POLICY & ARCHITECTING & MIGRATIONS & ADVANCED SERVICES

Azure Policy (3 paragraphs)

Azure Policy is a governance tool that allows you to enforce rules across your Azure environment. These policies ensure that resources comply with corporate standards and service-level agreements. Common examples include restricting the creation of resources in specific regions, enforcing tag usage for cost management, or limiting VM sizes. Policies can be defined individually or grouped into initiatives.

An initiative is a collection of related policy definitions that help achieve a broader goal. For instance, a security compliance initiative might include policies for encryption, NSG use, and naming conventions. Policies themselves are created with definitions that describe their purpose, mode of operation (e.g., "All" or "Indexed"), and logic (typically expressed in JSON using conditions and effects).

Policy parameters allow reuse of definitions with different values, increasing flexibility. Policies or initiatives are assigned to scopes such as subscriptions, management groups, or resource groups. Each policy has an "effect" that determines what action it takes — such as "Deny", "Audit", or "Append". These help maintain compliance and drive accountability.

Custom Policies (1 paragraph)

Custom policies are JSON documents written to define governance rules that aren’t available out-of-the-box. They’re used when built-in policies are insufficient. While rarely needed, they provide flexibility for advanced scenarios, such as enforcing a custom tag format or preventing specific SKUs from being deployed. Writing custom policies requires a good grasp of Azure Policy structure and syntax.

Architecture: Choosing a Compute Platform (2 paragraphs)

Choosing the right compute platform in Azure depends on factors like scalability, management overhead, and deployment complexity. Options include Azure Virtual Machines for full control, App Services for simplified hosting of web applications, Azure Kubernetes Service (AKS) for container orchestration, and Azure Functions for serverless computing. Each has tradeoffs in terms of control, scalability, and pricing.

For example, VMs offer maximum control and compatibility for legacy apps but require more management. App Services abstract much of the infrastructure, making them ideal for standard web apps. AKS provides fine-grained control over containerized workloads, while Azure Functions are perfect for event-driven tasks that need to scale massively and cost-effectively.

Choosing a Data Platform (2 paragraphs)

Azure offers a range of data platforms tailored to different needs. SQL Database is a managed relational service suited for most enterprise applications, offering scaling, security, and high availability. Cosmos

DB supports globally distributed, multi-model data with low latency, ideal for internet-scale apps. Azure Table Storage is a simple key-value store for massive, inexpensive data.

Choosing the right data platform involves considering query patterns, consistency requirements, latency, and data structure. For instance, if you need relational features like joins and stored procedures, go with SQL Database. For unstructured or rapidly changing data across geographies, Cosmos DB shines. Blobs, tables, queues, and files are suitable for raw storage, logging, or simple async messaging.

Choosing a Messaging Platform (2 paragraphs)

Azure provides several messaging options. Azure Service Bus supports reliable enterprise messaging with queues and topics, ideal for complex routing and ordering needs. Azure Event Grid is best for lightweight, event-driven architectures and supports webhook subscriptions. Azure Event Hubs is a high-throughput event ingestion service, great for telemetry and analytics pipelines.

Choosing depends on message volume, latency tolerance, and system design. Service Bus works well for financial or supply-chain systems that require guaranteed delivery. Event Hubs suits real-time processing, such as IoT or logging scenarios. Event Grid is efficient for triggering serverless functions or notifying components when events occur across resources.

Implementing Security (2 paragraphs)

Security in Azure starts with controlling access. Restricting access to VMs and App Services involves using Network Security Groups (NSGs) to control traffic and minimizing open ports. Encryption should be enforced at rest and in transit, and sensitive data should use customer-managed keys when possible. Authentication should rely on strong mechanisms like Azure AD with Multi-Factor Authentication (MFA).

Role-Based Access Control (RBAC) ensures users only access what they need. Service Principals and Managed Identities help services authenticate securely. Resources should be protected using Private Endpoints where possible, reducing exposure to the public internet. Security Center and Defender for Cloud offer ongoing recommendations and threat detection.

Implementing Logging and Monitoring (2 paragraphs)

Effective monitoring includes setting up alerts for abnormal conditions like failed deployments or high CPU usage. Azure Monitor provides a centralized solution to track performance and logs. Log Analytics allows querying historical logs and telemetry to detect patterns or issues over time.

Dashboards can visualize system health using metrics, logs, and custom tiles. Application Insights, part of Azure Monitor, provides deep insight into app performance and user behavior, including response times and exceptions. Integrating monitoring into your CI/CD and DevOps workflow helps you detect and fix issues early.

Azure Architecture Center (2 paragraphs)

The Azure Architecture Center is a Microsoft-maintained repository of guidance, best practices, and reference architectures. It provides patterns and anti-patterns for building secure, scalable, and resilient

cloud applications. The center addresses key design pillars: scalability, availability, manageability, security, and cost.

It’s particularly useful when planning new workloads or migrating existing systems. Each reference architecture includes diagrams, descriptions, implementation guidance, and often, links to deployable templates. For example, you can find architectures for web apps with CI/CD pipelines, microservices, or hybrid cloud solutions.

Migration Process (9 paragraphs)

Migrating to the cloud is not just a technical endeavor — it starts with understanding *why* you're migrating. Common motivations include cost savings, taking advantage of cloud elasticity and capabilities, modernization of aging systems, and improving employee satisfaction by working on modern platforms. A detailed cost estimation should be conducted to ensure cloud costs are well understood, and any promises of agility must be verified in context.

System assessment is critical before any move. This includes analyzing the current runtime (e.g., .NET Framework vs .NET Core), deployment methods, databases used, authentication mechanisms, hardware dependencies, and integration with internal systems. In some cases, this assessment reveals cloud migration isn’t viable. Azure Migrate helps at this stage by assessing your on-prem environment and offering sizing, compatibility, and readiness analysis — but it’s only the beginning.

There are three main migration strategies: **Lift and Shift**, where VMs are moved as-is to the cloud with minimal changes; **Refactoring**, which replaces some VM-based components with PaaS offerings like App Services or SQL Database; and **Rewrite**, where the app is redesigned from scratch to be fully cloud-native, using services like Azure Functions, AKS, or Cosmos DB.

After migrating, applications are often enhanced. Typical enhancements include improving observability with Azure Monitor, and adding layers of security, such as Azure Application Gateway with WAF (Web Application Firewall) for network-level protection. Cost controls and tagging mechanisms should also be introduced at this point.

Testing is vital before going live. It involves performance benchmarking to compare against the on-prem baseline, verifying connectivity to critical systems, and ensuring the logging and monitoring pipelines are in place. Strong emphasis must be placed on verifying logging granularity and alert configurations.

Going live involves switching production traffic to the cloud environment. Budget alerts should be configured to avoid unexpected costs. Tagging resources helps track ownership, purpose, and cost center. Monthly cost reviews help identify idle resources or inefficient configurations that can be optimized.

Post-migration, you may need to upskill your team or revise your DevOps practices. You should also review disaster recovery (DR) plans in light of the new architecture. Monitoring should evolve as usage patterns stabilize, ensuring the system remains healthy over time.

Ongoing optimization is key. Consider autoscaling, reservation pricing for predictable workloads, and possible further modernization to take full advantage of Azure's ecosystem. Engage with cost calculators and the Azure Advisor tool to identify waste or misconfigurations.

Lastly, document the migration process thoroughly. This includes technical decisions, configurations, and fallback plans. Share learnings across teams and capture lessons that will be useful for future migrations or similar projects in the organization.

ADVANCED SERVICES

IoT Hub (2 paragraphs)

IoT Hub is a fully managed Azure service that enables secure communication between IoT devices and the cloud. It supports bi-directional messaging, device identity management, and fine-grained control over device provisioning and access. It's ideal for scenarios like smart homes, industrial equipment monitoring, or connected vehicles.

Messages from devices can be routed based on rules to other Azure services like Event Hubs, Storage, or Service Bus. For instance, telemetry from sensors can be processed by Stream Analytics, then visualized with Power BI. IoT Hub ensures reliable delivery, security via per-device credentials, and device twins for state synchronization.

Notification Hub (2 paragraphs)

Notification Hub is used for sending push notifications to mobile devices across platforms (iOS, Android, Windows, etc.). It can send personalized, scalable messages to millions of devices. You can use it to notify users of updates, promotions, or important alerts in real time.

It integrates with backend systems through APIs and supports templates for language or platform-specific customization. It's ideal for consumer-facing apps needing instant engagement, such as e-commerce apps, news services, or transportation apps. You can also segment audiences based on tags for targeted messaging.

Cognitive Services Set of APIs (2 paragraphs)

Azure Cognitive Services provides pre-built APIs that bring AI capabilities to your applications without requiring deep ML knowledge. These services include Computer Vision (image analysis), Text Analytics (sentiment and language detection), Speech Recognition, and Language Translation.

For example, you can use Face API for identity verification or OCR for scanning receipts. These APIs allow rapid prototyping and intelligent feature integration, like chatbots, document scanning, or smart search. They are often used in customer support, accessibility apps, and content moderation tools.