Recommend an Azure Data Solution Based on Requirements

Tuesday, 11 August 2020 12:27 am

Introduction to Azure Storage

Big data storage technologies

- Azure Storage blobs
- Azure Data Lake Store
- Azure Cosmos DB
- HBase on HDInsight

Key Selection criteria

- size of dataset
- performance requirements (Horizontal or Vertical scaling)
- data types to be ingested
- where will data be consumed
- How will data be accessed,
- Data redundancy and availability

Big data Storage Technology

SQL Database

Purpose Transaction processing
Data Capture Single source
Benefits Consistent data that can handle complex queries
Data Quality Organized and consistent

SQL Data Warehouse

Purpose Multiple source reporting and queries
Data Capture Multiple relational sources
Benefits Complex data that can handle complex queries through MPP
Data quality Organized and consistent

Data Lake Storage

Purpose Dynamic big data storage at any speed
Data capture Any number of sources
Benefits Different data types that can change at will
Data quality completely dependent upon Rules of Engagement (ROE)

- Collection of objects and files organized into a hierarchy of directories and subdirectories
- File systems that can be used by anyone

Cosmos DB

Purpose Global distribution

Cosmos DB
Purpose — Global distribution
Data capture — Multiple geo-replicated sources
Benefits — Multi-modal, elastic, global-replication
Data quality — organized and consistent

Cosmos DB is non-relational database

Global distribution and speed ⇒ Cosmos DB
Consistency and integrity ⇒ SQL DB / data warehouse

Choosing the right data store

General considerations
- Functional requirements
- Non-functional requirements
- Management and costs
- Security
- DevOps

DP-201 scope
- RDBMS
- Key/value stores
- Document database
- Column-family databases
- Graph databases

- SQL database
- SQL data warehouse
- Cosmos DB
  * SQL API
  * Gremlin API
  * Mongo DB API
  * Cassandra API
  * Table API

RDBMS

Two dimensional tables with rows and columns

structured schema on write
uses SQL
Incorporates ACID model

Use cases:
- strong consistency guarantees are important
- schema provides opportunity for complex queries

Azure applications:

Azure applications:
Azure SQL database
Azure SQL data warehouse

Key-value stores:

- Simple databases that uses an associative array.
  ↳ key is associated with one value in collection.
- Query, insert and delete operations

Typical use cases
- user profiles
- Product recommendations
- Session management

Azure applications:
- Cosmos DB
- Redis cache

Document Database
delux key-value store
documents can be encoded in XML, JSON, YAML etc.
schema provides opportunity for complex queries
free form storage system

Azure Applications
- Cosmos SQL API
- Cosmos MongoDB API

Column family databases:

- Similar to RDBMS. Each table has rows and colums.

Groups related columns into families
Duplicates are possible
Joins are out

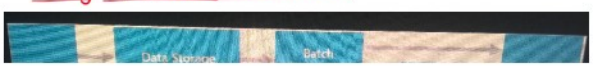
Azure Applications:
- HBase in HDInsight
- Cosmos Cassandra API

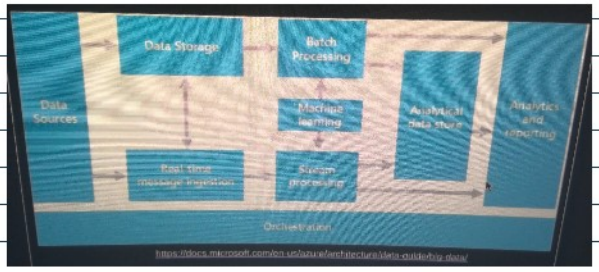
Graph databases:

Nodes = Entities
Edges = Relationships

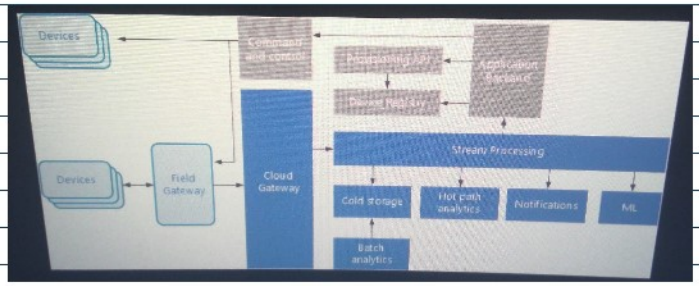
Azure applications:
Cosmos Gremlin API

Big data Architecture





Machine Learning
Analytical data store
↳ HBase
↳ SQL Data Warehouse



⇐ IoT Architecture

Capability Matrix :

File storage capabilities

Capability	Azure Data Lake Store	Azure Blob Storage containers
Purpose	Optimized storage for big data analytics workloads	General purpose object store for a wide variety of storage scenario
Use cases	Batch, streaming analytics, and machine learning data such as log files, IoT data, click streams, large datasets	Any type of text or binary data, such as application back end, backup data, Media, storage for streaming, and general purpose data
Structure	Hierarchical file system	Object store with flat namespace
Authentication	Based on Azure Active Directory Identities	Based on shared secrets Account Access Keys and Shared Access Signature Keys, and role-based access control (RBAC)
Authentication protocol	OAuth 2.0 Calls must contain a valid JWT token issued by Azure Active Directory	For account level authorization use Account Access Keys. For account container, or blob authorization use shared Access Signature Keys
Auditing	Available	Available
Encryption at rest	Transparent, server side	Transparent, server side, client side encryption.
Developer SDKs	.NET, Java, Python, Node.js	.NET, Java, Python, Node.js, C++, Ruby
Analytics Workload Performance	Optimized performance for parallel analytics workloads, High Throughput and IOPS	Not optimized for analytics workloads
Size limits	No limits on account sizes, file sizes or number of files	Specific limits exists

Geo-redundancy	Locally redundant (LRS), globally redundant (GRS), read-access globally redundant (RA-GRS), zone-redundant (ZRS)	LRS GRS ZRS RA-GRS

NoSQL database capabilities:

Capability	Azure Cosmos DB	HBase on HDInsight
Primary database model	Document store, graph, key-value store, wide column store	Wide column store
Secondary Indexes	Yes	No
SQL language support	Yes	Yes (using Phoenix JDBC driver)
Consistency	Strong, bounded staleness, session, consistent prefix, eventual	Strong
Native Azure Functions integrations	Yes	No
Automatic global distribution	Yes	No, HBase cluster replication can be configured across regions with eventual consistency.
Pricing model	Elastically scalable request units (RUs) charged per-second as needed, elastically scalable storage	Per-minute pricing for HDInsight cluster (horizontal scaling of nodes), storage

Designing Nonrelational cloud stores

Wednesday, 12 August 2020 12:53 am

Data partitioning in Azure

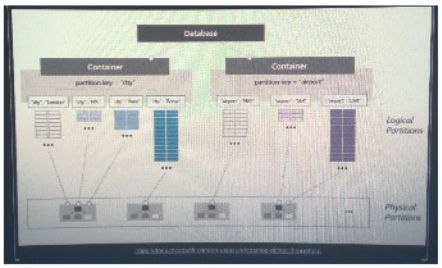
Partitioning in Cosmos DB

used to scale individual containers to achieve performance

- Logical partitions are based on partition keys
- Physical partitions have components that can vary

Logical partitions

- Automatically formed based on partition keys
- Item index is a combination of PartitionKey and item ID



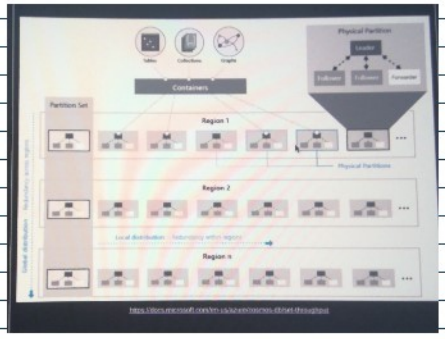
Logical partitions – building a Partition key

Even distribution

- Partition key is property that will exist on every object
- Every document with the same PartitionKey belongs to the same logical partition
- PartitionKey decides the placement of documents.

Building a great PartitionKey

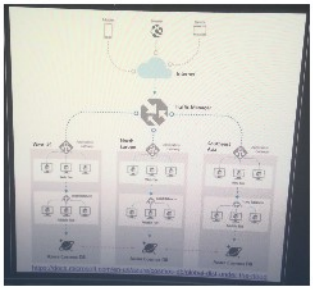
- Anticipate top queries
- Avoid fans
- Choose well balanced keys
- Everything is subject to change
- Keys are immutable, so choose wisely
- 10 GB data maximum



Design for Global Distribution in Azure

Importance of global distribution in Cosmos DB

- Unlimited elastic write and read scalability
- 99.999 % read and write availability worldwide
- Guaranteed read and write < 10 milliseconds at the 99th percentile



If a region goes down, Cosmos DB Traffic Manager automatically serves traffic from other region.

$ Cosmos DB Pricing $

Per container → Throughput

- Single region write: T * N
- Multi region writes: T * (N+1) RU/s

Storage
- $.25 / GB per month / per region
- No max storage limits

T → Throughput
N → # regions

Consistency in Cosmos DB

1. **Strong Consistency**
   - Highest consistency
   - lowest availability
   - High cost
2. **Bounded Staleness**
   - lag between updates (eg: scoreboard updated frequently without interupting the game)
3. **Session consistency**
   - Default for Cosmos DB

   Game A Game B games

   Consistency across single game
   Lag for the other game
   - same session consistency
   - Cross session lag
4. **Consistent prefix**

   Game A and Game B both are updated parallely. Final scores are eventually consistent.
5. **Eventual Consistency**
   - weakest consistency
   - lowest resources

Cosmos DB API

Chosen at account creation

Cannot be modified later

API

1. **Core (SQL) API**
   - default
   - "SQL like"
   - JavaScript type, expression evaluation, and function invocation
   - provides document data model
2. **Mongo DB API**
   - supports MongoDB Wire protocol
   - full compatibility with existing MongoDB app
   - Built in find method
3. **Cassandra API**
   - Columnar data model
   - can define data schema upfront
   - Cassandra Query Language (CQL)
4. **Azure Table API**
   - Great for table storage with advanced requirements
   - No index management required
   - Key-value data model
   - New projects, shift to SQL API
5. **Gremlin API**

- Columnar data model
- can define data schema upfront
- Cassandra Query Language (CQL)

New projects, shift to SQL API

5. Gremlin API
   - Graph data model
   - Edges & nodes or vertex

DISASTER RECOVERY & HIGH AVAILABILITY

High Availability

SLA
- multiple regions: at least N x 4 copies
- single region: 99.99 SLA
- multi region
  - single writes: 99.99 W, 99.999 R
  - multiple writes: 99.999

Need to enable automatic failover to avoid conflicts when the region comes back online due to unreplicated data

Disaster Recovery

Online backup & data restore

- Auto backup every 4 hours → Only 2 latest snapshots are available for 30 days
- Blob storage
- Azure Cosmos DB for data

No additional costs for automated backups

Additional Backups

- Auto geo-redundant storage
- Configure multi-region

- ADF or Cosmos DB changed feeds can be used for custom backup needs

Data Lake Gen 2

Best of Gen 1 + Blob storage

- Hot, cold and Archive storage (Access Tier)
- Redundant
- HDFS compatibility
- Hierarchical file system
- Elastic, scalable environment with enhanced performance

Design Relational Cloud Data Stores

Thursday, 20 August 2020 11:59 pm

PARTIONING TABLES

Table Partitions

- Smaller groups of data
- Improve maintenance
- Query performance

DATA DISTRIBUTION

HASH Distribution

- Row based on values
- High performance queries large tables

Replicated tables

- Full copy
- Small tables with performance

⚠ Does not require data movement

Round Robin

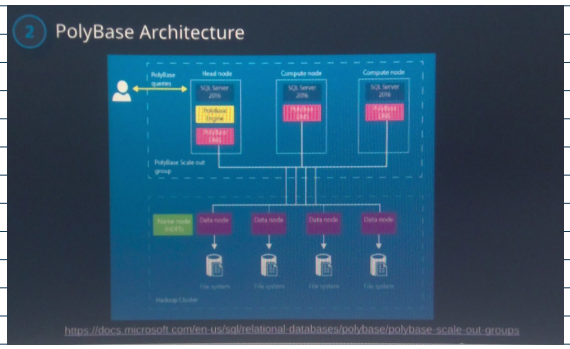
- Rows distributed evenly
- Random distribution
- Faster queries
- Improves load speed

⚠ Used for staging tables

INGESTION WITH POLYBASE

Polybase

- Query across relational and Hadoop data
- Bypass traditional ETL for ELT
  - Perform transformations in DW
  - Distribute transformations across multiple databases



ELT steps

- Extract source data into text files
- Land data into Azure Blob or ADLS
- Prepare data for loading
- Load data into staging with PolyBase
- Transform
- Insert into production tables

⚠ CTAS : optimized for bulk loads

```
CREATE TABLE [NWI].[dimension_City]
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT * FROM [ext].[dimension_City]
```

Design for Scale in SQL DB

1. Scale-Up
   - Vertical scalability
   - CPU, memory, IO throughput and storage
   - DTU & vCore models
   - Dynamic scalability
   - ⚠ Not autoscale
2. Scale-Out
   - Load-balance SQL DB (read-only workload)
   - Default for premium and business critical tiers
   - Available in Hyperscale with secondary replica
3. Elastic pools
   - Unpredictable demands
   - Shared pool resources to manage & scale multiple databases
   - Pricing options
     - DTU
     - vCore

Scale

Scale SQL DW

DW scale or pause

- Scale up during heavy demand
- Pause to cut cost
- Modify with GUI, Powershell or TSQL

Disaster Recovery & High Availability

High Availability

Data Warehouse	SQL DB
⚠ No high availability	Auto-failover through failover groups

Disaster Recovery

Data Warehouse	SQL DB
Snapshots and restoration	Auto-failover through failover groups

Long-term retention strategies

- Allows user defined restore point (upto 42 instances)

- Allows user defined restore point (upto 42 instances)

Default retention period → 7 days

SQL DB Automated Backup

Automated Backups

- RA-GRS storage
- Full weekly / differential / transactional
  - differential: every 12 hrs
  - transactional: every 5-10 minutes

Defaults → DTU
- Basic: 1 week
- Standard & Premium: 5 weeks

vCore: 1 week

Point-in-time Restore (PITR) max 14 days

Long-Term Retention

Leverages PITR - blob for long-term storage

Eg.
- W=0, M=0, Y=2, WeekOfYear=2
- W=0, M=4, Y=0
- W=3, M=12, Y=7, WeekOfYear=10

Disaster Recovery Strategy

Protects organization from unacceptable negative events following a disaster.

2 core elements
- Data Backup
- Business Continuity

Recovery Time Objective (RTO)
⇓
Maximum time before system is brought back online

Recovery Point Objective (RPO)
⇓
Acceptable gap of data lost when restoring backups

Design Batch Processing Solutions

Saturday, 3 October 2020 1:15 pm

Data Ingestion

① Data Factory

- Orchestration service
- ETL / ELT
- Core functions
  - Ingest
  - Data flow
  - Automate
  - Monitor
- Copy function

② Storage Explorer & AZ Copy

- Storage explorer
  - Manual file or folder movement
- Az copy
  - Blob or files from storage account
  - local computers or other storage accounts

③ Apache Sqoop & others

- Apache Sqoop
  - Data movement tool into Hadoop HDFS
  - ETL tool

Data Processing

ELT

- Self service
  - Data Lakes and zones
  - Raw data accessed by multiple thought leaders
- Time
  - Accessibility
  - Maintenance
  - Load and transfer
- Cost and complexity
  - Data quality
  - Data management
  - Cost
    - * Storage
    - * Movement
    - * Transformation
    - * Resources

Processing with Databricks

Terminology

- Cluster
- Workspaces
- Notebooks
- Cells

Key Features

- Interactive workspace for exploration and visualization
- Core API
  - R
  - SQL

- Notebooks
- Cells
- Libraries
- Tables

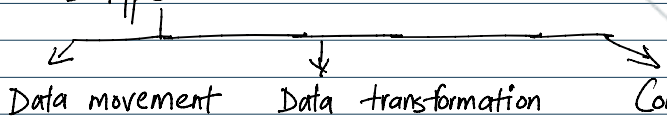
- R
- SQL
- Python
- Scala
- Java

- Streaming and batch processing
- Dynamic scaling and secure integration

Batch Processing using Data Factory

Core concepts

- Pipeline
  - Logical group of activities
- Activity
  - Processing step in pipeline
  - 3 types



- Datasets
  - Data structures within data stores
- Linked services
  - Connections to data

$ Pricing $$$

- Inactive v/s active
- Pipeline orchestration and execution
- Data flow execution and debugging
- Data factory operations (e.g. creating pipelines & monitoring)

Design Real-Time Processing Solutions

Saturday, 3 October 2020 6:20 pm

Stream Processing in Azure

Azure Stream processing use cases

- Never-ending streams of data
- Extreme precision is not needed
- IOT

⚠ Complex big data solutions involve stream and batch processing

Disaster Recovery

- Handle lost data
  - Windowing
  - Early and late arriving events

Input → Stream Analytics ← Query

Stream Analytics → Output

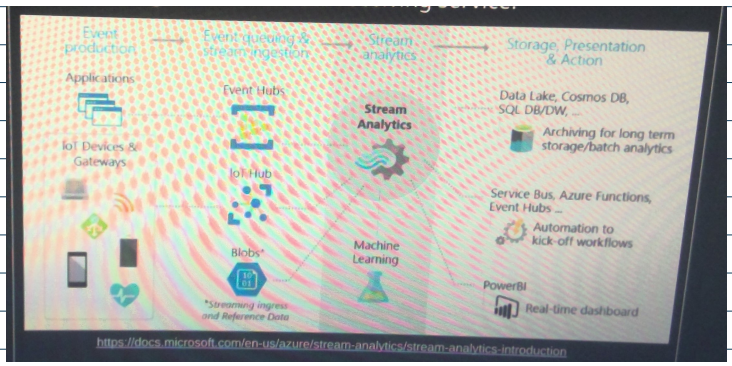
- Defines and connects inputs

Event Hubs
IOT Hub
Blob storage

- Store and save data
- Configure and validate receiving service

Data Lake, Cosmos DB, SQL DB / DW
Service Bus, Azure Functions, Event Hubs
Power BI

- Build and test stream analytics queries
- Estimate pricing



Stream Analytics jobs can be deployed on Cloud or Edge device

Pricing is based on streaming units

$$ Pricing is based on streaming units

Event ordering allows handling of late and out of order events

Streaming in Databricks

Streaming options

- Spark Structured Streaming
- Event Hubs
- IoT Hub

Optimize Stream Analytics

① Streaming Units

- Computing resources allocated to job
- Review SU % → 80 % should be redline
- Start with 6 SUs for queries not using PARTION BY

② Parallelization

- Partitioning key for optimization
  - Inputs are already partitioned
  - Outputs need to be partitioned
- Events should go to same partition of input
- Use PARTITION BY in all steps
- Input partition must equal output partition

Optimize Streaming in Databricks

① Recover query failures

② Spark scheduler pools

① Recover query failures

- Enable checkpointing
- Restart jobs on failures
- Recover after changes

② Spark scheduler pools

- Fair scheduler pools
  - Group jobs into pools by weight

③ Optimize configuration

- Use updated DBR
- Compute optimized instances
- shuffle partitions → 1-2x cluster number

④ Watermark policies

⑤ Auto Scaling

Streaming job config

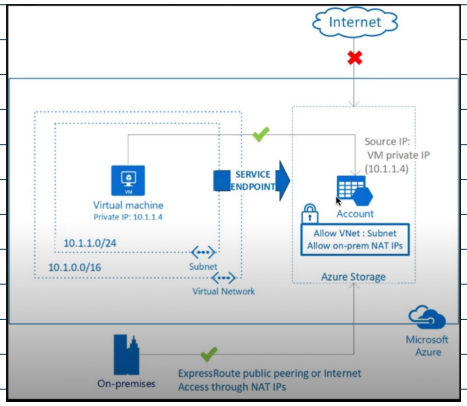
- Max concurrent jobs → 1
- Timeout → None
- Retries → Unlimited

Design security for Source Data Access

Sunday, 4 October 2020 8:19 am

Secure Endpoints

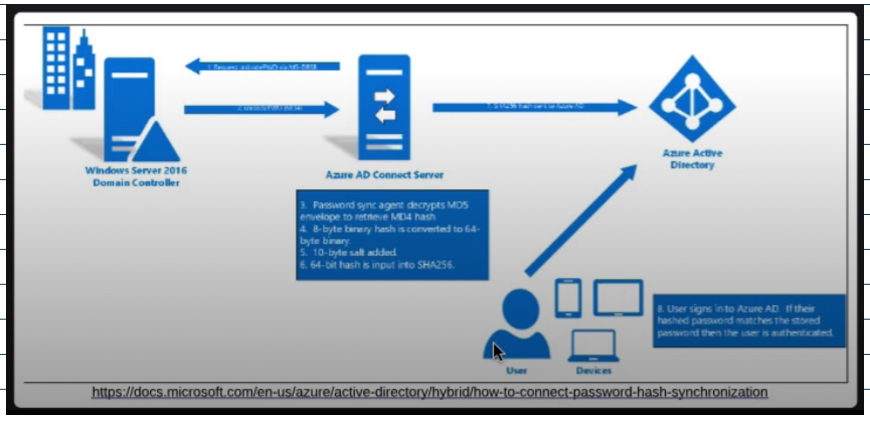
- Virtual Network (VNet) service endpoint
- Network extender to include Azure services



Private Link

- Creates private endpoint for Azure services
- Creates private IP restricting the traffic to Azure backbone

Authenticate with Azure Active Directory

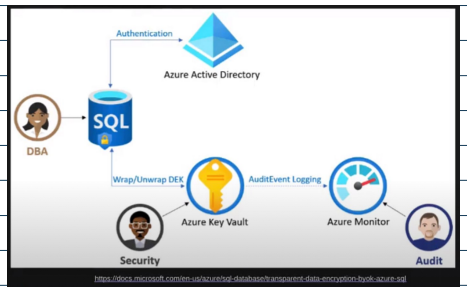


Authenticate with Access keys

Authenticate with Access keys

Azure Key Vault

- Securely store and access
  - Tokens
  - Passwords
  - Certificates
  - API Keys
- Centralize storage of secrets
- Monitor access



Authenticate with Shared Access Signature

Shared Access Signature

- Secure delegated access

SAS types

1. User delegation SAS
   - Blob storage only
2. Service SAS
   - Storage Account Key
   - Azure Storage services only
3. Account SAS
   - Storage Account key
   - Delegates to one or more storage services

- Storage Account key
- Delegates to one or more storage services

Design Security for Data Policies and Standards

Sunday, 4 October 2020 9:53 am

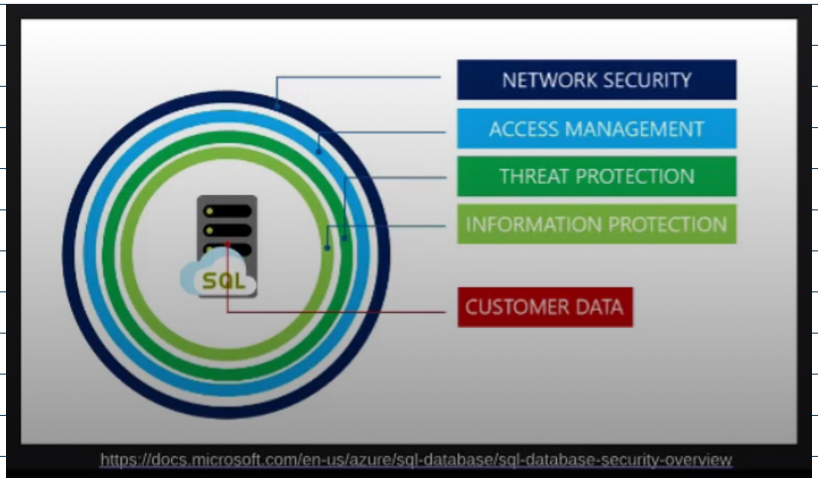
Defense in Depth

CIA

Confidentiality
Integrity
Availability

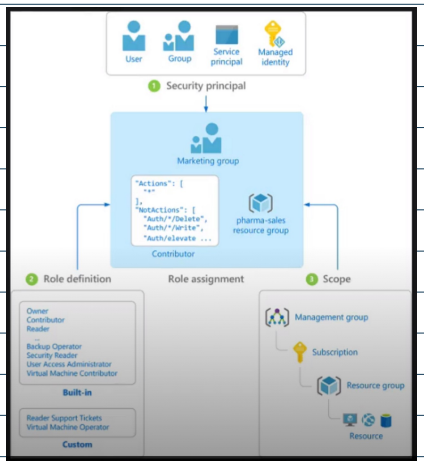
Security Layers

- Physical security – Building and computing hardware
- Identity and access – MFA
- Perimeter – Firewall
- Network – Network connectivity
- Compute – VM & other compute resources
- Application
- Data – Dynamic data masking



RBAC

- Role Assignment
  - Enforces permissions
  - Types
    - Security Principal
    - Role definition
    - Scope



Data Encryption in Transit & at Rest

Available for SaaS, PaaS, IaaS

Transparent Data Encryption

Available for
- SQL Server
- Azure SQL DB
- Azure Synapse

Uses
- AES and Tripple Data Encryption standard
- Data encryption key

Cosmos DB Database Encryption

- Encrypted by default
- Uses secure key storage systems, encrypted networks and cryptographic APIs

encrypted networks and cryptographic APIs

Data Lake Encryption

- Encrypted by default
- MS managed keys, can be BYOK
- 3 keys
  - Master Encryption Key (MEK)
  - Data Encryption Key (DEK)
  - Block Encryption Key (BEK)

Encrypting Data in Motion

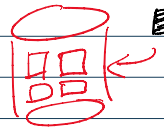
- TLS / SSL
  - Transport Layer Security → Between cloud and customer
  - Strong authentication
  - Message privacy
  - Integrity
  - Perfect Forward Secracy (PFS)
    - ↳ Protects data between customer's client systems and cloud
- Shared Access Signatures
  - Delegate access to Azure Storage object
- Data Lake
  - Data in transit always encrypted
  - HTTPs is the only protocol available for REST interface

SQL Database Auditing

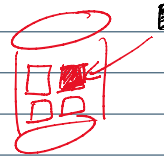
Use cases

- Retain → Audit trails
- Report → Event reporting
- Analyze → Spot trends and unusual activities

Audit policy



- Server-level
  - Applies to all existing and new databases
  - Recommended approach



- Database-level
  - Specific database auditing
  - Automatically enabled for secondary databases when geo-replication is active

Data Masking

Dynamic Data Masking

- Real-time data masking
- Limits sensitive data exposure to non-privileged users by masking
- Types
  - Default
  - Credit card
  - Email
  - Random
  - Custom

Use cases
- Protect non-production data
- Protect against insider threats
- Regulatory requirements

Archiving Strategy

Archiving Strategy

Azure Blob Storage

- Hot
  - Frequently used data
  - Fast and expensive
    - * Low read and write cost
    - * High storage cost
- Cool
  - 30 days
  - short-term backup and DR
    - * 1/2 cost compared to hot
    - * 2x cost to read and write
- Archive
  - 180 days
  - Affordable* if we don't move data
  - Long time to retrieve data
  - Compliance and regulatory

Data Retention Policy

Tape v/s Archive

- Convinience → cloud based v/s physical
- Discovery → GDPR & other regulatory requirements
- Protection → Physical v/s cloud
- Maintenance → Tape curruption, cloud always secure

Cool v/s archive

- Cost
  - Storage
  - Data movement
  - Early deletion fees
- Accessibility

Lifecycle Management

Lifecycle Management

- Rules for automating blob tier movement
- Policies to drive movement of data:
  - from hot to cool
  - from cool to archive
  - delete blobs
  - delete snapshots

Azure Data Security Best Practices

Key Vault

- Safegaurd Keys, certificates and secrets
- Control access points

Protect data at rest

- Automated (ADLS, SQL DB, DW)
- TDE

Data in transit

- Endpoints
- HTTPS
- Azure storage
  - ExpressRoute

Active Directory

- Authorization
- Authentication
- RBAC
  - Security principal
  - Role
  - Definition