向新搜索服务
  - 进行充分测试以验证搜索结果及性能
- **后续维护与监控**

- 配置定期索引更新和数据同步机制
- 设置监控和日志审计，确保搜索服务稳定运行及数据安全

---

1) 迁移到 Azure Cognitive Search（Azure AI Search）

前期准备

- 分析 OpenSearch 中的索引设置（包括分词器、同义词、过滤规则等）
- 确认待迁移的数据和元数据存储位置，评估数据导出方式（如使用 Scroll API 导出 JSON 数据）
- 创建 Azure Cognitive Search 服务实例，设定所需的配额和分区方案

迁移步骤

a. **数据导出与转换**

- 从 Amazon OpenSearch 导出所有索引数据，并转换为符合 Azure Cognitive Search 要求的格式（包括字段映射和文本预处理）

b. **建立 Azure 索引**

- 在 Azure Cognitive Search 中创建索引，定义字段、数据类型以及排序和过滤规则
- 可以通过 Azure 门户或 REST API 实现

c. **数据导入与索引构建**

- 利用 Azure Cognitive Search 的 Indexer（适用于 Azure 存储中的数据）或 Push API 将数据导入
- 验证索引构建情况与数据加载状态

d. **更新应用与验证测试**

- 更新应用配置，将搜索请求指向 Azure Cognitive Search 服务终端
- 测试搜索准确性、响应速度及查询功能

后续维护

- 配置自动索引更新机制
- 使用 Azure Monitor 监控查询性能和错误日志，定期优化索引设置

---

2) 迁移到 Azure 上的 OpenSearch 镜像部署

前期准备

- 回顾原有 OpenSearch 集群的版本、索引配置和数据量
- 制定在 Azure 部署 OpenSearch 集群的镜像方案，利用 Azure Marketplace 提供的预构建镜像或 Docker 镜像进行快速部署
- 规划网络设置，确保新集群与数据源和应用系统互联（使用 VPN 或内网互联）

迁移步骤

a. **部署 OpenSearch 镜像**

- 从 Azure Marketplace 或使用预构建的容器镜像，在 Azure 上快速部署 OpenSearch 集群
- 根据业务负载选择适当的虚拟机大小或 AKS 配置，参考 Azure 部署最佳实践

b. **数据迁移与索引恢复**

- 在 AWS OpenSearch 中使用 Snapshot/Restore 功能创建索引快照
  - 配置快照仓库，生成快照数据
  - 使用 AzCopy 将快照导出并上传至 Azure 存储（例如 Azure Blob 存储）
- 在新部署的 OpenSearch 集群中配置快照仓库，并恢复索引数据
- 如有需要，利用 Reindex API 重建索引以适应新集群配置

c. **更新应用与验证测试**

- 修改应用配置，将查询请求重定向至 Azure 上的 OpenSearch 集群

- 验证查询准确性、聚合结果和整体性能

后续维护

- 制定定期快照和备份策略，防止数据丢失
- 监控集群性能（CPU、内存、查询延迟等），根据负载调整集群规模
- 定期更新 OpenSearch 版本，确保安全和性能改进

微软迁移参考链接

- Azure Cognitive Search 官方文档
- Azure Cognitive Search 迁移指南
- 在 Azure 部署 Open Distro for Elasticsearch（OpenSearch 类似方案）

注：本方案提供两种迁移策略，请根据实际业务需求、数据规模及查询性能要求选择最合适的方案。迁移过程中请确保充分测试和验证，以保障搜索服务的稳定性与数据准确性。

11. S3 storage

S3 到 Azure 存储迁移手册

迁移前确认云服务内容
在开始迁移前，请务必确认以下云服务相关配置与内容，确保迁移过程中各项参数和访问权限正确：

- **AWS 相关**：
  - S3 存储桶的区域、访问策略、生命周期管理、数据加密及备份设置
  - 数据访问权限配置（IAM 角色、访问密钥等），确保迁移工具具备足够权限
  - 网络配置及带宽情况，确认是否需要通过 VPN 或内网互联保持稳定连接
- **Azure 相关**：
  - 目标存储账户（Azure Blob 存储或 ADLS Gen2）的创建与配置，包括存储类型、区域、访问策略及加密设置
  - Azure 虚拟网络（VNet）、防火墙和子网设置，确保能够安全、高效地与 AWS 对接
  - 相关管理权限与成本估算，确保迁移后资源可持续运营

迁移工具清单：

- **AzCopy**：适用于对象数据的高速传输，常用于从 S3 迁移到 Azure Blob 存储或 ADLS Gen2。
  使用手册：AzCopy 官方文档
- **Azure Data Factory (ADF)**：通过构建数据管道，实现 S3 到 Azure 存储（包括 ADLS Gen2）的自动化、可调度的数据迁移。
  使用手册：Azure Data Factory 概览
- **DISP**：适用于大数据 Hadoop 场景的数据迁移工具，能够满足海量数据处理和高速传输需求。
  开源位置：DISP GitHub Repository
  使用手册：DISP 使用手册

迁移整体流程概览

- **工具与方案选择**

  - 迁移到 Azure Blob 存储：主要使用 AzCopy 工具
  - 迁移到 Azure Data Lake 存储：可采用 Azure Data Factory 管道进行数据迁移，或借助 AzCopy 同步到 ADLS Gen2
  - 对于大数据 Hadoop 场景，则建议采用 DISP 工具以满足海量数据传输需求

- **数据迁移实施**

- 配置网络及身份认证，确保 S3 与 Azure 目标可互访（必要时通过 VPN 或内网互联）
- 运行数据迁移任务，监控迁移进度与错误日志
- 验证数据一致性，执行数据校验

- **后续维护**

  - 配置自动备份、日志审计
  - 优化访问性能、调整容量及监控指标

---

1) S3 到 Azure Blob 存储迁移

1.1) 迁移步骤

a. **初始化迁移环境**

- 在迁移服务器上配置好 AzCopy 工具，并确认能够通过 AWS CLI 访问 S3 数据。
- 创建目标 Azure 存储账户和相应的 Blob 容器。

b. **数据传输**

- 使用 AzCopy 命令从 S3 读取数据并上传至 Azure Blob 存储。示例命令：
  `AzCopy copy "https://s3.amazonaws.com/your-s3-bucket" "https://<your_storage_account>.blob.core.windows.net/<your_container>" --recursive`
- 根据数据量和网络情况，适当调整并发数和重试策略。

c. **验证与校验**

- 通过比对对象数量、MD5 校验和或文件大小，确保数据一致性。
- 检查 AzCopy 日志，确保迁移过程中无关键错误。

1.2) 性能调优与后续维护

- **优化策略**：
  - 根据 Azure 存储的吞吐量和并发限制，调整 AzCopy 的参数（如并发数和分片大小）。
  - 定期监控目标存储账户的请求数、延时和错误率，必要时调整网络设置。
- **后续维护**：
  - 配置自动备份和访问日志记录，并定期进行数据完整性检查。

---

2) S3 到 Azure Data Lake 存储迁移

2.1) 前期准备

- **数据评估**：
  了解 S3 中的数据格式、目录结构及权限设置，评估数据总量和更新频率。
- **环境规划**：
  - 在 Azure 中创建 ADLS Gen2 存储账户，并设置合适的目录结构。
  - 配置网络接入，必要时设置 VPN 或内网互联，确保数据传输通路安全稳定。
- **工具准备**：
  - 可采用 Azure Data Factory（ADF）创建数据管道，实现定时或触发式的自动迁移。
  - 或选择使用 AzCopy 进行数据同步，依据数据量和自动化需求决定方案。

2.2) 迁移步骤

a. **ADF 方案**

- 在 Azure Data Factory 中建立 S3 到 ADLS Gen2 的数据复制管道，配置源（S3 存储桶）和目标（ADLS Gen2 容器）的连接。
- 配置管道参数，如并发复制数量、错误重试机制，确保数据及目录结构保持一致。

b. **AzCopy 方案**

- 与 Blob 存储迁移类似，使用 AzCopy 将数据从 S3 传输到 ADLS Gen2。示例命令：
  ```
  AzCopy copy "https://s3.amazonaws.com/your-s3-bucket"
  "https://<your_adls_account>.dfs.core.windows.net/<your_container>" --recursive
  ```
- 设置目标路径以保持 ADLS 中的目录结构与 S3 一致。

c. **数据验证**

- 验证对象和目录结构的完整性，通过文件大小、校验和检查数据一致性。
- 对 ADF 管道或 AzCopy 日志进行监控，确保迁移过程中无数据丢失或权限问题。

2.3) 性能调优与后续维护

- **优化策略**：
  - 对于 ADF 方案：根据并发量和复制速率调整管道配置，监控活动运行状态并调整重试次数。
  - 对于 AzCopy 方案：调整并发参数和超时设置，确保传输高效稳定。
- **后续维护**：
  - 建议配置定期数据同步或增量更新机制，确保 ADLS 中的数据与 S3 保持一致（可利用 ADF 定时任务）。
  - 定期监控数据传输日志、权限设置和存储使用情况，及时调整配置。

注：本手册为 S3 到 Azure 存储迁移的基础指导，实际实施时请根据具体业务场景、数据规模与系统架构制定详细方案，并充分进行测试和验证，确保迁移平稳安全。

12. WAF (Web Application Firewall)

1) 需求分析与现状梳理

```
- 梳理 AWS WAF 现有规则集：包括 IP 白名单 / 黑名单、自定义规则（URI/Header 匹配、正则表达式规则）、托管规则组（如 AWS Managed
- 记录业务流量特征：如流量入口（API Gateway、ALB 等）、后端服务架构（EC2 实例、容器化应用、Serverless 服务）、地域分布及流量峰值
- 明确迁移目标：确定使用 Azure WAF 部署形态（Application Gateway WAF 或 Front Door WAF），前者适用于区域内流量防护，后者支持
```

2) AWS、AZURE核心功能对比

功能维度	AWS WAF	Azure WAF	核心差异
规则类型	自定义规则、 速率限制、 托管规则组	自定义规则、速率限制、 OWASP CRS 托管规则	Azure 托管规则基于 OWASP 标准，AWS 为自研规则，需手动映射等效功能。
正则引擎	PCRE（Perl 兼容）	.NET 正则 （语法略有差异，如 \d 等效 [0-9]）	需验证正则兼容性，避免迁移后匹配失效。
速率限制	按秒 / 自定义键（如 URI）限制	按分钟 / 客户端 IP 限制	阈值单位不同（秒 vs 分钟），Azure 仅支持 IP 或请求头限制。

功能维度	AWS WAF	Azure WAF	核心差异
托管规则	AWS Managed Rules（自研）	OWASP CRS 3.1/3.2（社区标准）	Azure 覆盖更广泛行业规则，需补充 AWS 特有业务逻辑规则。
部署模式	关联 AWS 服务（ALB、CloudFront 等）	Application Gateway（区域）/Front Door（全局）	Azure 提供 “区域 + 全局” 防护，Front Door 支持边缘节点与 DDoS 防护。
日志集成	CloudWatch Logs（需手动配置）	Azure Monitor/Log Analytics（原生集成）	Azure 日志分析更便捷，支持实时查询与警报。
优先级逻辑	规则组嵌套优先级	平铺规则优先级（不支持嵌套）	Azure 需手动平铺规则顺序，确保与 AWS 执行逻辑一致。
地域支持	按区域独立部署	Application Gateway（区域）/Front Door（全局）	全局防护场景下，Azure 无需跨区域重复配置，管理更高效。

3) 准备工作

3.1) 技术准备

- - **账户权限：**确保拥有 AWS 账户的 WAF 只读权限 及 Azure 账户的 网络参与者 + WAF 策略管理员 权限。
  - **规则清单：**整理 AWS WAF 规则文档，标注规则类型、匹配条件、优先级及依赖关系。
  - **测试环境：**搭建与生产环境隔离的测试环境，包含完整的前端（WAF）、中端（负载均衡）、后端服务链路。

3.2) 资源规划

- - **Azure 资源选型：**根据流量规模选择 Application Gateway 规格（如 v2 SKU 支持 WAF_v2）或 Front Door 标准 / 高级层。
  - **网络架构：**规划 VNet 布局、子网划分、peering 连接（混合架构场景），确保 WAF 可访问后端服务。

3.3) 人员与流程

- - **团队分工：**明确迁移负责人（规则转换、网络配置、测试验证），制定回退方案（如流量切换失败时恢复 DNS 配置）。
  - **时间窗口：**选择业务低峰期执行流量切换，预留充足时间处理突发问题。

4) 规则配置迁移

a. **导出AWS WAF规则**
- 使用 AWS CLI 或控制台导出规则集，保存为 JSON 格式，包含规则、规则组、优先级及匹配条件（如 ByteMatch、RegexMatch、RateBasedRule）。
- 标注规则依赖关系：识别规则间的逻辑关系（如 “与”“或” 条件）、正则表达式语法（AWS 支持 PCRE，Azure 支持 .NET 正则，需调整语法差异，如 \d 替换为 [0-9]）。
b. **规则转换与Azure策略创建**

自定义规则转换

- - AWS ByteMatchStatement → Azure 匹配条件：将请求 URI、Header、查询参数等匹配条件映射至 Azure WAF 的 “请求 URI”“请求头”“查询字符串” 等字段。
  - AWS RateBasedRule → Azure 速率限制规则：在 Azure WAF 策略中配置 “速率限制” 规则，设定每分钟请求阈值及受保护的客户端 IP 范围。
  - 正则表达式调整：使用在线工具（如 RegExr）验证正则表达式在 .NET 引擎下的兼容性，修复语法错误（如 AWS 中的 \\ 需转为 Azure 中的 \）。

托管规则适配

- - 对比 AWS Managed Rules 与 Azure OWASP CRS 托管规则（3.1/3.2 版本），记录缺失的规则条目（如特定 CRS 规则 ID），通过自定义规则补充。
    - 在 Azure 策略中启用 OWASP CRS 规则，并配置规则 exclusion（如允许内部 IP 绕过部分规则）。

批量导入规则

- - 通过 ARM 模板 或 Azure CLI 脚本 批量创建规则，示例如下：

az network application-gateway waf-policy create \

--resource-group my-resource-group \ --name my-waf-policy \ --type ManagedRuleSet \

--managed-rule-set-type OWASP \

--managed-rule-set-version 3.2

a. **规则优先级映射**
- 按 AWS WAF 规则执行顺序（优先级数字越小越先执行），在 Azure WAF 中设置相同优先级，确保逻辑顺序一致（如先执行白名单，再执行速率限制规则）。

5) 网络架构适配

a. **后端服务迁移与连通性配置**
- 全量迁移至 Azure：使用 Azure Migrate 或 Site Recovery 迁移 EC2 实例至 Azure VM/VMSS，或容器化部署至 AKS，确保后端服务部署在与 Application Gateway 同 VNet 或通过 VNet Peering 连通。
- 混合架构配置：若保留 AWS 后端，通过 ExpressRoute 或 VPN 建立 Azure 与 AWS VPC 的私网连接，配置 Application Gateway 或 Front Door 指向 AWS 负载均衡器公网 IP，同时通过 NSG 限制流量仅来自 Azure WAF。
b. **流量路由切换**
- 灰度切换：通过 DNS 分阶段切换（如先将 20% 流量通过 CNAME 指向 Azure Front Door，逐步提升比例），使用 Azure Traffic Manager 或 DNS 解析服务商（如 Cloudflare）实现流量分流。
- 健康检查配置：在 Front Door 或 Application Gateway 中设置后端健康探测，确保流量仅转发至正常运行的后端实例。
c. **验证与优化**
d. **功能验证**
- 规则匹配测试：使用 Postman 或 curl 模拟攻击请求（如 SQL 注入、XSS），验证 Azure WAF 是否按预期拦截，对比 AWS 日志与 Azure 日志的事件一致性。
- 例外场景测试：验证白名单 IP、允许的 URI 路径（如健康检查接口）是否正常通行，无误拦截。
e. **性能与流量验证**
- 吞吐量测试：通过工具（如 JMeter）模拟峰值流量，监控 Application Gateway/Front Door 的 CPU / 内存使用率、延迟及请求处理能力，确保满足业务需求。
- 日志分析：启用 Azure Monitor 及 Log Analytics，对比迁移前后的拦截事件数量、误报率，分析规则覆盖率是否达标（如 AWS 规则 100% 映射至 Azure）。
f. **安全合规验证**
- 确认 Azure WAF 策略满足业务合规要求（如 PCI-DSS、GDPR），补充缺失的合规相关规则（如敏感数据泄露检测）。

6) 核心部署工具

工具 / 服务	*用途*	*优势*
AWS CLI	导出 AWS WAF 规则集	支持批量导出，兼容脚本自动化
Azure CLI / PowerShell	创建 Azure WAF 策略、规则批量导入	支持脚本化部署，减少人工错误
ARM 模板	基础设施即代码（IaC）部署	支持版本控制，适合大规模环境复制

工具 / 服务	*用途*	*优势*
Azure Portal	可视化配置 WAF 策略、监控日志	适合快速验证与小规模规则调整
Azure Migrate	后端 EC2 实例迁移	自动化评估迁移可行性，简化迁移流程
Log Analytics	日志分析与威胁检测	支持自定义查询，实时监控 WAF 拦截事件
Postman / JMeter	功能与性能测试	模拟真实业务流量，验证规则有效性

7) 验证清单

7.1) 规则配置验证

- - 所有 AWS 自定义规则已转换为 Azure 支持的格式（匹配条件、正则表达式语法正确）。
  - 托管规则组（OWASP CRS）版本与 AWS 功能对齐，缺失规则已通过自定义规则补充。
  - 规则优先级顺序与 AWS 一致，无逻辑冲突（如白名单规则优先级高于拦截规则）。

7.2) 网络连通性验证

- - 后端服务（Azure 或混合架构）与 WAF 实例网络可达（通过 telnet/HTTP 测试端口连通性）。
  - 流量切换后，DNS 解析正确指向 Azure 入口（Front Door/Application Gateway）。

7.3) 功能与安全验证

- - 正常业务流量无中断，白名单 IP/URI 可正常访问。
  - 攻击模拟请求被正确拦截（如访问 example.com/?id=1' OR 1=1-- 应触发 SQL 注入拦截）。
  - 速率限制规则生效：短时间内高频访问同一端点时，超出阈值的请求被拦截。

7.4) 日志与监控验证

- - Azure 日志中记录的拦截事件与预期一致，无大量误报或漏报。
  - 监控指标（如请求速率、延迟、错误率）在正常范围内，无异常波动。

8 参考文档

8.1) 官方文档

- ***AWS WAF CLI命令参考***：AWS WAF CLI 命令参考：

https://docs.aws.amazon.com/cli/latest/reference/wafv2/

- ***Azure WAF 部署文档***：
  - Application Gateway WAF 配置：

https://learn.microsoft.com/en-us/azure/application-gateway/waf-overview

- - Front Door WAF 规则创建：

https://learn.microsoft.com/en-us/azure/frontdoor/afront-door-waf-overview

- - OWASP CRS 规则说明：

https://www.owasp.org/index.php/Category:OWASP_Charset_Revelation_Cheat_Sheet

- ***混合网络连接***：Azure 与 AWS 混合连接最佳实践：

https://learn.microsoft.com/en-us/azure/architecture/hybrid-guide/connectivity

8.2) 最佳实践

- 规则转换案例库：包含常见 AWS 规则到 Azure 的转换脚本。

https://github.com/Azure-Samples/application-gateway-waf-policy-examples

- 性能优化指南：针对高流量场景的 WAF 配置建议。

https://learn.microsoft.com/en-us/azure/application-gateway/performance-tuning

9) 回退策略

a. 备份 AWS WAF 原始规则集及 Azure 迁移过程中生成的 ARM 模板，确保可快速恢复至迁移前状态。
b. 流量切换前，记录 DNS 配置快照，若验证失败，通过 DNS 服务商快速回切流量至 AWS 入口。
c. 保留 AWS WAF 实例运行至少 72 小时，期间对比双方日志，确认无关键规则遗漏后再下线。

13. Key Management (AWS KMS)(`Done`)

AWS KMS 到 Azure Key Vault 密钥迁移方案（密钥对换）

本方案通过密钥对换方式，将 AWS KMS 中的密钥迁移到 Azure Key Vault。由于 AWS KMS 的密钥通常不可导出，因此主要通过在 Azure Key Vault 中重新创建新密钥并更新应用配置来实现切换。下面给出重点步骤及部分实作细节：

迁移前确认

- **AWS 部分：**
  - 清点当前所有使用中的 AWS KMS 密钥及其用途。
  - 检查各密钥的访问控制策略（IAM 角色、策略）及日志记录配置。
- **Azure 部分：**
  - 创建 Azure Key Vault，并配置基本的访问策略。
  - 确认网络与身份验证设置（如 VPN 或内网连接）正常，保证应用能够安全调用 Key Vault。

迁移步骤

- **密钥对换：**
  - **在 Azure Key Vault 中创建新密钥**
    使用 Azure CLI 命令创建密钥。例如，要创建一个对称密钥：

```
az keyvault key create --vault-name <YourKeyVaultName> --name <NewKeyName> --protection software
```

    或者如需创建非对称密钥：

```
az keyvault key create --vault-name <YourKeyVaultName> --name <NewKeyName> --kty RSA --size 2048
```

    注意：确保新密钥的算法、用途与旧密钥相匹配。

  - **（可选）使用 BYOK 导入密钥**
    若有外部生成且允许导出的密钥，可以采用 BYOK 机制。需先导出密钥文件，再使用 Azure CLI 进行导入：

```
az keyvault key import --vault-name <YourKeyVaultName> --name <ImportedKeyName> --file <PathToKeyFile>
```

- **更新应用配置：**
  - 修改应用程序中原先调用 AWS KMS 的部分配置，将其更新为调用 Azure Key Vault 的接口和新密钥标识。
  - 例如，在 Python 中使用 Azure SDK 访问 Key Vault：

```
from azure.keyvault.keys import KeyClient
from azure.identity import DefaultAzureCredential

credential = DefaultAzureCredential()
key_vault_url = "https://<YourKeyVaultName>.vault.azure.net/"
client = KeyClient(vault_url=key_vault_url, credential=credential)

new_key = client.get_key("<NewKeyName>")
# 更新应用中使用的加密/解密逻辑，确保调用 new_key 对象
```

- **测试验证：**
  - 在预生产或测试环境中验证数据加解密、数字签名及身份验证流程是否正常工作。
  - 检查 Azure Key Vault 的监控日志，确认调用没有异常错误。
  - 对比原有 AWS KMS 与新系统的功能输出，确保数据处理和安全性符合预期。

后续维护与监控

- **密钥轮换**
  - 配置自动或手动密钥轮换策略，确保密钥符合最新安全要求。
- **备份与恢复**
  - 利用 Azure Key Vault 的备份功能，定期备份密钥数据：

```
bash az keyvault key backup --vault-name <YourKeyVaultName> --name <NewKeyName> --file <BackupFilePath>
```

- **日志审计与安全监控**
  - 启用并配置 Azure Key Vault 审计日志，记录密钥调用及操作行为，以便安全监控和合规检查。
- **应用更新**
  - 定期更新应用配置和 SDK 版本，确保与 Azure Key Vault 的接口保持兼容和安全。

微软迁移参考链接

- Azure Key Vault 官方文档
- Azure Key Vault 迁移指南

注：本方案基于密钥对换方式实现 AWS KMS 到 Azure Key Vault 的迁移，重点在于重新创建密钥和更新应用调用。请在迁移前后充分测试与验证，确保数据安全和业务连续。