Azure-Specific Questions

What is Azure Active Directory (Azure AD), and how does it relate to cloud security?

Azure Active Directory (Azure AD, now called Microsoft Entra ID) is Microsoft's cloud-based identity and access management service providing authentication and authorization for Azure resources and Microsoft 365.

Core capabilities:

- **Identity management** - centralized user and group management, syncs with on-premises Active Directory via Azure AD Connect, guest user access for external collaboration (B2B), and consumer identity management (B2C).
- **Authentication** - supports modern authentication protocols (OAuth 2.0, OpenID Connect, SAML), multi-factor authentication (MFA) for enhanced security, passwordless authentication (FIDO2 keys, Windows Hello), and single sign-on (SSO) across applications.
- **Access control** - role-based access control (RBAC) for Azure resources, conditional access policies enforcing access requirements, Privileged Identity Management (PIM) for just-in-time admin access, and Identity Protection detecting and remediating identity risks.

Security features:

- **Conditional Access** - context-aware access decisions based on user, location, device, application, and risk level. Example: require MFA when accessing from untrusted networks, block access from specific geographic locations, or require compliant devices for sensitive applications.
- **Identity Protection** - uses machine learning detecting identity-based risks: leaked credentials, anonymous IP usage, atypical travel, and malware-linked IP addresses. Automatically triggers remediation (require password reset, block access) based on risk level.
- **Privileged Identity Management (PIM)** - just-in-time privileged access reducing standing admin permissions, time-bound role assignments, approval workflows for activation, audit logging of privileged operations, and access reviews ensuring admins still need elevated access.
- **MFA** - second authentication factor beyond password: SMS codes, phone calls, Microsoft Authenticator app, FIDO2 security keys, and enforced via conditional access policies.

Relation to cloud security: Azure AD is **foundational to Azure security** - every Azure resource access goes through Azure AD authentication, RBAC policies in Azure reference Azure AD identities, audit logs track Azure AD authentication and authorization, and compromised Azure AD credentials mean compromised Azure resources.

Identity as security perimeter - modern security focuses on identity not network, Azure AD enforces zero-trust principles, and strong Azure AD security directly translates to strong Azure security.

Integration with Azure services: Azure resources like VMs, storage accounts, databases use Azure AD for access control, managed identities for Azure resources eliminate credential management, and Azure AD application proxy provides secure remote access.

Best practices: Enforce MFA for all users especially administrators, implement conditional access policies for context-aware security, use PIM for admin access requiring justification and approval, enable Identity Protection automatically responding to risks, regularly review access ensuring least privilege, use managed identities for applications not service principals with secrets, monitor Azure AD sign-in logs for anomalies, enable Azure AD audit logging for compliance, and integrate with SIEM for advanced threat detection.

Azure AD security directly impacts overall Azure security posture making it critical focus area.

How do you secure Azure Virtual Machines (VMs)?

Securing Azure VMs requires multiple layers.

Network security:

- **Network Security Groups (NSGs)** - stateful firewall rules controlling inbound/outbound traffic: create NSG allowing only necessary ports (HTTPS 443, RDP 3389 from bastion only), deny internet access on management ports, apply NSGs at subnet and NIC levels for defense in depth.
- **Azure Bastion** - managed bastion service for secure RDP/SSH: eliminates public IPs on VMs, RDP/SSH through Azure portal over TLS, no need to manage bastion hosts, and comprehensive session logging.
- **Private endpoints** - use for PaaS services accessed from VMs keeping traffic on Microsoft backbone not internet.
- **Just-in-time (JIT) VM access** - Security Center feature providing time-limited port access: RDP/SSH ports closed by default, users request access with justification, access granted for specific time period (1-24 hours), automatically revokes after expiration, and logs all access requests.

Access control:

- **Azure AD integration** - use Azure AD for VM authentication: Azure AD login for Windows and Linux VMs, centralized identity management, MFA for VM access, and eliminates local account management.
- **Managed identities** - VMs use managed identities accessing Azure resources: no credentials in code or configuration files, automatic credential rotation, and assign only necessary permissions following least privilege.
- **RBAC** - control who can manage VMs: separate permissions for VM start/stop vs. full management, require approval for sensitive operations.

Encryption:

- **Disk encryption** - Azure Disk Encryption (ADE) using BitLocker (Windows) or dm-crypt (Linux): encrypts OS and data disks at rest, keys managed in Azure Key Vault, protects against unauthorized access if disks stolen, enable with:

```
az vm encryption enable --resource-group RG --name VM --disk-encryption-keyvault
KEY_VAULT
```

- **Encryption at host** - additional layer encrypting temp disks and OS disk cache.

Vulnerability management:

- **Update management** - automate OS and application patching: Azure Automation Update Management schedules patches, assess update compliance, deploy critical updates on schedule, and reboot if needed during maintenance windows.
- **Microsoft Defender for Servers** - advanced threat protection for VMs: vulnerability assessment scanning for CVEs, adaptive application controls (allowlisting), file integrity monitoring detecting unauthorized changes, just-in-time network access, and security alerts on suspicious activity.
- **Azure Security Center** - provides security recommendations: unpatched VMs, missing antimalware, weak NSG rules, and prioritized remediation guidance.

Malware protection: **Microsoft Antimalware** - built-in antimalware for Azure VMs: real-time protection, scheduled scanning, malware remediation, configurable exclusions, and monitoring and alerting. **Endpoint protection** - or third-party endpoint security solutions integrated with Security Center.

Backup and disaster recovery: **Azure Backup** - automated VM backups: application-consistent backups, encrypted at rest and in transit, long-term retention, and quick restore capabilities. **Azure Site Recovery** - disaster recovery as a service: replicate VMs to secondary region, automated failover and failback, and regular DR drills.

Monitoring and logging: **Azure Monitor** - collect VM metrics and logs: CPU, memory, disk usage, application logs, and security events. **Boot diagnostics** - screenshot and serial console for troubleshooting. **Microsoft Sentinel** - SIEM integration for advanced threat detection correlating VM logs with other Azure logs.

Configuration management: **Azure Policy** - enforce VM configurations: require encryption, mandate antimalware, enforce allowed VM SKUs, and deny creation without NSG. **Azure Automation State Configuration** - ensure VMs maintain desired configuration using DSC preventing drift.

Hardening: **CIS benchmarks** - configure VMs following CIS benchmarks for OS hardening, Security Center assesses compliance, and remediate findings. **Disable unnecessary services** - minimize attack surface, remove unused software, and configure secure baselines.

Example secure VM deployment:

```
# Create NSG
```

```
az network nsg create --resource-group SecureRG --name SecureNSG

# Add inbound rule allowing HTTPS only
az network nsg rule create \
  --resource-group SecureRG \
  --nsg-name SecureNSG \
  --name AllowHTTPS \
  --priority 100 \
  --source-address-prefixes '*' \
  --destination-port-ranges 443 \
  --access Allow \
  --protocol Tcp

# Create VM with managed identity and encryption
az vm create \
  --resource-group SecureRG \
  --name SecureVM \
  --image UbuntuLTS \
  --admin-username azureuser \
  --assign-identity \
  --nsg SecureNSG \
  --public-ip-address "" \
  --encryption-at-host \
  --security-type TrustedLaunch

# Enable Azure AD login
az vm extension set \
  --publisher Microsoft.Azure.ActiveDirectory \
  --name AADSSHLoginForLinux \
  --resource-group SecureRG \
  --vm-name SecureVM

# Enable disk encryption
az vm encryption enable \
  --resource-group SecureRG \
  --name SecureVM \
  --disk-encryption-keyvault SecureKeyVault
```

This comprehensive approach protects Azure VMs from common attack vectors through defense in depth.

Explain Azure Security Center and its key features.

Azure Security Center (now **Microsoft Defender for Cloud**) is unified security management and threat protection platform for Azure, hybrid, and multi-cloud environments.

Key features:

- **Secure Score** - numerical representation (0-100%) of security posture: aggregates security recommendations by severity, tracks improvement over time, compares against benchmarks, and prioritizes remediation by impact on score.
- **Security recommendations** - actionable guidance improving security: identifies misconfigurations (unencrypted storage, missing MFA, weak NSGs), provides step-by-step remediation, categorizes by severity and effort, and some recommendations offer quick fixes (one-click remediation).
- **Threat protection** - advanced detection across resource types:
  - **Defender for Servers** provides threat detection for VMs (fileless attack detection, suspicious PowerShell, lateral movement indicators), vulnerability assessment scanning for CVEs, adaptive application controls limiting executable apps, file integrity monitoring detecting unauthorized changes, just-in-time VM access reducing attack surface.
  - **Defender for Storage** detects unusual data access patterns, potential malware uploads to storage accounts, anonymous access anomalies, and data exfiltration attempts.
  - **Defender for SQL** identifies SQL injection attempts, vulnerable database configurations, unusual data access, and suspicious login patterns.
  - **Defender for Kubernetes** detects container vulnerabilities, suspicious pod deployments, privilege escalation attempts, and cryptocurrency mining.
  - **Defender for App Service** protects web applications identifying code injection attacks, malicious file uploads, communication with malicious domains, and vulnerable application configurations.
- **Regulatory compliance dashboard** - tracks compliance with standards: Azure Security Benchmark, PCI DSS, ISO 27001, HIPAA, SOC TSP, and custom standards. Shows compliance percentage per standard, identifies non-compliant resources, and provides remediation guidance.
- **Integrated with Azure Policy** - enforces security policies: audit mode identifies non-compliance, deny mode prevents non-compliant resource creation, and custom policies for organization-specific requirements.
- **Multi-cloud and hybrid support** - protects resources beyond Azure: onboards AWS and GCP accounts showing unified security posture, protects on-premises servers via Azure Arc, and centralizes security across environments.
- **Security alerts** - real-time notifications of threats: alerts include severity, affected resources, attack timeline, recommended response actions, and integration with Microsoft Sentinel for SOAR.
- **Automation and orchestration**: Workflow automation responds to alerts triggering Logic Apps, automatic remediation fixes issues without manual intervention, and integration with ServiceNow or Jira for ticketing.
- **Vulnerability assessment** - built-in scanning: Qualys integration for VM vulnerability scanning, container image scanning in Azure Container Registry, SQL vulnerability assessment.
- **Adaptive security**: **Adaptive application controls** - ML-based allowlisting for VMs, identifies safe applications to allow, blocks unknown executables. **Adaptive network hardening** - analyzes traffic patterns recommending NSG improvements: suggests tightening overly

permissive rules, identifies unused inbound rules.

Tiers:

- **Free tier** - Secure Score, security recommendations, Azure Policy integration, and basic compliance dashboard.
- **Paid tier** - all threat protection features (Defender for Servers, Storage, SQL, etc.), advanced compliance dashboards, extended threat detection, and premium support.

Best practices: Enable Defender for Cloud on all subscriptions, review and remediate security recommendations regularly, prioritize by secure score impact, enable all relevant Defender plans (Servers, Storage, SQL, Containers), configure email notifications for high-severity alerts, implement workflow automation for common responses, regularly review compliance dashboard, conduct quarterly security reviews with Defender for Cloud findings, integrate with Azure Sentinel for advanced SIEM capabilities, and export security data to Log Analytics for long-term analysis.

Security Center/Defender for Cloud provides centralized security visibility and control essential for managing security at scale in Azure.

How does Azure DDoS Protection mitigate distributed denial-of-service attacks?

Azure DDoS Protection provides multi-layer defense against DDoS attacks.

Built-in protection (Basic) - automatic, always-on for all Azure resources at no extra cost: protects against common network layer attacks (SYN floods, UDP amplification, ICMP floods), leverages Azure's global scale absorbing attack traffic, monitors traffic using always-on monitoring and machine learning, automatically mitigates detected attacks without user intervention, and protects Azure platform itself benefiting all customers.

DDoS Protection Standard - enhanced protection with advanced features:

- **Adaptive tuning** - learns normal traffic patterns per application creating baselines tailored to your traffic, adapts thresholds based on application behavior, and reduces false positives.
- **Attack analytics** - detailed post-attack reports showing attack characteristics (volume, duration, sources, type), impact on resources, and mitigation effectiveness.
- **Always-on traffic monitoring** - inspects all traffic to and from public IPs detecting attacks within minutes.
- **Application layer protection** - integrates with Azure Application Gateway WAF protecting against L7 DDoS attacks (HTTP floods, Slowloris).
- **DDoS rapid response support** - dedicated support team during active attacks providing expert guidance, attack analysis, and custom mitigation rules.
- **Cost protection** - SLA-backed guarantee with credits for scaling costs incurred during documented DDoS attacks.

How mitigation works:

- **Detection** - continuous monitoring analyzes traffic to Azure public IPs, machine learning baselines normal traffic patterns, detects deviations indicating attacks (traffic spikes, unusual protocols, suspicious sources).
- **Mitigation** - scrubbing centers activate when attack detected filtering malicious traffic: drops attack packets before reaching target, allows legitimate traffic through to application, happens inline with minimal latency, and scales automatically handling Tbps-scale attacks.

Types of attacks mitigated:

- **Volumetric attacks** - flood network with traffic (UDP floods, amplification attacks): Azure's scale absorbs traffic, distributed scrubbing reduces load.
- **Protocol attacks** - exploit weaknesses in Layer 3/4 (SYN floods, fragmented packet attacks): stateful packet inspection identifies malicious patterns.
- **Resource layer attacks** - target application layer (HTTP floods, DNS query floods): WAF integration filters malicious requests, rate limiting prevents overwhelming backends.

Configuration: Enable DDoS Protection Plan on virtual network:

```
# Create DDoS Protection Plan
az network ddos-protection create \
  --resource-group SecurityRG \
  --name MyDDoSPlan

# Enable on VNet
az network vnet update \
  --resource-group SecurityRG \
  --name MyVNet \
  --ddos-protection-plan MyDDoSPlan \
  --ddos-protection true
```

Monitoring: Azure Monitor provides DDoS metrics: under DDoS attack (yes/no), inbound packets dropped, inbound TCP packets mitigated, inbound UDP packets mitigated. Set up alerts: alert when under DDoS attack begins, notification when mitigation completes. View in Azure portal DDoS Protection dashboard showing real-time and historical attack data.

Integration with Azure services: Works with Application Gateway providing L7 protection, Azure Front Door for global load balancing with DDoS protection, Load Balancer for L4 traffic distribution, and Public IP addresses (Standard SKU required for DDoS Protection Standard).

Best practices: Enable DDoS Protection Standard for production workloads especially internet-facing, configure diagnostic logs forwarding to Log Analytics or Storage, set up alerts for DDoS attack detected, combine with WAF for application layer protection, design architecture for high availability complementing DDoS protection, regularly review DDoS protection logs and metrics, conduct DDoS simulation testing (coordinate with Azure), document DDoS response procedures, and maintain contact list for DDoS Rapid Response team.

Limitations: Protects only Azure public IPs (not applicable to VMs in VNets without public IPs), focuses on volumetric and protocol attacks (application-specific attacks need WAF), effectiveness

depends on architecture (distributed, resilient applications benefit most).

Azure DDoS Protection leverages Microsoft's global infrastructure providing enterprise-grade DDoS defense that individual organizations couldn't achieve alone.

What is Azure Key Vault, and how does it manage cryptographic keys?

Azure Key Vault is cloud-based secrets management service securely storing and managing cryptographic keys, secrets, and certificates.

Core capabilities:

- **Key management** - create, import, and control cryptographic keys (RSA, EC keys), keys protected by FIPS 140-2 validated HSMs, software-protected or HSM-protected keys, key operations (encrypt, decrypt, sign, verify) performed within Key Vault, keys never exposed to applications.
- **Secret management** - store application secrets (connection strings, API keys, passwords), version control tracking secret changes, access control via Azure AD and RBAC, audit logging of all secret access.
- **Certificate management** - automate SSL/TLS certificate lifecycle, integrate with certificate authorities (DigiCert, GlobalSign), automatic renewal before expiration, store certificates securely with private keys.

Types of keys:

- **Software-protected keys** - stored and protected by software-based mechanisms, FIPS 140-2 Level 1 validated, suitable for most scenarios, lower cost than HSM-protected.
- **HSM-protected keys** - stored in Hardware Security Modules, FIPS 140-2 Level 2 validated (Premium tier), keys never leave HSM boundary in plaintext, required for high-security or compliance scenarios, higher cost but maximum key protection.
- **Managed HSM** - dedicated single-tenant HSM pool, FIPS 140-2 Level 3 validated, complete control over HSMs, suitable for strictest regulatory requirements (banking, government).

Key operations: Applications don't retrieve keys directly - instead call Key Vault APIs for crypto operations:

- **Encrypt/Decrypt** - Key Vault performs encryption with key, ciphertext returned to application, decryption happens inside Key Vault.
- **Sign/Verify** - create digital signatures, verify signature authenticity.
- **Wrap/Unwrap** - envelope encryption pattern where data encryption key wrapped by Key Vault key.
- **Import/Export** - import existing keys (including HSM-to-HSM transfer), export public keys only (private keys never exportable from HSM).

Access control:

- **Azure AD integration** - all access authenticated via Azure AD, users, groups, service principals, and managed identities.
- **RBAC permissions** - Key Vault Administrator, Key Vault Secrets Officer, Key Vault Crypto Officer, Key Vault Reader, and granular permissions (keys/get, secrets/set, certificates/list).
- **Access policies** (classic model) - specify which principals can perform which operations, separate permissions for keys, secrets, certificates.
- **Network security** - firewall rules restricting access to specific VNets or IP ranges, private endpoints keeping traffic on Microsoft network, disable public access entirely.

Key rotation:

- **Automatic rotation** - configure rotation policy for keys and secrets:

  ```
  az keyvault key rotation-policy update --vault-name MyVault --name MyKey --value
  rotation-policy.json
  ```

  Policy specifies rotation frequency and notification timing.

- **Manual rotation** - create new key version, update applications to use new version, disable or delete old version after transition.
- **Versioning** - each rotation creates new version, previous versions retained for decryption of old data, applications can specify version or use latest.

Integration with Azure services: **Azure Disk Encryption** - encrypts VM disks using keys from Key Vault, **Storage Service Encryption** - customer-managed keys for storage accounts, **SQL TDE** - Transparent Data Encryption with Key Vault keys, **Managed identities** - Azure resources access Key Vault without credentials, **Azure DevOps** - secrets stored in Key Vault referenced in pipelines.

Monitoring and logging: **Diagnostic logging** - logs all Key Vault operations to Azure Monitor, Storage Account, Event Hub, or Log Analytics. Captures who accessed which key/secret when, operation type (get, encrypt, decrypt), success or failure, and source IP address. **Alerts** - notify on unusual activity: failed authentication attempts, key/secret deletion, access from unexpected locations, or specific operations (export key).

Backup and recovery:

- **Soft delete** - deleted keys/secrets retained for recovery period (7-90 days):

  ```
  az keyvault update --name MyVault --enable-soft-delete true --retention-days 90
  ```

  Allows recovery of accidentally deleted items.

- **Purge protection** - prevents permanent deletion during retention period, even by administrators, required for certain compliance scenarios.

- **Backup/restore** - export keys/secrets to encrypted blob for disaster recovery, restore to same or different Key Vault.

Best practices: Enable soft delete and purge protection on all vaults, use managed identities for application access eliminating secrets, implement least privilege access policies, use HSM-backed keys for sensitive data or compliance requirements, enable diagnostic logging forwarding to centralized location, monitor for unauthorized access attempts, rotate keys regularly per security policy, use separate Key Vaults for different environments (dev/test/prod), implement network restrictions allowing only necessary access, regularly audit Key Vault access and permissions, use private endpoints for production workloads, and test disaster recovery procedures.

Example: Using Key Vault with managed identity:

```
# Create Key Vault
az keyvault create \
  --name SecureVault \
  --resource-group SecurityRG \
  --location eastus \
  --enable-soft-delete true \
  --enable-purge-protection true \
  --sku premium

# Create HSM-protected key
az keyvault key create \
  --vault-name SecureVault \
  --name EncryptionKey \
  --protection hsm \
  --size 2048 \
  --kty RSA

# Grant VM managed identity access to key
az keyvault set-policy \
  --name SecureVault \
  --object-id <VM_MANAGED_IDENTITY_ID> \
  --key-permissions encrypt decrypt
```

Application code using managed identity:

```
using Azure.Identity;
using Azure.Security.KeyVault.Keys.Cryptography;

// Authenticate with managed identity
var credential = new DefaultAzureCredential();
var keyClient = new CryptographyClient(
    new Uri("https://securevault.vault.azure.net/keys/EncryptionKey"),
    credential);

// Encrypt data
byte[] plaintext = Encoding.UTF8.GetBytes("sensitive data");
```

```
var encryptResult = await keyClient.EncryptAsync(
    EncryptionAlgorithm.RsaOaep, plaintext);
byte[] ciphertext = encryptResult.Ciphertext;
```

Key Vault provides enterprise-grade key management essential for encryption, secrets management, and compliance in Azure.

Describe the Azure Monitor and Azure Sentinel services in security monitoring.

Azure Monitor provides comprehensive observability across Azure resources.

For security:

- **Log Analytics workspace** - centralized log repository collecting security-relevant logs: Azure Activity Logs (management plane operations), Resource Logs (resource-specific logs - NSG flow logs, Key Vault access, SQL audit), Azure AD logs (sign-ins, audit), and application logs from VMs.
- **Kusto Query Language (KQL)** - powerful query language analyzing logs: `SecurityEvent | where EventID == 4625 | summarize count() by Account` finds failed login attempts by account.
- **Alerts** - trigger on security conditions: unusual number of failed authentications, specific security events, resource configuration changes, and anomalous activity patterns.
- **Workbooks** - visualize security data with interactive reports showing security posture over time, compliance status, and incident trends.
- **Integration** - connects with Azure Security Center, Microsoft Defender for Cloud, and third-party SIEM solutions.

Azure Sentinel is cloud-native SIEM and SOAR (Security Orchestration, Automated Response) solution.

Key capabilities:

- **Data connectors** - ingest data from multiple sources: Azure services (Activity Logs, Azure AD, Microsoft Defender), Microsoft 365 (Office 365, Microsoft Defender for Endpoint), third-party solutions (Palo Alto, Check Point, AWS CloudTrail), custom sources via REST API or Syslog/CEF.
- **Analytics rules** - detect threats using queries and ML:
  - **Scheduled rules** - KQL queries running periodically detecting patterns (multiple failed logins followed by success indicating brute force compromise).
  - **Microsoft security** - ingest alerts from Defender for Cloud, Defender for Endpoint.
  - **Fusion** - ML-based correlation detecting multi-stage attacks.
  - **Anomaly rules** - behavioral analytics identifying unusual user/entity behavior.
- **Incidents** - aggregate related alerts into manageable cases: automatically correlate alerts into incidents, assign ownership and severity, track investigation status, and provide timeline of related events.

- **Investigation graph** - visual representation showing entity relationships: connections between users, hosts, IPs, activities, helps understand attack scope and lateral movement.
- **Hunting** - proactive threat hunting using queries: built-in hunting queries for common threats, custom queries for organization-specific threats, bookmarks saving interesting findings, and livestream for real-time query results.
- **Automation and orchestration (SOAR)**:
  - **Playbooks** - automated response workflows using Azure Logic Apps: automatically isolate compromised VM, block malicious IP in firewall, send notification to security team, create ServiceNow ticket, and gather forensic evidence.
  - **Automation rules** - simpler automation for common tasks: auto-assign incidents based on criteria, auto-close false positives, escalate high-severity incidents.
- **Watchlists** - reference data for enrichment: VIP users requiring extra monitoring, known malicious IPs, approved admin accounts, and custom threat intelligence.
- **UEBA (User and Entity Behavioral Analytics)** - ML-based anomaly detection: establishes baseline normal behavior for users and entities, detects deviations indicating compromise (user accessing unusual resources, abnormal data download volume, login from atypical location/time), and risk scoring prioritizing investigation.
- **Threat intelligence integration** - enrich data with threat intel: Microsoft Threat Intelligence feed, custom threat intel feeds, TAXII/STIX feeds, and automatic indicator matching in logs.

Security monitoring workflow: Logs flow from sources → Sentinel data connectors ingest logs → Analytics rules evaluate logs detecting threats → Incidents created from alerts → Security analyst investigates using investigation graph and queries → Playbook automates response (isolate VM, block IP) → Incident closed with documentation → Metrics tracked showing MTTD (mean time to detect) and MTTR (mean time to respond).

Example analytics rule (KQL):

```
// Detect successful login after multiple failures (possible brute force)
let threshold = 5;
let timeframe = 1h;
SigninLogs
| where TimeGenerated > ago(timeframe)
| where ResultType != 0  // Failed sign-ins
| summarize FailedAttempts = count() by UserPrincipalName, IPAddress,
bin(TimeGenerated, 5m)
| where FailedAttempts >= threshold
| join kind=inner (
    SigninLogs
    | where TimeGenerated > ago(timeframe)
    | where ResultType == 0  // Successful sign-in
) on UserPrincipalName, IPAddress
| where TimeGenerated1 > TimeGenerated
| project-away TimeGenerated1
| extend AccountCustomEntity = UserPrincipalName, IPCustomEntity = IPAddress
```

Example playbook (Logic App): Trigger: Sentinel incident created → Condition: Severity is High → Action: Get VM details → Action: Isolate VM (remove from NSG) → Action: Create snapshot for forensics → Action: Send email to security team → Action: Create Jira ticket → Action: Update incident status.

Best practices: Deploy Sentinel in dedicated subscription for cost management and isolation, enable all relevant data connectors for comprehensive visibility, start with Microsoft-provided analytics rules then customize, implement playbooks for common response scenarios, regularly review and tune analytics rules reducing false positives, use watchlists for context enrichment, enable UEBA for advanced behavioral analytics, integrate threat intelligence feeds, conduct regular threat hunting exercises, track MTTD and MTTR metrics optimizing response, and train security team on KQL and Sentinel features.

Cost management: Sentinel charges based on data ingested - filter unnecessary logs before ingestion, use tiered pricing for predictable costs, set daily cap preventing runaway costs, archive old data to Log Analytics or Storage, and regularly review data usage optimizing connectors.

Azure Monitor provides foundational logging and alerting while Sentinel adds advanced SIEM/SOAR capabilities for enterprise security operations at scale.

How do you implement network security groups (NSGs) in Azure?

NSGs provide network-level access control for Azure resources acting as stateful firewalls.

NSG basics: NSG contains security rules defining allowed/denied traffic, rules evaluated by priority (lower number = higher priority, 100-4096), default rules (priority 65000+) allowing VNet traffic and denying internet, and stateful operation (return traffic automatically allowed).

Rule structure: Each rule specifies priority number, name, direction (inbound or outbound), action (allow or deny), protocol (TCP, UDP, ICMP, Any), source (IP address, CIDR, service tag, application security group), source port range, destination (IP, CIDR, service tag, ASG), and destination port range.

Creating and applying NSGs:

```
# Create NSG
az network nsg create \
  --resource-group SecurityRG \
  --name WebServerNSG

# Add rule allowing HTTPS from internet
az network nsg rule create \
  --resource-group SecurityRG \
  --nsg-name WebServerNSG \
  --name AllowHTTPS \
  --priority 100 \
  --source-address-prefixes Internet \
```

```
  --source-port-ranges '*' \
  --destination-address-prefixes '*' \
  --destination-port-ranges 443 \
  --access Allow \
  --protocol Tcp \
  --direction Inbound

# Deny all other inbound (explicit deny)
az network nsg rule create \
  --resource-group SecurityRG \
  --nsg-name WebServerNSG \
  --name DenyAllInbound \
  --priority 4096 \
  --source-address-prefixes '*' \
  --destination-address-prefixes '*' \
  --access Deny \
  --protocol '*' \
  --direction Inbound

# Associate NSG with subnet
az network vnet subnet update \
  --resource-group SecurityRG \
  --vnet-name MyVNet \
  --name WebSubnet \
  --network-security-group WebServerNSG
```

Application Security Groups (ASGs): Logical grouping of VMs for simplified NSG management: create ASG representing application tier (web-asg, app-asg, db-asg), associate VM NICs with ASGs, use ASGs as source/destination in NSG rules. Instead of managing individual IP addresses, rules reference ASGs: "Allow traffic from web-asg to app-asg on port 8080". When VMs added/removed from ASG, rules automatically apply.

Service tags: Predefined groups of IP addresses for Azure services: `Internet` (all internet IPs), `VirtualNetwork` (all VNet address space), `AzureLoadBalancer` (Azure LB health probes), `Storage` (Azure Storage IP ranges), `Sql` (Azure SQL Database), and regional variants (`Storage.EastUS`). Simplifies rules and automatically updates as Azure IP ranges change.

NSG flow logs: Capture information about traffic flowing through NSG: source/destination IP, ports, protocol, action (allowed/denied), flow direction, and packet/byte counts. Enable diagnostic logging:

+

```
az network watcher flow-log create --location eastus --name MyFlowLog --nsg
WebServerNSG --storage-account flowlogstorage --enabled true --retention 7
```

+ Analyze with Traffic Analytics for insights: top talkers, blocked traffic, geographic distribution, and anomalous traffic patterns.

Security best practices:

- **Default deny** - explicit deny rule at lowest priority ensuring nothing allowed unless specifically permitted.
- **Minimize inbound rules** - only allow required services, deny SSH/RDP from internet (use Bastion instead).
- **Separate tiers** - different NSGs for web, application, database tiers, use ASGs representing tiers in rules.
- **Use service tags** - instead of hardcoding IP ranges, leverage service tags automatically updating.
- **Logging** - enable NSG flow logs for all production NSGs, analyze with Traffic Analytics detecting anomalies.
- **Regular audits** - quarterly review of NSG rules removing unused permissions, ensure least privilege, and validate rules match security requirements.
- **Testing** - verify NSG rules work as expected using Network Watcher IP flow verify:

  ```
  az network watcher test-ip-flow --vm MyVM --direction Inbound --protocol TCP
  --local 10.0.0.4:443 --remote 203.0.113.25:12345
  ```

  Shows whether traffic allowed/denied and which rule made decision.

NSG effective security rules: When NSG applied at both subnet and NIC: subnet NSG rules evaluated first, then NIC NSG rules, most restrictive wins (if subnet allows but NIC denies, traffic denied). View effective rules: Azure Portal → VM → Networking → Effective security rules.

Example three-tier architecture:

```
# Web tier NSG - allows HTTPS from internet
az network nsg rule create --nsg-name WebNSG --name AllowHTTPS --priority 100 \
  --source-address-prefixes Internet --destination-port-ranges 443 --access Allow

# App tier NSG - allows traffic only from web tier
az network nsg rule create --nsg-name AppNSG --name AllowFromWeb --priority 100 \
  --source-address-prefixes 10.0.1.0/24 --destination-port-ranges 8080 --access Allow

# Database tier NSG - allows traffic only from app tier
az network nsg rule create --nsg-name DbNSG --name AllowFromApp --priority 100 \
  --source-address-prefixes 10.0.2.0/24 --destination-port-ranges 1433 --access Allow
```

NSGs provide fundamental network security in Azure, essential for implementing defense in depth and network segmentation.

What are the security implications of Azure Functions, and how can they be addressed?

Azure Functions serverless architecture introduces specific security considerations.

Security implications:

- **Broad IAM permissions** - Functions often granted excessive permissions during development: over-permissioned managed identities accessing more resources than necessary, shared function app-level identity instead of per-function granularity.
- **Code vulnerabilities** - injection attacks, insecure dependencies, exposed secrets in code.
- **Trigger security** - HTTP triggers without authentication publicly accessible, queue/blob triggers processing untrusted data.
- **Data exposure** - logging sensitive data in Application Insights, connection strings in configuration.
- **Dependencies** - vulnerable npm/pip packages, outdated runtime versions.

Addressing security:

Authentication and authorization:

- **Require authentication** on HTTP triggers: Function-level keys, Azure AD authentication, API Management frontend, or custom authentication in code. Configure authentication: Azure Portal → Function App → Authentication → Add identity provider (Microsoft, Google, Facebook).
- **Managed identities** - eliminate connection strings and keys: system-assigned identity unique to function app, user-assigned identity shared across resources, grant identity minimal permissions to required resources (Storage, Key Vault, SQL).

Example accessing Key Vault with managed identity:

```
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;

var client = new SecretClient(
    new Uri("https://myvault.vault.azure.net"),
    new DefaultAzureCredential()); // Uses managed identity
var secret = await client.GetSecretAsync("ConnectionString");
```

Network security:

- **VNet integration** - functions connect to resources via private network: integrate function app with VNet, access resources via private endpoints, egress traffic routes through VNet.
- **Private endpoints** - make function app accessible only via private IP: disables public access, access via VNet or ExpressRoute/VPN, combines with firewall rules for IP restrictions.
- **IP restrictions** - allow specific IPs accessing function app: corporate IP ranges, Azure services,

partner networks.

Secrets management:

- **Never hardcode secrets** - use Application Settings stored encrypted at rest, reference Key Vault secrets: `@Microsoft.KeyVault(SecretUri=https://myvault.vault.azure.net/secrets/DbPassword)`, automatic secret retrieval using managed identity.
- **Environment variables** - secrets exposed as environment variables to function: `Environment.GetEnvironmentVariable("SECRET_NAME")`, not visible in code or source control.

Code security:

- **Input validation** - validate and sanitize all inputs: HTTP request bodies, queue messages, blob content, prevents injection attacks.
- **Dependency scanning** - regularly scan dependencies for vulnerabilities: npm audit, pip check, Dependabot integration in GitHub.
- **SAST** - static analysis in CI/CD pipeline detecting code vulnerabilities before deployment.
- **Least privilege** - managed identity with minimum necessary permissions: read-only where possible, scoped to specific resources.
- **Runtime version** - keep runtime updated: use latest LTS versions, monitor for security patches.

Monitoring and logging: **Application Insights** - comprehensive telemetry: request traces, exceptions, dependencies, custom events. **Sanitize logs** - ensure sensitive data not logged: PII, credentials, financial data. **Alerts** - notify on security events: authentication failures, unusual invocation patterns, error spikes. **Azure Sentinel integration** - forward logs to Sentinel for SIEM analysis.

Secure configuration: **Deployment slots** - test security configurations before production: validate authentication, network restrictions in staging, swap to production when verified. **Deployment credentials** - use deployment tokens not publishing profiles, rotate regularly, SCM access restrictions separate from function access.

Example secure function configuration:

```
# Create function app with managed identity
az functionapp create \
  --name SecureFunction \
  --resource-group SecurityRG \
  --storage-account funcstorage \
  --runtime dotnet \
  --runtime-version 6 \
  --assign-identity [system]

# Enable VNet integration
az functionapp vnet-integration add \
  --name SecureFunction \
  --resource-group SecurityRG \
  --vnet MyVNet \
```

```
  --subnet FunctionSubnet

# Configure authentication
az functionapp auth update \
  --name SecureFunction \
  --resource-group SecurityRG \
  --enabled true \
  --action LoginWithAzureActiveDirectory

# Add IP restrictions
az functionapp config access-restriction add \
  --name SecureFunction \
  --resource-group SecurityRG \
  --rule-name AllowCorporate \
  --action Allow \
  --ip-address 203.0.113.0/24 \
  --priority 100
```

Best practices: Require authentication on all HTTP triggers, use managed identities for Azure resource access, store secrets in Key Vault referenced via application settings, implement network restrictions limiting function access, enable VNet integration for accessing private resources, scan dependencies regularly for vulnerabilities, validate and sanitize all inputs, minimize managed identity permissions, enable Application Insights with sensitive data sanitized, keep runtime and dependencies updated, use deployment slots for security testing, implement comprehensive logging and monitoring, and conduct regular security reviews of function code and configuration.

Azure Functions security requires careful attention to identity, network access, secrets management, and code security throughout the development lifecycle.

How can you secure Azure Blob Storage and Azure SQL Database?

Securing Azure Blob Storage:

- **Access control** - Azure AD integration for identity-based access: assign built-in roles (Storage Blob Data Reader, Contributor, Owner), use managed identities for applications, avoid shared key access for better auditability.
- **Shared Access Signatures (SAS)** - time-limited delegated access: account-level or service-level SAS, specify permissions, IP restrictions, protocol (HTTPS only), and expiration.
- **Public access prevention** - disable anonymous access: account-level setting preventing public containers, enables compliance with security policies.
- **Encryption** - data encrypted at rest by default using Microsoft-managed keys, customer-managed keys in Key Vault for control, double encryption for additional security.
- **Network security** - firewall rules allowing specific IP ranges or VNets, private endpoints for access via private IP, disable public access entirely for highly sensitive data.

- **Versioning and soft delete** - versioning retains previous blob versions, soft delete recovers deleted blobs within retention period (7-365 days), protects against accidental deletion and ransomware.
- **Immutable storage** - WORM (Write Once, Read Many) policies: time-based retention preventing deletion/modification until expiration, legal hold for compliance/litigation, combined with versioning for comprehensive protection.

Example secure Blob Storage:

```
# Create storage account with secure defaults
az storage account create \
  --name securestorage \
  --resource-group SecurityRG \
  --sku Standard_GRS \
  --encryption-services blob \
  --https-only true \
  --min-tls-version TLS1_2 \
  --allow-blob-public-access false

# Enable soft delete
az storage blob service-properties delete-policy update \
  --account-name securestorage \
  --enable true \
  --days-retained 30

# Configure firewall
az storage account network-rule add \
  --account-name securestorage \
  --ip-address 203.0.113.0/24

# Create private endpoint
az network private-endpoint create \
  --name BlobPrivateEndpoint \
  --resource-group SecurityRG \
  --vnet-name MyVNet \
  --subnet PrivateSubnet \
  --private-connection-resource-id /subscriptions/.../securestorage \
  --group-id blob \
  --connection-name BlobConnection
```

Securing Azure SQL Database:

- **Authentication** - Azure AD authentication (preferred over SQL authentication): centralized identity management, MFA support, managed identities for applications, no passwords in connection strings.
- **Network security** - firewall rules restricting client IPs, VNet rules allowing specific subnets, private endpoints for private connectivity, disable public endpoint for maximum security.
- **Encryption** - TDE (Transparent Data Encryption) enabled by default: encrypts database files at

rest, customer-managed keys in Key Vault optional, always encrypted for column-level encryption protecting data from DBAs.

- **Dynamic data masking** - obfuscates sensitive data in query results: mask credit cards, SSNs, emails, custom masking rules, doesn't change stored data.
- **Row-level security** - filters rows based on user context: users see only their own data, implemented via security predicates.
- **Auditing** - tracks database events: all queries, schema changes, permission changes, logs to Storage Account, Event Hubs, or Log Analytics.
- **Threat detection** - Microsoft Defender for SQL: identifies SQL injection attempts, unusual data access patterns, potential vulnerabilities, brute force attacks.
- **Backup and recovery** - automated backups with point-in-time restore, geo-redundant backups for DR, long-term retention for compliance.

Example secure Azure SQL:

```
# Create SQL server with AD admin
az sql server create \
  --name securesqlserver \
  --resource-group SecurityRG \
  --location eastus \
  --admin-user sqladmin \
  --admin-password <secure-password> \
  --enable-ad-only-auth \
  --external-admin-principal-type User \
  --external-admin-name [email protected] \
  --external-admin-sid <AAD-USER-SID>

# Configure firewall (deny all, add specific)
az sql server firewall-rule create \
  --resource-group SecurityRG \
  --server securesqlserver \
  --name AllowCorporate \
  --start-ip-address 203.0.113.1 \
  --end-ip-address 203.0.113.254

# Enable auditing
az sql server audit-policy update \
  --resource-group SecurityRG \
  --name securesqlserver \
  --state Enabled \
  --storage-account secureauditstorage

# Enable Advanced Threat Protection
az sql server threat-policy update \
  --resource-group SecurityRG \
  --name securesqlserver \
  --state Enabled \
```

```
  --email-account-admins Enabled
```

Best practices: Use Azure AD authentication eliminating SQL passwords, implement network restrictions with firewall rules or private endpoints, enable auditing and threat detection for monitoring, use TDE with customer-managed keys for sensitive data, implement dynamic data masking for PII, enable automated backups with appropriate retention, use managed identities for application database access, regularly review and minimize database permissions, monitor for unusual access patterns, keep SQL Server and database compatibility level updated, and conduct periodic security assessments.

Both services benefit from defense in depth combining multiple security controls for comprehensive protection.

What is Azure Bastion, and how does it enhance security in Azure?

Azure Bastion is fully managed PaaS service providing secure RDP/SSH connectivity to Azure VMs without exposing them via public IPs.

How it works: Bastion deployed in VNet on dedicated subnet (AzureBastionSubnet /26 or larger), users connect to VMs through Azure Portal over HTTPS (443), Bastion proxies RDP/SSH connection to target VM via private IP, VM doesn't need public IP or special agent.

Security benefits:

- **No public IP exposure** - VMs remain completely private without internet-facing endpoints: eliminates attack surface for RDP/SSH brute force, no need to manage NSG rules for bastion access, reduces internet exposure.
- **No bastion host management** - fully managed service eliminates maintaining and patching bastion VMs: Microsoft handles security updates, HA built-in across availability zones, no OS hardening needed.
- **Centralized access point** - single entry for all VM access: consistent access control via Azure RBAC, comprehensive session logging for audit, easier to secure than multiple entry points.
- **Protocol hardening** - RDP/SSH over TLS 443: encrypted with TLS preventing eavesdropping, standard HTTPS port typically allowed through corporate firewalls, no custom ports or protocols.
- **Integration with Azure AD** - authenticate users via Azure AD: MFA enforcement for VM access, conditional access policies (device compliance, location, risk), just-in-time access via PIM.
- **Session recording** - comprehensive audit trail: all bastion sessions logged, recording can be enabled for compliance, tracks who accessed which VM when.

Deployment:

```
# Create bastion subnet
az network vnet subnet create \
```

```
  --resource-group SecurityRG \
  --vnet-name MyVNet \
  --name AzureBastionSubnet \
  --address-prefixes 10.0.255.0/26

# Create public IP for bastion
az network public-ip create \
  --resource-group SecurityRG \
  --name BastionPublicIP \
  --sku Standard \
  --location eastus

# Deploy Azure Bastion
az network bastion create \
  --name MyBastion \
  --resource-group SecurityRG \
  --vnet-name MyVNet \
  --public-ip-address BastionPublicIP \
  --location eastus
```

Access workflow: User navigates to VM in Azure Portal → clicks "Connect" → selects "Bastion" → enters VM credentials or SSH key → bastion establishes session → user interacts with VM via browser.

NSG requirements: Bastion subnet NSG must allow: inbound 443 from Internet (user connections), inbound 443 from GatewayManager (control plane), outbound 443/22 to VirtualNetwork (VM connections), outbound 443 to AzureCloud (logging). Target VM NSG must allow: inbound 3389 (RDP) or 22 (SSH) from bastion subnet.

Features: **Native client support** - connect using native RDP/SSH clients instead of browser, better performance for intensive sessions. **Copy/paste** - clipboard integration for text between local machine and VM. **Shareable links** - generate link for support access without portal login. **IP-based connection** - connect to VMs via private IP not just VM name. **Multiple VM support** - single bastion serves all VMs in VNet (or peered VNets).

SKUs: **Basic** - fundamental connectivity, browser-based only, up to 25 concurrent sessions. **Standard** - all basic features plus native client support, shareable links, IP-based connections, higher session capacity (up to 100+), and host scaling.

Comparison to alternatives: **VPN** - Bastion eliminates VPN client management, no split-tunneling concerns, easier for occasional admin access. **Jump box** - Bastion eliminates VM management overhead, automatic HA and patching, no OS licensing costs. **Public IP + NSG** - Bastion removes internet exposure, better audit capabilities, easier access control.

Best practices: Deploy bastion in all production VNets with VMs, use Standard SKU for native client support and better performance, enable diagnostic logging capturing connection logs, implement just-in-time access via PIM for bastion access, configure NSGs properly on bastion subnet and target VMs, use separate bastion for different security zones if needed, monitor bastion usage and sessions, combine with Azure AD conditional access policies, regularly review bastion access permissions, and test connectivity before emergency situations.

Limitations: Requires dedicated subnet (cannot share with other resources), incurs hourly cost plus outbound data transfer, slightly higher latency than direct RDP/SSH, limited to RDP and SSH protocols (no other protocols).

Despite limitations, Bastion significantly improves security posture by eliminating public VM exposure.

An Azure VM is showing signs of compromise. How would you isolate the VM, investigate the issue, and remediate it?

Immediate containment (0-15 minutes):

1. **Network isolation** - modify VM's NSG to deny all traffic:

   ```
   az network nsg rule create --resource-group RG --nsg-name VM-NSG --name DenyAll
   --priority 100 --access Deny --source-address-prefixes '*' --destination-address
   -prefixes '*'
   ```

   Preserves VM state for forensics while preventing attacker communication and lateral movement. Alternative: create new NSG with deny-all rules and swap.

2. **Tag for tracking** - apply tags indicating compromise:

   ```
   az vm update --resource-group RG --name CompromisedVM --set
   tags.SecurityIncident=IR-2026-001 tags.Status=Quarantined
   tags.IsolatedBy=SecurityTeam tags.IsolatedDate=2026-01-20`
   ```

3. **Notify stakeholders** - alert security team, application owners, and management of isolation.

Forensic preservation (15-45 minutes):

1. **Snapshot all disks** - capture current state before investigation: `az snapshot create --resource-group RG --name VM-OS-Snapshot-20260120 --source /subscriptions/.../Microsoft.Compute/disks/VM-OS-Disk`. Create snapshots of all attached disks.
2. **Memory dump** (if possible) - capture VM memory:

   ```
   az vm run-command invoke --resource-group RG --name CompromisedVM --command-id
   RunPowerShellScript --scripts "C:\Windows\System32\comsvcs.dll MiniDump <lsass-pid>
   C:\memdump.dmp full"
   ```

   Alternative: Linux using LiME.

3. **Export logs** - collect logs before potential loss: Azure Monitor logs, VM boot diagnostics, NSG

flow logs, Azure Activity Logs showing recent VM operations, and Application logs from VM.

Investigation (1-4 hours):

1. **Timeline reconstruction** - Azure Activity Log shows recent operations:

   ```
   az monitor activity-log list --resource-group RG --start-time 2026-01-19T00:00:00Z
   --query "[?contains(resourceId, 'CompromisedVM')]"
   ```

   Identify who accessed VM, configuration changes, extension installations.

2. **Analyze NSG flow logs** - identify suspicious connections: unusual outbound destinations (C2 servers), port scanning activity, large data transfers (exfiltration).
3. **Microsoft Defender for Servers** - review alerts and findings: check for malware detections, suspicious process execution, file integrity violations, and anomalous network connections.
4. **Forensic disk analysis** - mount snapshot to forensic workstation: create VM from snapshot in isolated VNet, analyze without booting compromised OS, examine file timestamps, registry changes (Windows), command history, persistence mechanisms (scheduled tasks, startup items), and malware artifacts.
5. **Log analysis** - Security Event Log (Windows) or auth.log (Linux): authentication attempts and successes, privilege escalation, new account creation, and unusual commands executed.
6. **Check for persistence** - common locations: Scheduled tasks/cron jobs, startup folders/rc.local, registry run keys (Windows), SSH authorized_keys, and web shells in web directories.

Determine scope (concurrent with investigation):

1. **Lateral movement check** - analyze if attacker accessed other resources: check for RDP/SSH from compromised VM to others, examine Azure AD sign-ins for stolen credentials usage, and review access to storage accounts, databases, Key Vault.
2. **Data access assessment** - determine what data attacker accessed: storage account access logs, database audit logs, Key Vault access logs, and identify sensitive data exposure.

Remediation (after investigation complete):

1. **Containment verification** - ensure attacker access terminated: rotated all credentials VM had access to, deleted any attacker-created accounts, removed backdoors/persistence mechanisms.
2. **VM recovery decision**:
   - **Option 1: Rebuild from known-good image** (preferred): Deploy new VM from trusted image or backup, reconfigure from infrastructure-as-code, migrate data from clean backup (verified pre-compromise), update credentials and certificates, thoroughly test before production.
   - **Option 2: Remediation in place** (if rebuild not feasible): Remove malware using antimalware tools, patch vulnerabilities that enabled compromise, reset all credentials, validate system integrity, extensive testing before returning to service.
3. **Hardening**: Apply CIS benchmarks, disable unnecessary services, implement application

allowlisting, deploy EDR agent, enhanced monitoring and alerting.

Recovery (staged approach):

1. **Validation environment** - restore to isolated VNet: thoroughly test functionality, security scanning for residual compromise, penetration testing.
2. **Production restoration**: Use blue-green deployment if possible, monitor intensively for 48-72 hours post-restoration, and maintain incident response readiness.

Post-incident (after recovery):

1. **Root cause analysis** - determine initial compromise vector: unpatched vulnerability, weak credentials, misconfiguration, or social engineering.
2. **Lessons learned**: What detection gaps existed, how to improve response time, what preventive controls could have stopped attack, and documentation updates.
3. **Improvements**: Patch vulnerabilities, enhance monitoring, deploy additional controls, security awareness training, and update incident response procedures.

Example isolation script:

```
#!/bin/bash
INCIDENT_ID="IR-2026-001"
RG="ProductionRG"
VM="CompromisedVM"
TIMESTAMP=$(date +%Y%m%d_%H%M%S)

echo "Isolating VM $VM..."
# Deny all network traffic
az network nsg rule create \
  --resource-group $RG \
  --nsg-name ${VM}-NSG \
  --name EmergencyIsolation \
  --priority 100 \
  --access Deny \
  --direction Inbound \
  --source-address-prefixes '*'

# Tag VM
az vm update --resource-group $RG --name $VM \
  --set tags.Incident=$INCIDENT_ID tags.Isolated=$TIMESTAMP

# Snapshot disks
DISKS=$(az vm show -g $RG -n $VM --query "storageProfile.osDisk.name" -o tsv)
az snapshot create --resource-group $RG \
  --name ${VM}-Snapshot-${TIMESTAMP} \
  --source /subscriptions/.../Microsoft.Compute/disks/$DISKS

echo "VM isolated. Snapshot created. Begin investigation."
```

This systematic approach ensures proper containment, preserves evidence, enables thorough investigation, and supports complete recovery while preventing similar future incidents.