AZ-305 - Comprehensive Azure Study Guide

AZ-305: Designing Microsoft Azure Infrastructure Solutions — Comprehensive Study Guide

> Audience: Experienced Azure Infrastructure Architect designing solutions at the expert level. Exam Version: Skills measured as of April 17, 2026

---

1. Design Identity, Governance, and Monitoring Solutions (25-30%)

1.1 Design Solutions for Logging and Monitoring

Logging Solution Design

Decision matrix:

Requirement	Solution
Centralized log aggregation	Log Analytics workspace
Long-term retention (>2 years)	Export to Storage Account (cheap archival)
Real-time streaming	Event Hub (stream to SIEM, Splunk, etc.)
Multi-tenant/organization	Per-tenant workspace + Lighthouse for cross-tenant
Compliance archival	Storage Account with immutable blob policy

Log Analytics workspace design: - **Workspace per:** Organization (centralized), Subscription, Region, Workload - **Centralized:** Easier cross-resource queries, simpler RBAC. Can be a bottleneck at very high ingestion. - **Decentralized:** Data sovereignty compliance, granular access control. Harder cross-workspace queries. - **Hybrid (recommended):** Centralized for security/audit, per-workload for application logs

Data retention: 30-730 days in workspace. Export to Storage Account for longer retention. **Cost:** Per-GB ingested + Per-GB retained (after 31 days). Monitor and set daily caps.

Log Routing Design

Source	Destination	Method
Azure Monitor metrics	Log Analytics, Storage, Event Hub	Diagnostic settings
Activity log	Log Analytics, Storage, Event Hub	Diagnostic settings (legacy: log profile)
Resource logs	Log Analytics, Storage, Event Hub	Diagnostic settings
Entra ID logs	Log Analytics, Storage, Event Hub	Diagnostic settings on Entra ID
VM logs	Log Analytics	Azure Monitor Agent + DCR

Application logs	Application Insights	Auto-instrumentation or SDK

Log routing best practices: - Send security logs to both Log Analytics AND Event Hub (for Sentinel + third-party SIEM) - Archive compliance-required logs to Storage Account with immutable policy - Use DCRs to filter at source (reduce ingestion cost by sending only needed data)

Monitoring Solution Design

Azure Monitor: - **Metrics:** Lightweight, near real-time. Good for dashboards, auto-scaling, fast alerts. - **Logs:** Rich querying (KQL). Good for deep analysis, trend detection, compliance. - **Application Insights:** APM for apps. Request rates, response times, failure rates, dependency maps.

Alert strategy: - **Metric alerts:** Fast (<1 min). For infrastructure health (CPU, memory, disk, response time). - **Log alerts:** Richer queries but slower (1-5 min). For complex patterns (multiple failed logins, anomalous traffic). - **Smart alerts:** AI-powered. Detect anomalies automatically. Lower noise. - **Action groups:** Define once, reuse across alerts. Email, SMS, webhook, Logic App, Function. - **Alert processing rules:** Suppress during maintenance, route to different teams by severity.

Dashboard strategy: - **Executive:** High-level KPIs, SLA compliance, cost trends - **Operations:** Infrastructure health, active alerts, capacity trends - **Security:** Secure Score, threat alerts, compliance status - **Application:** Response time, error rate, throughput, dependency health

1.2 Design Authentication and Authorization Solutions

Authentication Solution Design

Decision framework:

Scenario	Authentication Solution
Internal employees	Entra ID + Conditional Access + MFA
Partners/contractors	Entra ID B2B (guest accounts)
Customers/consumers	Entra ID B2C
Legacy apps (Kerberos/NTLM)	Entra Domain Services
On-prem + cloud	Entra Connect (password hash sync, PHS, or pass-through auth, PTA)
Passwordless	FIDO2 keys + Authenticator + Windows Hello

Entra Connect sync methods:

Method	How It Works	Pros	Cons
Password Hash Sync (PHS)	Hash of password hash synced to cloud	Simple, resilient (works when on-prem is down), supports leaked credential detection	Password hash in cloud (acceptable risk for most)
Pass-Through Auth (PTA)	Auth request forwarded to on-prem agent	No password hash in cloud, real-time on-prem policy evaluation	Requires on-prem agents, single point of failure if not HA
Federated (AD FS)	Full federation with AD FS	Full control, supports smart cards, claims-based apps	Complex, requires AD FS infrastructure, hard to maintain

Best practice: Deploy PHS as primary (always works, even during PTA/federation outages). Use PTA or federation if policy requires on-prem authentication.

Identity Management Solution Design

Entra ID architecture for enterprise: 1. **Single tenant** for most organizations 2. **Multi-tenant** for organizations with strict data isolation requirements (subsidiaries) 3. **B2B** for partner access (avoid creating separate tenants) 4. **B2C** for customer-facing apps

Identity governance: - **Access reviews:** Quarterly reviews of group memberships and role assignments. Reviewers approve/deny. Auto-remove denied access. - **Entitlement management:** Define access packages (groups, apps, SharePoint sites). Users request via portal. Approval workflow. Time-limited access. Auto-expire. - **PIM:** JIT access for privileged roles. Approval, MFA, justification, time limits.

Authorization Design

Azure RBAC design: - Use built-in roles whenever possible. Custom roles only when necessary. - Assign roles at the highest scope that makes sense (MG > Sub > RG > Resource) - Use groups for role assignments (not individual users) - Separate data plane roles (Storage Blob Data *) from management plane roles (Storage Account Contributor)

On-premises authorization: - **AD DS groups:** For on-prem resource access - **Entra ID groups:** For cloud resource access - **Group sync:** Entra Connect syncs on-prem groups to cloud - **Cloud-only groups:** For cloud-only scenarios (dynamic groups)

Secrets, Certificates, and Keys Management

Key Vault design: - **Single vault per application per environment** (separation of concerns, blast radius) - **Or:** Shared vault per environment with access policies per application - **HSM-backed keys:** Premium SKU. Keys never leave the HSM boundary. FIPS 140-2 Level 2 (Standard) or Level 3 (Premium with HSM). - **Managed HSM:** Dedicated HSM pool. Full control. FIPS 140-2 Level 3. For regulated workloads.

Certificate management: - Auto-enrollment and auto-renewal via Key Vault - Integration with DigiCert and GlobalSign CAs - Use App Service Managed Certificates for free auto-renewing SSL - Certificate rotation: Automate via Event Grid + Function

1.3 Design Governance

Management Group and Subscription Design

Common patterns:

Pattern 1: By Environment

```
Root
├── Production
│   ├── Shared Services Sub
│   └── Workload Sub(s)
├── Non-Production
│   ├── Dev Sub
│   └── Test Sub
└── Sandbox
    └── Individual Sub(s)
```

Pattern 2: By Business Unit

```
Root
├── Finance MG
│   ├── Prod Sub
│   └── Non-Prod Sub
```

```
├── Engineering MG
│   ├── Prod Sub
│   └── Non-Prod Sub
└── Marketing MG
```

Pattern 3: Hybrid (recommended for enterprise)

```
Root
├── Platform MG (shared services, networking, identity)
│   ├── Connectivity Sub (hub VNets, ExpressRoute, VPN)
│   ├── Identity Sub (Entra Connect, Domain Services)
│   └── Management Sub (Monitor, Automation, Defender)
├── Landing Zones MG
│   ├── Corp MG (internal workloads)
│   │   ├── Prod Sub
│   │   └── Non-Prod Sub
│   └── Online MG (external workloads)
│       ├── Prod Sub
│       └── Non-Prod Sub
└── Sandbox MG
    └── Individual Sub(s)
```

Subscription scaling factors: - Resource limits (25K VMs, 980 RGs per sub) - Management boundaries (different teams, different billing) - Compliance boundaries (different data classifications) - Network boundaries (VNet peering limits ~500)

Tagging Strategy

Tag Key	Purpose	Example
Environment	Workload classification	prod, staging, dev, test
CostCenter	Financial tracking	CC-1001, CC-2002
Owner	Responsible party	john@contoso.com
Application	Application name	app-invoicing, app-webportal
DataClassification	Security classification	public, internal, confidential, restricted
DR	DR tier	mission-critical, business-critical, non-critical
ExpirationDate	Auto-deletion for temp resources	2026-06-01

Enforcement via Azure Policy: - Require tag on creation - Inherit tag from resource group - Add default tag value if missing

Compliance Design

Azure Policy for compliance: - **Initiatives:** Map each compliance standard to an initiative (NIST, ISO, CIS, etc.) - **Guest configuration:** Audit settings inside VMs (password policy, installed software, file settings) - **Policy as code:** Store policy definitions in Git. CI/CD pipeline deploys policy assignments. - **Exemptions:** Time-bound or permanent exemptions for specific resources (documented and approved)

Microsoft Purview for data compliance: - Data classification (sensitive, PII, PHI) - Data lineage (track data flow across systems) - Data access policies (who can access what data) - Compliance dashboard (track compliance across data estate)

---

2. Design Data Storage Solutions (20-25%)

2.1 Design Data Storage Solutions for Relational Data

Relational Database Service Selection

Service	Description	Max Size	Use Case
Azure SQL Database	Managed SQL Server (single DB)	128 TB	SaaS apps, modern cloud apps, microservices
Azure SQL Managed Instance	Managed SQL Server (instance)	16 TB	Lift-and-shift, cross-database queries
Azure Database for PostgreSQL	Managed PostgreSQL	64 TB	Open-source preference, GIS, extensions
Azure Database for MySQL	Managed MySQL	16 TB	Open-source preference, LAMP stack
Azure Database for MariaDB	Managed MariaDB	16 TB	MariaDB-specific workloads
SQL Server on VM	Self-managed SQL on IaaS	256 TB	Full control, specific OS/SQL version needed

Service Tier Selection

Azure SQL Database:

Tier	DTU/vCore	Storage	Use Case
Basic	5 DTU	2 GB	Small dev/test
Standard	10-3000 DTU	1 TB	Production, mixed workloads
Premium	175-4000 DTU	4 TB	Mission-critical, high performance
General Purpose	2-80 vCores	Up to 128 TB	Most production workloads
Business Critical	2-80 vCores	Up to 128 TB	Low latency, high throughput, in-memory OLTP
Hyperscale	2-80 vCores	Up to 128 TB	Large databases, fast restore, read replicas

Purchasing model: - **DTU:** Simple, bundled compute+storage+IO. Good for small/medium workloads. - **vCore:** Fine-grained control. Separate compute and storage. Better for large workloads. Azure Hybrid Benefit applies.

Database Scalability

Vertical scaling (scale up): - Change service tier or vCore count - Online operation (minimal downtime for vCore changes) - Scheduled scaling for predictable workloads (scale down at night)

Horizontal scaling (scale out): - **Read replicas:** Offload read workloads to replicas (Geo-replication, Hyperscale replicas) - **Sharding:** Distribute data across multiple databases (Elastic Pool, sharding pattern) - **Elastic pools:** Shared eDTU/vCore pool across multiple databases. Cost-efficient for SaaS with variable demand.

Storage scaling: - Auto-grow: Automatically increase storage when approaching limit - Manual: Pre-provision storage (cheaper for predictable workloads)

Data Protection for Relational Data

- **Point-in-time restore (PITR):** Restore to any point within retention period (7-35 days)
- **Long-term retention (LTR):** Weekly/monthly/yearly backups for up to 10 years
- **Geo-replication:** Async replication to another region. Up to 4 readable secondaries.
- **Active geo-replication:** Readable secondaries. Manual failover.
- **Auto-failover groups:** Automatic failover to secondary region. Group of databases failover together.
- **Zone-redundant databases:** Synchronous replication across availability zones (Premium/Business Critical)

2.2 Design Data Storage Solutions for Semi-Structured and Unstructured Data

Semi-Structured Data Storage

Service	Format	Use Case
Azure Cosmos DB	JSON (multiple APIs)	Global distribution, low latency, multi-model
Azure Table Storage	Key-value (simple)	Simple, cheap NoSQL
Azure Cache for Redis	In-memory key-value	Caching, session store, real-time leaderboards
Azure Service Bus	Messages	Enterprise messaging, transactions
Event Hubs	Event streams	Big data streaming, event ingestion
Event Grid	Events	Reactive event routing, serverless

Cosmos DB design: - **Consistency levels (5):** Strong, Bounded Staleness, Session (default), Consistent Prefix, Eventual - **Multi-region writes:** Active-active replication. Conflict resolution (Last Write Wins, Custom) - **APIs:** NoSQL (native), MongoDB, PostgreSQL, Cassandra, Gremlin (graph), Table - **RU/s:** Request Units per second. Auto-scale or manual provisioning. 1 RU ≈ 1 KB read. - **Partition key:** Determines data distribution. Critical for performance. Choose high-cardinality, evenly distributed key.

Unstructured Data Storage

Service	Use Case
Azure Blob Storage	Files, images, videos, backups, logs, data lake
Azure Files	SMB/NFS file shares, lift-and-shift, shared config
Azure NetApp Files	Enterprise NFS/SMB, SAP HANA, high-performance
Azure Data Lake Storage Gen2	Big data analytics (HDFS-compatible), on top of Blob

Blob storage design: - **Container naming:** Organize by application, data type, or retention policy - **Naming convention:** Use prefixes for query performance (e.g., `logs/2026/04/23/`) - **Access patterns:** Hot (frequent), Cool (infrequent), Cold (rare), Archive (compliance) - **Lifecycle:** Automate tier transitions and deletion

Data Storage Balancing Features, Performance, and Cost

Decision framework:

Requirement	Primary Storage	Secondary/Tier
High throughput transactional	Premium SSD, Ultra Disk	N/A

Web/application files	Blob Hot	Blob Cool/Cold
Big data analytics	ADLS Gen2 Hot	ADLS Gen2 Cool
Shared file access	Azure Files Premium	Azure Files Standard
Archive/compliance	N/A	Blob Archive
In-memory caching	Redis Cache Premium	Redis Cache Standard
Global low-latency reads	Cosmos DB multi-region	Blob CDN

Data Protection and Durability

- **Geo-redundancy:** GRS/GZRS for cross-region DR
- **Backup:** Azure Backup for VMs, SQL, Files. Operational backup for Blobs.
- **Immutable storage:** WORM for compliance
- **Soft delete:** Recovery of deleted data
- **Versioning:** Point-in-time recovery for blobs
- **Point-in-time restore:** For blob storage (requires versioning + soft delete)

2.3 Design Data Integration

Data Integration Solutions

Service	Use Case
Azure Data Factory	ETL/ELT orchestration, data pipeline
Azure Synapse Analytics	Analytics, data warehouse, data lake
Azure Databricks	Spark-based analytics, ML
Azure Stream Analytics	Real-time stream processing
Azure Functions	Event-driven lightweight processing
Logic Apps	Workflow orchestration, integration

Data Factory design: - **Pipelines:** Orchestration of activities (copy, transform, monitor) - **Datasets:** Data structures and references - **Linked services:** Connection information - **Integration Runtimes:** Compute for data movement (self-hosted for on-prem, Azure for cloud) - **Triggers:** Schedule, tumbling window, event-based - **Mapping Data Flows:** Visual ETL (Spark-based) - **Wrangling Data Flows:** Power Query-based data preparation

Data Analysis Solutions

Service	Use Case
Azure Synapse Analytics	Enterprise data warehouse + analytics
Power BI	Business intelligence, visualization
Azure Analysis Services	Tabular data models for BI
Azure Databricks	Big data processing, ML, data science
Azure Stream Analytics	Real-time analytics on streaming data

---

3. Design Business Continuity Solutions (15-20%)

3.1 Design Solutions for Backup and Disaster Recovery

Recovery Objectives

Metric	Definition	Example
RPO (Recovery Point Objective)	Maximum acceptable data loss (time)	15 minutes, 1 hour, 24 hours
RTO (Recovery Time Objective)	Maximum acceptable downtime (time)	4 hours, 24 hours, 72 hours

Recovery Solutions by Workload

Azure VMs: - **Azure Backup:** Application-consistent backups. Daily/weekly frequency. Retention 7-9999 days. Instant restore. - **Azure Site Recovery:** Replicate VMs to secondary region. RPO: near-zero. RTO: minutes. Test failover. - **Availability Zones:** HA within region (protects against zone failure). RPO: 0. RTO: 0 (if load-balanced).

Databases: - **SQL Database:** PITR (7-35 days), LTR (up to 10 years), Geo-replication, Auto-failover groups, Zone-redundant - **Cosmos DB:** Multi-region writes, automatic failover. RPO: 0 for strong consistency within region. - **PostgreSQL/MySQL:** PITR (7-35 days), Read replicas for scale-out, Geo-redundant backup

Unstructured data: - **Blob storage:** GRS/GZRS for geo-redundancy. Soft delete for accidental deletion. Versioning for corruption. Point-in-time restore. - **Azure Files:** Azure Backup for file shares. Snapshot for point-in-time. GRS/GZRS for geo-redundancy.

Backup Strategy Design

3-2-1 rule (adapted for cloud): - 3 copies of data - 2 different storage types (Azure Backup vault + Storage Account) - 1 offsite (different region)

Azure Backup design: - **Recovery Services vault:** Per-region, per-workload-type - **Backup policies:** Frequency, retention, instant restore - **Cross-region restore:** Backup data available in paired region (requires GRS vault) - **Soft delete:** 14-day protection against malicious deletion. Enable purge protection. - **MARS agent:** For on-prem Windows servers - **DPM/MABS:** For enterprise backup (application-aware, SQL, Exchange, SharePoint)

DR Strategy Design

DR Strategy	RPO	RTO	Cost	Complexity
Backup & Restore	Hours	Hours-Days	Low	Low
Warm Standby (ASR)	Minutes	Minutes-Hours	Medium	Medium
Hot Standby (Active-Passive)	Seconds	Seconds-Minutes	High	Medium
Active-Active	Zero	Zero	Highest	Highest

Azure Site Recovery design: - **Replication policy:** RPO threshold, recovery point retention, app-consistent snapshot frequency - **Network mapping:** Map source VNet to target VNet - **Recovery plans:** Orchestrate failover of multiple VMs in correct order. Pre/post scripts. - **Test failover:** Non-disruptive DR testing. Isolated VNet. Delete after test. - **Failback:** Reverse replication after failover. Sync delta changes back.

3.2 Design for High Availability

Compute HA

Solution	SLA	Protects Against	Cost Impact
Single VM with Premium SSD	99.9%	Disk failure	None
Availability Set	99.95%	Rack/switch failure	None (same VM cost)
Availability Zones	99.99%	Datacenter failure	2x VM cost (2+ VMs)
Region pairs (DR)	-	Region failure	2x all resources

App Service HA: - Standard+ tier: 99.95% SLA - Premium v3: Zone-redundant (99.99% effective) - Deployment slots for zero-downtime deployments - Auto-scaling for demand spikes - Traffic Manager / Front Door for multi-region

AKS HA: - Multi-zone node pools (spread across 3 zones) - Pod disruption budgets (ensure min available pods during updates) - Cluster autoscaler (maintain capacity) - Multi-region clusters with Traffic Manager / Front Door

Relational Data HA

SQL Database: - Zone-redundant configuration (Premium/Business Critical/General Purpose) - Active geo-replication (up to 4 readable secondaries) - Auto-failover groups (automatic failover across regions) - Always On availability groups (SQL MI)

Cosmos DB: - Multi-region writes: Active-active across regions - Automatic failover: Priority-ordered region failover - Consistency levels: Strong for single-region, Session for multi-region

Unstructured Data HA

- **ZRS:** HA within a single region (3 zones)
- **GZRS:** HA within region + geo-redundancy
- **RA-GZRS:** Read from secondary during primary outage
- **CDN/Front Door:** Global caching and failover for blob content

---

4. Design Infrastructure Solutions (30-35%)

4.1 Design Compute Solutions

Workload Requirements Analysis

Requirement	Compute Type	Azure Service
Full OS control, long-running	VMs	Azure VMs, VMSS
Containerized microservices	Containers	AKS, Container Apps, ACI
Event-driven, short-lived	Serverless	Functions, Logic Apps
Web/API hosting	PaaS	App Service
Batch processing	Batch	Azure Batch
GPU/AI workloads	Specialized	AKS with GPU, Azure ML compute
Desktop virtualization	VDI	Azure Virtual Desktop

VM-Based Solutions

Design considerations: - **Right-sizing:** Start with estimated size, monitor, and right-size. Use Azure Advisor

recommendations. - **Reserved Instances:** For steady-state workloads. 1-year or 3-year. Up to 72% savings. - **Spot VMs:** For fault-tolerant, interruptible workloads. Up to 90% savings. - **Burstable (B-series):** For dev/test with occasional spikes. Credits accumulate during low usage. - **Storage optimization:** Ultra Disk for databases, Premium SSD for production, Standard for dev. - **Network optimization:** Accelerated Networking (SR-IOV) for low latency. Proximity Placement Groups for co-location.

VM architecture patterns: - **N-tier:** Web tier (VMSS) → App tier (VMSS) → Data tier (SQL/Cosmos). Classic pattern. - **Lift-and-shift:** Migrate on-prem VMs as-is to Azure. Modernize incrementally. - **Big compute:** HPC with HB/HC-series VMs. CycleCloud for orchestration. Batch for job scheduling.

Container-Based Solutions

Service	Orchestration	Scaling	Use Case
AKS	Full Kubernetes	HPA, KEDA, cluster autoscaler	Complex microservices, K8s teams
Container Apps	Managed K8s (abstracted)	KEDA (scale to zero)	Serverless containers, event-driven
ACI	None	Manual	Simple containers, batch, CI/CD
Service Fabric	SF runtime	Built-in	Stateful microservices, reliable services

AKS design: - **Node pools:** System pool (3+ nodes, dedicated) + User pools (per workload type) - **Multi-zone:** Spread nodes across zones. Zone-aware pod scheduling. - **Private cluster:** API server on private endpoint. No public access. - **Network plugin:** CNI (pod IPs from VNet) vs kubenet (NAT through node IP). CNI recommended. - **Ingress:** Application Gateway Ingress Controller (WAF) or NGINX Ingress Controller - **Secrets:** Secrets Store CSI Driver with Key Vault provider

Serverless Solutions

Azure Functions: - **Hosting plans:** Consumption (pay-per-use, scale to zero), Premium (pre-warmed, VNet, unlimited duration), Dedicated (App Service plan, full control) - **Durable Functions:** Stateful orchestration. Patterns: Function Chaining, Fan-out/Fan-in, Async HTTP API, Monitor, Human Interaction. - **Triggers:** HTTP, Timer, Queue, Service Bus, Event Hub, Blob, Cosmos DB, Event Grid, SignalR

Design considerations: - Cold start: Consumption plan has cold start latency. Premium plan pre-warms instances. - Execution timeout: 5 min (Consumption), 60 min (Premium), unlimited (Dedicated) - State management: Use external storage (Cosmos DB, Storage) for state. Don’t store state in function instance.

Batch Processing Solutions

Azure Batch: - Managed job scheduling and auto-scaling for HPC workloads - Pool of VMs (Linux/Windows). Auto-scale based on task queue. - Job scheduling: Job → Tasks. Tasks run on pool nodes. - Use for: Rendering, financial modeling, scientific simulation, data transformation - **Low-priority VMs:** Use spot capacity for batch workloads. Up to 80% savings.

4.2 Design Application Architecture

Messaging Architecture

Service	Pattern	Ordering	Transactions	Use Case
Service Bus	Brokered messaging	FIFO (with sessions)	Yes	Enterprise integration, order processing
				Reactive, serverless,

Event Grid	Event routing	No	No	notifications
Event Hubs	Event streaming	Partition-based	No	Big data, telemetry, log ingestion
Storage Queues	Simple messaging	No	No	Simple task distribution, low cost

Design decisions: - Need transactions and FIFO? → Service Bus - Need massive throughput (MB/s+)? → Event Hubs - Need reactive/event-driven architecture? → Event Grid - Need simple, cheap queue? → Storage Queues - Need pub/sub? → Service Bus topics or Event Grid

Event-Driven Architecture

Pattern: Event Sourcing + CQRS - Store all changes as events (append-only log) - Materialized views for queries (CQRS read models) - Azure services: Event Hubs (event stream), Cosmos DB Change Feed (event processing), Functions (event handlers)

Pattern: Pub/Sub - Publishers emit events without knowing subscribers - Event Grid: Push events to subscribers (webhooks, Event Hubs, Service Bus, Functions) - Service Bus topics: Pub/sub with filtering, sessions, transactions

API Integration

Service	Use Case
API Management	API gateway, rate limiting, auth, transformation, developer portal
Application Gateway	Layer 7 load balancing, WAF, SSL termination for web APIs
Azure Front Door	Global routing, WAF, CDN for APIs with global users
Azure Relay	Expose on-prem APIs to cloud without opening firewall ports

API Management design: - **Products:** Group APIs for access control and monetization - **Policies:** Transform requests/responses, validate JWT, rate limit, cache, retry - **Versions/Revisions:** API versioning without downtime. Revisions for safe changes. - **Developer portal:** Self-service API discovery, documentation, try-it - **Backends:** Multiple backend types (HTTP, Service Fabric, Function App)

Caching Solutions

Service	Type	Use Case
Azure Cache for Redis	In-memory, distributed	Session state, data cache, rate limiting, leaderboards
CDN (Front Door/Azure CDN)	Edge cache	Static content, media, API responses
App Service local cache	Local disk cache	Read-only content, faster app startup

Redis caching patterns: - **Cache-aside:** Application checks cache first, on miss loads from DB and populates cache - **Write-through:** Write to cache and DB simultaneously - **Write-behind:** Write to cache, async write to DB - **TTL:** Set expiration to prevent stale data. Use appropriate TTL per data type.

Application Configuration Management

Service	Use Case
Azure App Configuration	Centralized app settings, feature flags, dynamic configuration
Key Vault	Secrets, certificates, keys (NOT general config)

Environment variables	Simple, per-deployment settings

App Configuration features: - Feature flags: Toggle features without redeployment. Targeting (by user/group/percentage) - Dynamic configuration: Change settings without restart - Labeling: Different values per environment (dev, staging, prod) - Point-in-time snapshot: Rollback configuration changes

Automated Deployment Solutions

Service	Use Case
Azure DevOps Pipelines	CI/CD for Azure workloads, YAML-based
GitHub Actions	CI/CD integrated with GitHub
Azure Pipelines	Part of Azure DevOps. Multi-stage YAML pipelines
Azure Deployment Slots	Zero-downtime deployment for App Service

Deployment strategies: - **Blue/Green:** Two identical environments. Deploy to idle (green), swap to active (blue) - **Canary:** Deploy to small percentage of users. Monitor. Gradually increase. - **Rolling:** Gradually replace old instances with new. Zero downtime. - **Feature flags:** Deploy code with features off. Enable for users gradually.

4.3 Design Migrations

Cloud Adoption Framework

Migration phases: 1. **Strategy:** Define business justification and expected outcomes 2. **Plan:** Rationalize digital estate, build migration backlog 3. **Ready:** Build landing zone (Azure infrastructure ready for workloads) 4. **Migrate:** Execute migration waves 5. **Govern:** Implement governance and compliance 6. **Manage:** Operate, monitor, optimize

On-Premises Assessment

Azure Migrate: - **Server assessment:** Discover VMs, performance data, suitability analysis, sizing, cost estimation - **Database assessment:** Assess SQL Server readiness for Azure SQL Database/MI - **Web app assessment:** Assess IIS/ASP.NET apps for App Service migration - **Dependencies:** Agentless dependency mapping (visualize server dependencies)

Migration Solutions

IaaS Migration (Lift and Shift): - **Azure Migrate Server Migration:** Agentless (VMware) or agent-based (Hyper-V, physical) - **Replication:** Continuous block-level replication to Azure - **Test migration:** Test in isolated VNet before final cutover - **Cutover:** Planned downtime. Final delta sync → failover → validate → go-live

PaaS Migration (Modernize): - **Azure Database Migration Service (DMS):** Migrate SQL Server → Azure SQL Database/MI. Online (minimal downtime) or offline. - **App Service Migration Assistant:** Assess and migrate IIS web apps to App Service - **Azure Migrate for databases:** Assess and migrate databases

Database migration strategies: - **Offline:** Stop application → migrate data → start application on Azure. Simplest, most downtime. - **Online:** Keep app running → sync data to Azure → cutover. Minimal downtime. Requires DMS. - **Hybrid:** Migrate schema first → sync data → cutover application.

Unstructured Data Migration

- **AzCopy:** High-performance CLI. For TB-scale migrations over network.
- **Azure Data Box:** Physical device. For PB-scale or limited-bandwidth migrations.
- **Azure File Sync:** Migrate file servers gradually. Sync on-prem to cloud.
- **Storage Migration Service:** Windows Server built-in. Discover, inventory, migrate file servers.

4.4 Design Network Solutions

Internet Connectivity

Architecture:

```
Internet → Azure Front Door / CDN (global routing, WAF)
         → Application Gateway (regional routing, WAF, SSL termination)
         → Web Tier (App Service / AKS / VMSS)
         → App Tier
         → Data Tier
```

Design decisions: - Global users? → Front Door (global anycast, WAF) - Single region? → Application Gateway (WAF, SSL, routing) - Both? → Front Door → App Gateway (end-to-end WAF, defense in depth)

On-Premises Connectivity

Option	Bandwidth	Latency	Use Case
Site-to-Site VPN	Up to 10 Gbps	Variable (internet)	DR, backup, low-volume
Point-to-Site VPN	Varies	Variable (internet)	Remote workers
ExpressRoute	Up to 100 Gbps	Low, consistent	Production, high-volume, regulated
ExpressRoute Global Reach	Circuit-to-circuit	Low	Connect on-prem sites via Microsoft backbone

Hub-spoke network topology:

```
On-Premises → ExpressRoute/VPN → Hub VNet (shared services)
                                    ├── Azure Firewall / NVA
                                    ├── VPN Gateway
                                    ├── DNS
                                    └── Spoke VNets
                                        ├── App VNet
                                        ├── Data VNet
                                        └── Management VNet
```

VNet peering for hub-spoke: Each spoke peers with hub. Hub provides routing, firewall, and shared services. No spoke-to-spoke peering (traffic goes through hub).

Network Performance Optimization

- **Accelerated Networking:** SR-IOV bypasses hypervisor. Lower latency, higher throughput. Enable on all VMs.
- **Proximity Placement Groups:** Co-locate VMs for low inter-VM latency. Use for: clustered databases, HPC.
- **ExpressRoute FastPath:** Bypass gateway for data path. Lower latency, higher throughput.
- **CDN caching:** Cache content at edge. Reduce origin load, improve user latency.
- **TCP optimization:** Enable TCP selective ACK, window scaling on VMs.

Network Security Optimization

Defense in depth: 1. **Perimeter:** DDoS Standard, Azure Firewall, WAF 2. **Network:** NSGs, ASGs, UDRs to force traffic through firewalls 3. **Application:** API Management policies, OAuth/OIDC 4. **Data:** Private endpoints, encryption in transit

Zero Trust networking: - Assume breach. Verify explicitly. Least privilege. - Private endpoints for ALL PaaS services - Microsegmentation with NSGs per subnet - East-west traffic inspection (firewall between VNets) - Encrypted communication (TLS everywhere, VPN for management)

Load Balancing and Routing

Service	Layer	Scope	Routing	SSL	WAF
Azure Load Balancer	L4	Regional	IP:Port	No	No
Application Gateway	L7	Regional	URL/Host/Cookie	Yes	Yes
Front Door	L7	Global	URL/Host/Latency	Yes	Yes
Traffic Manager	DNS	Global	Latency/Priority/Geo	No	No

Design decisions: - Global, need WAF, latency-based routing? → Front Door - Single region, need WAF, URL routing? → Application Gateway - Layer 4, TCP/UDP, high performance? → Azure Load Balancer - DNS-level routing, no SSL termination needed? → Traffic Manager

---

Quick Reference: AZ-305 Design Decision Framework

Design Area	Key Decisions
Identity	PHS vs PTA vs Federation, B2B vs B2C, PIM strategy
Governance	MG hierarchy, subscription strategy, Policy initiatives
Monitoring	Workspace topology, retention, alert strategy
Relational Data	SQL DB vs SQL MI vs PostgreSQL, tier, HA/DR
Unstructured Data	Blob vs Files vs ADLS, tier, redundancy
Business Continuity	RPO/RTO requirements, backup strategy, DR pattern

---

Appendix A: Architecture Design Decision Trees

Compute Decision Tree

```
START: What are you running?
|
├── Existing on-prem application → Lift-and-shift to VMs
|   ├── Needs full OS control? → VMs / VMSS
|   ├── Can modernize? → Containerize → AKS / Container Apps
|   └── Simple web app? → App Service
|
├── New cloud-native application
|   ├── Web/API backend?
|   |   ├── Simple, single service → App Service
|   |   ├── Microservices → AKS or Container Apps
|   |   └── Event-driven → Functions + Event Grid
|   |
|   ├── Batch/processing?
|   |   ├── HPC → Azure Batch / CycleCloud
```

```
│   │   ├── Simple batch → Functions / Container Apps Jobs
│   │   └── Data pipeline → Data Factory + Databricks
│   │
│   └── AI/ML workload?
│       ├── Training → AKS with GPU / Azure ML Compute
│       ├── Inference → AKS / Container Apps / Azure ML Endpoints
│       └── Real-time → Functions + Cognitive Services
│
└── Desktop/app virtualization → Azure Virtual Desktop
```

Database Decision Tree

```
START: What kind of data?
│
├── Relational (structured, ACID)
│   ├── SQL Server workload?
│   │   ├── Single database, cloud-native → Azure SQL Database
│   │   │   ├── General Purpose → most workloads
│   │   │   ├── Business Critical → low latency, high throughput
│   │   │   └── Hyperscale → large DBs, fast restore
│   │   │
│   │   ├── Instance-level features needed → SQL Managed Instance
│   │   │   ├── Cross-database queries, SQL Agent, etc.
│   │   │   └── Lift-and-shift with minimal changes
│   │   │
│   │   └── Full control needed → SQL on VM
│   │
│   ├── PostgreSQL workload? → Azure Database for PostgreSQL
│   │   ├── Flexible Server (recommended)
│   │   └── Cosmos DB PostgreSQL
│   │
│   └── MySQL workload? → Azure Database for MySQL Flexible Server
│
├── NoSQL (semi-structured, flexible schema)
│   ├── Global distribution needed? → Cosmos DB
│   │   ├── API: NoSQL (native), MongoDB, Cassandra, Gremlin, Table
│   │   ├── Consistency: Strong → Eventual (5 levels)
│   │   └── Multi-region writes with automatic conflict resolution
│   │
│   ├── Simple key-value? → Azure Table Storage (cheap)
│   └── Need caching? → Azure Cache for Redis
│
├── Unstructured (files, images, videos)
│   ├── Object storage → Azure Blob Storage
│   │   ├── Hot: Active data
│   │   ├── Cool: Infrequent access (30-day min)
│   │   ├── Cold: Rare access (90-day min)
│   │   └── Archive: Compliance (180-day min)
│   │
│   ├── File shares → Azure Files (SMB) or NetApp Files (enterprise)
│   ├── Big data analytics → ADLS Gen2 (Blob + HDFS)
│   └── Data lake + warehouse → Synapse Analytics
│
└── Streaming data
    ├── Event ingestion → Event Hubs (high throughput)
```

```
    ├── Stream processing → Stream Analytics
    └── Message queuing → Service Bus (transactions) or Storage Queues (simple)
```

Network Decision Tree

```
START: What connectivity do you need?
│
├── Connect to on-premises
│   ├── Consistent low latency + high bandwidth? → ExpressRoute
│   │   ├── 1 Gbps+ needed? → ExpressRoute Direct
│   │   └── Standard? → ExpressRoute via provider
│   │
│   ├── VPN acceptable? → VPN Gateway
│   │   ├── Site-to-Site (DC to Azure)
│   │   ├── Point-to-Site (individual users)
│   │   └── Redundancy: Active-active gateways
│   │
│   └── Hybrid connectivity? → Virtual WAN
│       ├── Branch offices + Azure + on-prem
│       └── Secured hub with Azure Firewall
│
├── Connect Azure VNets
│   ├── Same region, few VNets → VNet Peering
│   ├── Cross-region → Global VNet Peering
│   ├── Many VNets, hub-spoke → Virtual WAN or Hub VNet
│   └── Need transitive routing → Virtual WAN or NVA + UDR
│
├── Load balancing
│   ├── Global, Layer 7 → Azure Front Door
│   ├── Regional, Layer 7 → Application Gateway
│   ├── Regional, Layer 4 → Azure Load Balancer
│   └── DNS-based → Traffic Manager
│
└── Security
    ├── Firewall → Azure Firewall (Standard or Premium)
    ├── WAF → Application Gateway WAF or Front Door WAF
    ├── DDoS → DDoS Standard
    └── Private access → Private Endpoints + Private Link
```

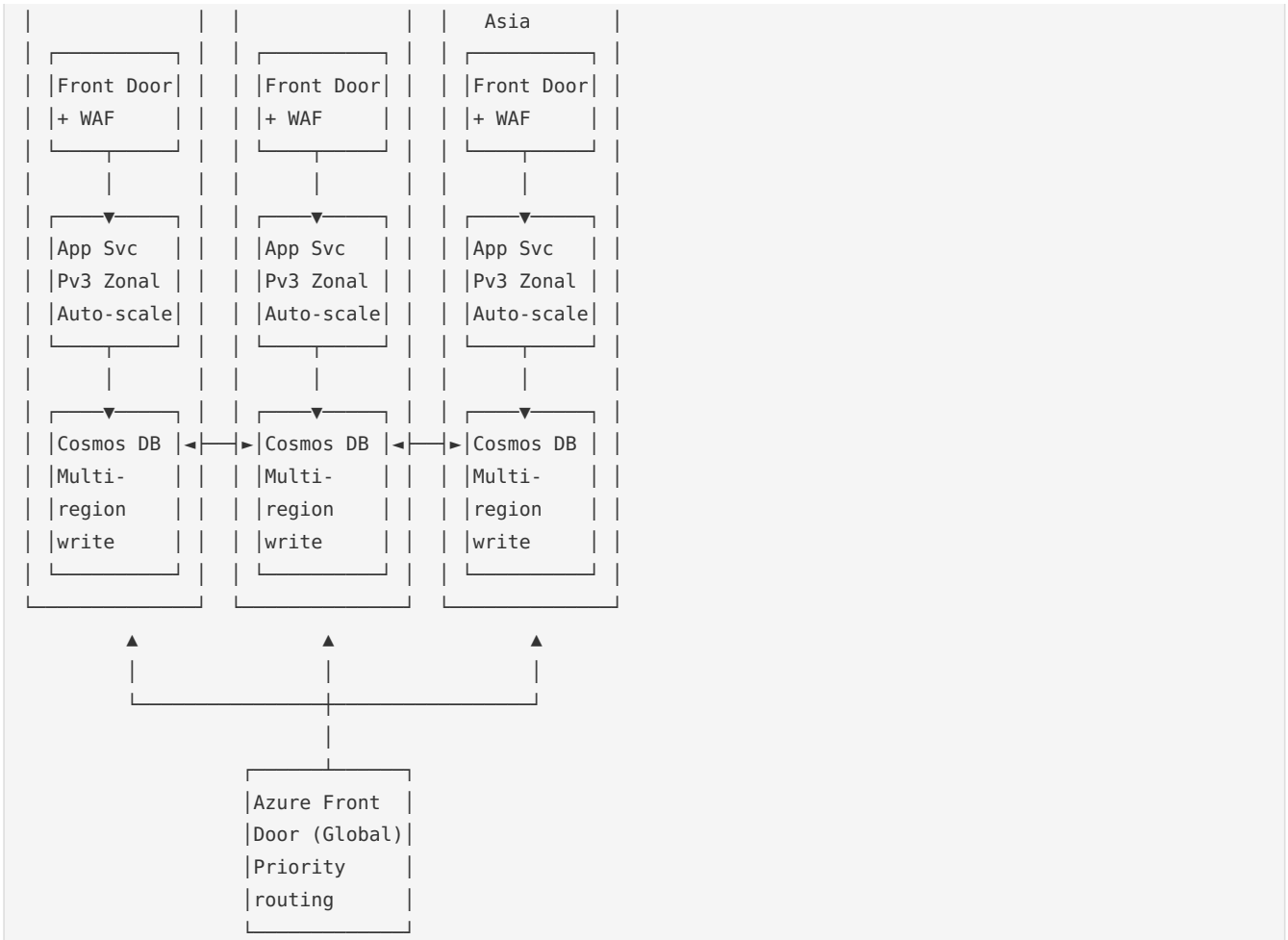
Appendix B: Design Scenarios

Scenario 1: E-Commerce Platform (Global, High Traffic)

Requirements: - Global customer base (US, Europe, Asia) - 99.99% availability SLA - Sub-200ms response time globally - PCI DSS compliance (payment data) - Seasonal traffic spikes (10x normal) - Real-time inventory updates

Design:

```
┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐
│  US Region      │  │ EU Region       │  │ Asia Region     │
│  East US        │  │ West Europe     │  │ Southeast       │
```

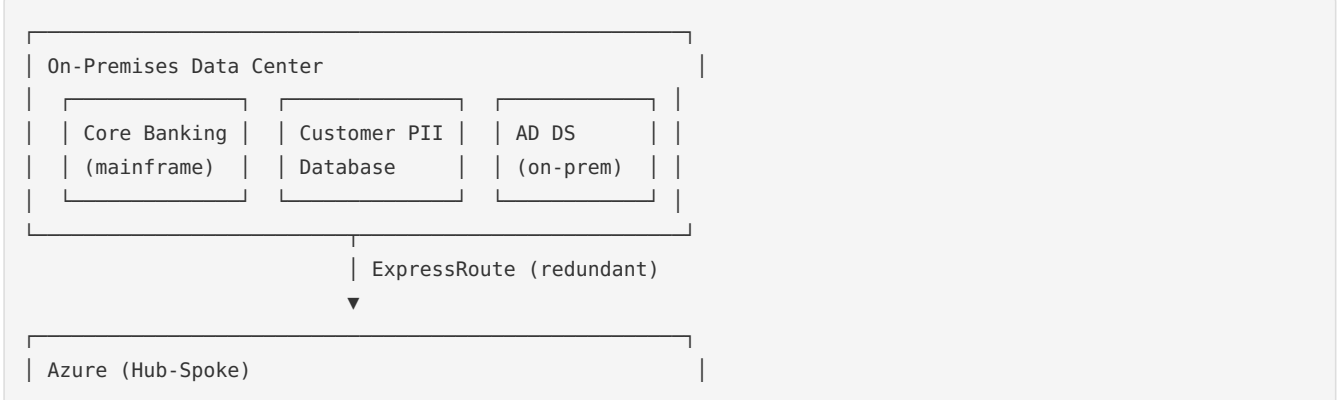


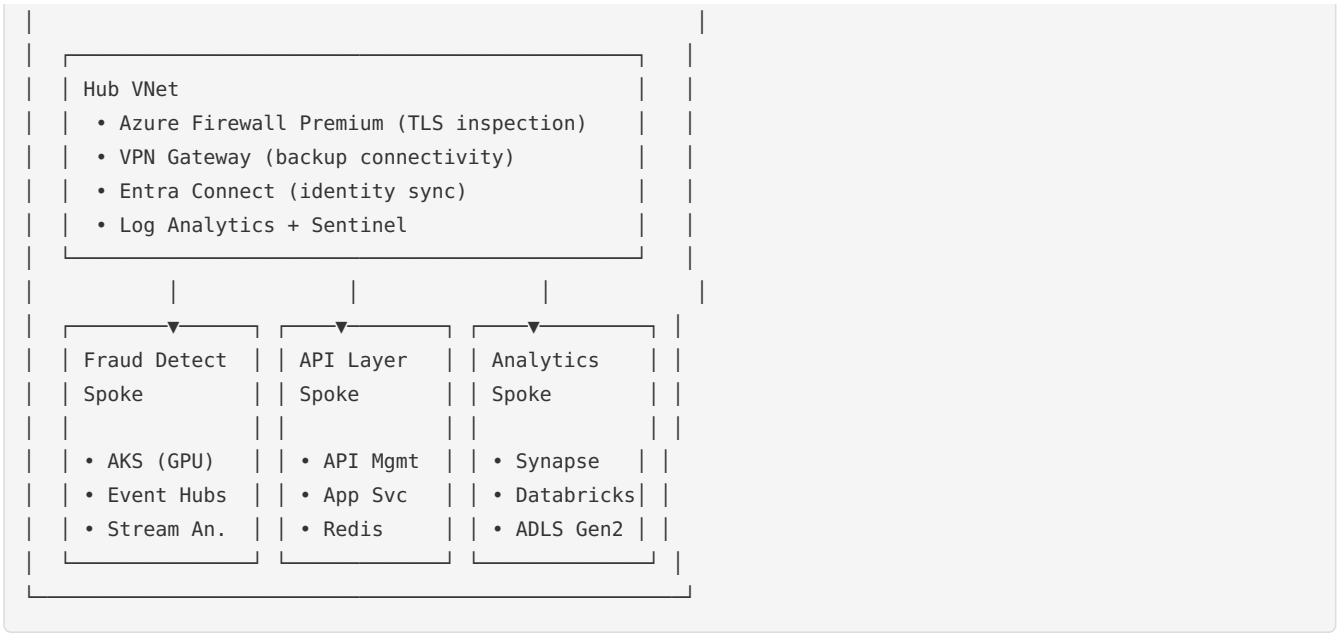
Key design decisions: - **Compute:** App Service Premium v3 (zone-redundant, auto-scaling). Handles 10x spikes via auto-scale rules. - **Database:** Cosmos DB multi-region writes. Session consistency for shopping cart. Strong consistency for inventory (single write region with change feed). - **Global routing:** Front Door with WAF. Priority routing to nearest region. Failover to secondary. - **Compliance:** Private endpoints for all PaaS. CMK encryption. PCI DSS initiative in Azure Policy. Application Gateway WAF with OWASP rules. - **Caching:** Redis Cache for session state and product catalog. Front Door caching for static assets. - **CDN:** Front Door integrated CDN for images and static content.

Scenario 2: Financial Services (Regulated, Hybrid)

Requirements: - Some data must stay on-premises (regulatory) - Real-time fraud detection - SOC 2 + ISO 27001 compliance - Audit trail for all access - Zero downtime for core banking

Design:





Key design decisions: - **Hybrid:** ExpressRoute for primary, VPN for backup. Core banking stays on-prem. - **Fraud detection:** Event Hubs → Stream Analytics → AKS (ML model). Sub-second detection. - **Compliance:** Azure Policy with ISO 27001 + SOC 2 initiatives. Key Vault CMK for all encryption. Complete audit trail in Sentinel. - **Security:** Azure Firewall Premium with TLS inspection and IDPS. Private endpoints everywhere. JIT VM access. PIM for all admin roles. - **HA:** Zone-redundant services. Cross-region DR with Site Recovery.

Scenario 3: SaaS Multi-Tenant Application

Requirements: - 1000+ tenants, growing - Tenant data isolation - Per-tenant customization - Cost-efficient at scale - Self-service onboarding

Design:

```
│                                                    │
│  ┌──────────────────────────────────────────┐      │
│  │ Shared Infrastructure                    │      │
│  │  • App Service Environment (isolated)    │      │
│  │  • Azure Cache for Redis (per-tenant prefix) │  │
│  │  • Storage Account (per-tenant container)    │  │
│  └──────────────────────────────────────────┘      │
└────────────────────────────────────────────────────┘
```

Key design decisions: - **Tenant isolation:** Database-per-tenant (strongest isolation, higher cost) or shared with partition key (lower cost, weaker isolation) - **Compute:** App Service Environment for full VNet isolation and high density - **Routing:** API Management with tenant-specific backends and policies - **Secrets:** Key Vault with per-tenant keys (access policy per app identity) - **Scaling:** Elastic pool auto-scales compute across tenant databases. App Service auto-scales on request count.

Appendix C: Well-Architected Framework Review

Five Pillars of Azure Well-Architected Framework

1. Reliability

Design principles: - Design for failure (assume everything fails) - Use redundancy at every layer (zones, regions, services) - Implement retry logic with exponential backoff - Test failure scenarios (chaos engineering) - Monitor health and set up proactive alerts

Key questions: - What is your RPO/RTO for each workload? - How do you test DR? - What happens when a zone goes down? A region? - How do you handle dependency failures?

2. Security

Design principles: - Zero Trust (verify explicitly, least privilege, assume breach) - Encrypt everything (at rest and in transit) - Automate security responses - Protect data at every layer - Regular security assessments

Key questions: - How are identities authenticated and authorized? - What is your blast radius if credentials are compromised? - How do you detect and respond to threats? - How do you classify and protect sensitive data?

3. Cost Optimization

Design principles: - Right-size resources (monitor and adjust) - Use reserved capacity for steady-state workloads - Implement auto-scaling to match demand - Use spot instances for interruptible workloads - Monitor and optimize continuously

Key questions: - What is the cost per transaction/user? - How do you track and allocate costs? - What is your waste percentage (idle/underutilized)? - When did you last right-size your resources?

4. Operational Excellence

Design principles: - Infrastructure as Code (all deployments automated) - Observable systems (monitoring, logging, alerting) - CI/CD for all changes - Incident response automation - Continuous improvement (blameless postmortems)

Key questions: - How long does it take to deploy a change? - How do you roll back a failed deployment? - How do you detect and respond to incidents? - What is your mean time to recovery (MTTR)?

5. Performance Efficiency

Design principles: - Design for scalability (horizontal over vertical) - Test at scale (load testing, performance testing) - Use caching strategically (Redis, CDN) - Optimize data access (indexing, partitioning, read replicas) - Monitor and optimize continuously

Key questions: - What is your maximum throughput? - How does your system perform under 10x load? - Where are your bottlenecks? - How do you validate performance before deployment?

Appendix D: Migration Design Patterns

Migration Wave Planning

```
Wave 1: Easy Wins (Low Risk, Low Dependency)
├── Dev/Test environments
├── Static websites → App Service
├── File shares → Azure Files
└── Simple VMs (no dependencies)

Wave 2: Supporting Infrastructure
├── Domain controllers → Entra Domain Services / DC VMs
├── DNS → Azure DNS
├── VPN connectivity → VPN Gateway / ExpressRoute
└── Monitoring → Log Analytics + Application Insights

Wave 3: Data Tier
├── SQL Server → Azure SQL Database / MI
├── MySQL → Azure Database for MySQL
├── MongoDB → Cosmos DB (MongoDB API)
└── File servers → Azure Files + File Sync

Wave 4: Application Tier
├── Web apps → App Service
├── APIs → App Service / API Management
├── Windows services → Container Apps / Functions
└── Batch jobs → Azure Batch / Functions

Wave 5: Complex / Legacy
├── Mainframe integrations
├── Legacy authentication
├── Complex multi-tier apps
└── Cross-dependency workloads
```

Migration Assessment Checklist

- ☐ Inventory all servers, databases, and applications
- ☐ Map dependencies between workloads (use Azure Migrate dependency mapping)
- ☐ Classify each workload: Rehost, Refactor, Rearchitect, Rebuild, or Replace
- ☐ Determine migration method per workload (agentless, agent-based, database migration)
- ☐ Estimate migration timeline and downtime windows
- ☐ Plan IP address allocation (avoid conflicts between on-prem and Azure)
- ☐ Design network connectivity (ExpressRoute or VPN, bandwidth requirements)
- ☐ Plan identity integration (Entra Connect, domain join)
- ☐ Document DNS changes and cutover plan

- [ ] Plan rollback strategy for each migration wave
- [ ] Estimate costs (Azure pricing calculator + migration tooling costs)
- [ ] Obtain stakeholder sign-off on migration plan