**AZ-204 Exam Cram Guide – Part 1 (Compute)**

*(25–30% of the exam)*

**📚 Deep Review**

**Azure App Service (Web Apps)**

* **Hosting options**: Windows, Linux, Containers.
* **Scaling**:
  + **Scale up** = bigger SKU (more CPU/memory).
  + **Scale out** = more instances (manual or autoscale).
* **Deployment Slots**:
  + Support zero-downtime deployments.
  + Swap slots for blue/green releases.
  + Can direct % traffic for canary testing.
* **Authentication**: Built-in EasyAuth supports AAD, Google, Facebook, etc.
* **App Settings**: Stored in config, not code. Secrets best via **Key Vault references**.
* **Always On**: Keeps apps responsive (avoids idle cold starts).

**Azure Functions**

* **Hosting Plans**:
  + **Consumption**: Pay per execution. Auto-scale. Cold starts possible.
  + **Premium**: Pre-warmed instances, avoids cold starts, supports VNET.
  + **Dedicated (App Service Plan)**: Reserved capacity. Predictable cost.
* **Triggers**: HTTP, Queue, Blob, Event Hub, Event Grid, Timer, Cosmos DB, etc.
* **Bindings**: Declarative input/output to services (e.g., read from Blob, write to Queue).
* **Durable Functions** (stateful workflows):
  + **Patterns**:
    - *Chaining* – Run tasks sequentially.
    - *Fan-out/fan-in* – Run tasks in parallel, aggregate results.
    - *Async HTTP API* – Return 202 Accepted with status URL for polling.
    - *External events* – Wait for external approval/signal.
    - *Entity functions* – Manage state like actors.
* **Security**:
  + Use **Function Keys** for basic auth (not recommended).
  + Prefer **Azure AD + Managed Identity** for secure calls.

**Containers**

* **Azure Container Instances (ACI)**
  + Best for short-lived, single-container workloads.
  + Ephemeral (can start/stop fast, billed per second).
  + Integrates with VNET for private networking.
* **Azure Container Apps (ACA)**
  + Serverless container hosting.
  + Auto-scales (including to zero) using **KEDA** (event-driven).
  + Supports Dapr sidecars for microservice patterns.
* **Azure Kubernetes Service (AKS)**
  + Full Kubernetes cluster with Azure-managed control plane.
  + Custom networking, ingress, secrets, scaling.
  + Best for **complex microservices** with orchestration needs.

**Batch Processing**

* **Azure Batch**
  + Large-scale parallel compute workloads.
  + Supports Windows/Linux VMs, autoscaling pools.
  + Often used for data processing, image rendering, simulations.

**🛠 CLI / SDK Commands**

**App Service**

# Deploy a web app

az webapp up --name mywebapp --resource-group myRG --runtime "DOTNET:6"

**Functions**

# Create Function App (Consumption)

az functionapp create --resource-group myRG --consumption-plan-location westeurope \

--runtime dotnet --functions-version 4 --name myfuncapp --storage-account mystorage

**Containers**

# Run a container in ACI

az container create --resource-group myRG --name mycontainer \

--image mcr.microsoft.com/azuredocs/aci-helloworld --dns-name-label myappdemo

**⚠️ Gotchas & Exam Traps**

* App Service **Always On** only available in **Basic plan and above**.
* **Consumption Functions** may cold start → pick **Premium** for steady latency.
* Deployment **Slots** cost money only in Standard tier and above.
* **ACI** is for **ephemeral workloads**. Use **AKS** if you need orchestration.
* **Durable Functions** exam questions often test patterns — know *when* to use *fan-out/fan-in* vs *async API*.
* Don’t confuse **Event Grid** (event notifications) with **Event Hubs** (telemetry streaming).

**❓ Mini Practice Qs**

**Q1.** Which Functions hosting plan avoids cold starts while still scaling automatically?

* A) Consumption
* B) Premium
* C) Dedicated
* D) Free  
  ✅ **Answer**: B – Premium provides pre-warmed instances.

**Q2.** You must deploy a single-container app for 12 hours of testing, cheapest option?

* A) AKS
* B) ACI
* C) ACA
* D) App Service Linux  
  ✅ **Answer**: B – ACI is pay-per-second, best for temporary workloads.

**Q3.** Which Durable Functions pattern should you use when an API call must return 202 Accepted with a status endpoint?

* A) Fan-out/fan-in
* B) Chaining
* C) Async HTTP API
* D) External events  
  ✅ **Answer**: C – Async HTTP API pattern.

**AZ-204 Exam Cram Guide – Part 2 (Storage)**

*(10–15% of the exam)*

**📚 Deep Review**

**Azure Blob Storage**

* **Blob types**:
  + *Block blobs* → text/binary (most common).
  + *Append blobs* → optimized for logging.
  + *Page blobs* → random access, used for VHDs.
* **Access Tiers**: Hot, Cool, Archive.
  + Archive requires **rehydration** before access.
* **Data protection**:
  + *Soft Delete* → recover deleted/overwritten blobs.
  + *Versioning* → automatic version history.
  + *Immutable storage (Legal Hold / Time-based retention)* → WORM compliance.
* **Access Control**:
  + *SAS tokens* (time-limited, scoped).
  + *RBAC via Azure AD* (preferred).
  + *Stored access policies* (manage SAS centrally).

**Azure Queue Storage**

* Simple queue system for message storage/retrieval.
* Messages up to **64 KB**, unlimited queue size.
* At-least-once delivery, not FIFO.
* Best for lightweight async workflows.

**Azure Table Storage**

* NoSQL key-value store.
* PartitionKey + RowKey for fast lookup.
* Schema-less, cost-effective.
* Alternative: **Cosmos DB Table API** for global scale.

**Azure Cosmos DB**

* Globally distributed, multi-model NoSQL DB.
* **APIs**: SQL (default), MongoDB, Cassandra, Gremlin, Table.
* **Consistency models**:
  + *Strong* → linearizability, highest latency.
  + *Bounded staleness* → predictable lag (time or versions).
  + *Session* → default; read-your-writes in session.
  + *Consistent prefix* → no out-of-order reads.
  + *Eventual* → lowest latency, may be stale.
* **Partitioning**:
  + Pick high-cardinality partition keys to avoid hot partitions.
* **Throughput**:
  + Measured in **RU/s**.
  + *Provisioned* or *Autoscale*.
  + *Serverless* = pay-per-request, no RU provisioning.
* **Change Feed**:
  + Streams inserts/updates in order for reactive processing.
* **Indexing**: Automatic by default; can use policies to optimize cost/performance.

**🛠 CLI / SDK Examples**

**Blob Storage**

# Create storage account

az storage account create -n mystorage -g myRG -l westeurope --sku Standard\_LRS

# Upload blob

az storage blob upload --account-name mystorage --container-name mycontainer \

--name myfile.txt --file ./localfile.txt

**Cosmos DB**

# Create Cosmos DB account with SQL API

az cosmosdb create --name mycosmos --resource-group myRG --kind GlobalDocumentDB

# Create a database and container

az cosmosdb sql database create --account-name mycosmos --name mydb --resource-group myRG

az cosmosdb sql container create --account-name mycosmos --database-name mydb \

--name mycontainer --partition-key-path "/userid" --throughput 400

**⚠️ Gotchas & Exam Traps**

* **Soft delete + versioning** = recovery from accidental delete/overwrite.
* **Immutable storage** = compliance WORM, can’t be undone.
* **Queue Storage** is not ordered (no FIFO) — use **Service Bus with sessions** if order matters.
* Cosmos DB **RU/s throttling (429)** must be handled with **retry + exponential backoff**.
* Cosmos DB **multi-master replication** enables global writes but requires **conflict resolution**.
* **Default consistency** = Session.
* **Archive tier** is **offline**; must rehydrate before access.

**❓ Mini Practice Qs**

**Q1.** Which two features allow blob recovery after deletion or overwrite?

* A) Soft Delete
* B) Versioning
* C) Legal Hold
* D) Archive tier  
  ✅ **Answer**: A & B – Soft Delete and Versioning.

**Q2.** Your Cosmos DB workload must support global writes with active-active replication. Which feature do you enable?

* A) Change Feed
* B) Multi-master replication
* C) Strong consistency
* D) Autoscale  
  ✅ **Answer**: B – Multi-master replication.

**Q3.** Which Cosmos DB consistency model guarantees linearizability but with highest latency?

* A) Eventual
* B) Strong
* C) Session
* D) Bounded Staleness  
  ✅ **Answer**: B – Strong consistency.

**Q4.** Which storage service is schema-less and best for large-scale key-value storage?

* A) Blob Storage
* B) Cosmos DB SQL API
* C) Azure Table Storage
* D) Azure SQL Database  
  ✅ **Answer**: C – Azure Table Storage.

**AZ-204 Exam Cram Guide – Part 3 (Security)**

*(15–20% of the exam)*

**📚 Deep Review**

**Azure Active Directory (AAD)**

* **App registrations** → identity for apps (client ID, secret/cert, redirect URIs).
* **Service principals** → app identity instance inside a tenant.
* **App roles & scopes** → claims added to tokens, enforce authorization.
* **Multi-tenant apps** → allow external orgs to authenticate.

**OAuth 2.0 & OpenID Connect**

* **Authorization Code w/ PKCE** → ✅ for SPAs (secure, avoids implicit).
* **Client Credentials** → ✅ for daemons/services (no user context).
* **Device Code Flow** → ✅ for devices w/ limited input (TV, Xbox).
* **Implicit Flow** → ❌ deprecated (use Auth Code + PKCE instead).
* **Refresh tokens** → obtain new access tokens without re-login.

**Managed Identities**

* **System-assigned**: lifecycle tied to the resource.
* **User-assigned**: standalone identity, reusable across resources.
* Use to access **Key Vault, Storage, Cosmos DB, SQL, etc.** without secrets.

**Azure Key Vault**

* Store **secrets, keys, certificates** securely.
* **Access Control**:
  + **RBAC** (recommended) → integrate with Azure RBAC.
  + **Access policies** (legacy, still supported).
* **Key Vault references**: inject secrets directly into App Service/Functions config using @Microsoft.KeyVault(...).
* **Firewall & networking**: restrict access to VNET/private endpoint.

**API Security**

* **Validate JWT tokens**: Always check:
  + iss (issuer) → trusted AAD endpoint.
  + aud (audience) → matches your API.
  + exp (expiry).
* Use **APIM validate-jwt policy** to enforce automatically.
* **Scopes vs Roles**:
  + *Scopes* → delegated permissions (user context).
  + *Roles* → app permissions (service context).

**🛠 CLI / SDK Examples**

**App Registration**

# Register a new app

az ad app create --display-name myapp --reply-urls http://localhost:3000

# Create a service principal

az ad sp create --id <appId>

**Key Vault**

# Create Key Vault

az keyvault create -n mykv -g myRG -l westeurope

# Store secret

az keyvault secret set --vault-name mykv --name "dbPassword" --value "P@ssw0rd!"

# Retrieve secret (using managed identity in code recommended)

az keyvault secret show --vault-name mykv --name "dbPassword"

**⚠️ Gotchas & Exam Traps**

* SPAs must **not** use Implicit flow. Use **Auth Code + PKCE**.
* **Managed Identity** = preferred way to access Key Vault, Storage, etc. No secrets.
* Key Vault → use **RBAC** for new deployments (legacy access policies are being phased out).
* **Always validate JWT** → audience + issuer claims (not just signature).
* **Multi-tenant apps** need consent from each external tenant.
* **Refresh tokens** expire → design for token renewal.

**❓ Mini Practice Qs**

**Q1.** Which flow should you use for a background service that needs to call Microsoft Graph with no user?

* A) Authorization Code
* B) Client Credentials
* C) Device Code
* D) Implicit  
  ✅ **Answer**: B – Client Credentials.

**Q2.** Your SPA is currently using Implicit Flow. Microsoft recommends switching to:

* A) Auth Code + PKCE
* B) ROPC
* C) Client Credentials
* D) Device Code  
  ✅ **Answer**: A – Auth Code with PKCE.

**Q3.** Which two claims should an API validate from a JWT token?

* A) exp (expiration)
* B) iss (issuer)
* C) aud (audience)
* D) nbf (not before)  
  ✅ **Answer**: B & C – issuer + audience.

**Q4.** Which Key Vault access model is recommended for new deployments?

* A) Access policies
* B) RBAC
* C) Public endpoint only
* D) SAS keys  
  ✅ **Answer**: B – RBAC is recommended.

**AZ-204 Exam Cram Guide – Part 4 (Monitoring, Troubleshooting & Optimization)**

*(10–15% of the exam)*

**📚 Deep Review**

**Azure Monitor**

* Unified platform for metrics, logs, alerts.
* Works across **VMs, App Service, Functions, AKS, Containers, Storage, etc.**

**Application Insights**

* Deep application monitoring (performance + telemetry).
* **Key features**:
  + *Live Metrics Stream* → real-time request telemetry.
  + *Profiler* → find performance bottlenecks in production.
  + *Snapshot Debugger* → capture app state on exceptions.
  + *Availability tests* → synthetic probes from global locations.
  + *Dependency tracking* → auto-tracks SQL queries, HTTP calls, etc.
* **Sampling**:
  + *Fixed* → static percentage.
  + *Adaptive* → adjusts dynamically, always keeps exceptions.

**Log Analytics + KQL**

* Log queries run in **Log Analytics Workspace**.
* **Common functions**:
  + where → filter
  + summarize → aggregate
  + bin() → group by time intervals
  + percentile() → latency thresholds (p95, p99)

**Alerts**

* **Metric Alerts** → trigger when counters (CPU %, Memory, Requests) exceed thresholds.
* **Log Alerts** → trigger based on KQL query results.
* **Activity Log Alerts** → track control-plane actions (e.g., VM deleted).
* **Action Groups** → define notification actions (email, SMS, webhook, Function).

**Diagnostics**

* **Diagnostic Settings** → route logs/metrics to Log Analytics, Event Hub, or Storage.
* **NSG Flow Logs** → capture network traffic, analyze with Traffic Analytics.
* **Container Insights** → monitor AKS/Container Apps.

**🛠 CLI / SDK Examples**

**Enable App Insights on Function App**

az monitor app-insights component create \

--app myfuncai --location westeurope --resource-group myRG

az functionapp update --name myfuncapp --resource-group myRG \

--set appInsightsKey=<InstrumentationKey>

**Enable Diagnostic Logs**

az monitor diagnostic-settings create \

--name diag1 --resource <resource-id> \

--workspace <log-analytics-id> \

--logs '[{"category": "AppServiceHTTPLogs","enabled": true}]'

**Sample KQL**

requests

| where timestamp > ago(1h)

| summarize count() by resultCode

**⚠️ Gotchas & Exam Traps**

* **App Insights Profiler** = performance bottlenecks.
* **Snapshot Debugger** = capture memory state on exception.
* **Live Metrics Stream** = real-time monitoring, no redeploy required.
* **Adaptive Sampling** keeps exceptions even when throttling telemetry.
* **Metric Alerts** = near real-time; **Log Alerts** = query-driven.
* **Activity Log Alerts** = control-plane (e.g., resource deleted).
* **Diagnostic Settings** must be enabled per-resource (not global).

**❓ Mini Practice Qs**

**Q1.** Which Application Insights feature provides near real-time telemetry during incidents?

* A) Profiler
* B) Snapshot Debugger
* C) Live Metrics Stream
* D) Availability Tests  
  ✅ **Answer**: C – Live Metrics Stream.

**Q2.** You want to capture memory dumps when an exception occurs in production. Which feature should you use?

* A) Profiler
* B) Snapshot Debugger
* C) Log Analytics
* D) Alerts  
  ✅ **Answer**: B – Snapshot Debugger.

**Q3.** Which alert type fires when CPU usage > 80% for 5 minutes?

* A) Log Alert
* B) Activity Log Alert
* C) Metric Alert
* D) Action Group  
  ✅ **Answer**: C – Metric Alert.

**Q4.** You need to centralize logs from multiple subscriptions. Which service do you use?

* A) Storage Account
* B) Log Analytics Workspace
* C) Metrics Explorer
* D) Action Groups  
  ✅ **Answer**: B – Log Analytics Workspace.

**AZ-204 Exam Cram Guide – Part 5 (Integration)**

*(20–25% of the exam)*

**📚 Deep Review**

**Event Grid**

* Lightweight **pub/sub eventing** service.
* Event-driven → push model (no polling).
* Scenarios: Blob created, Resource group deleted, Custom events.
* Supports **filters** & multiple subscribers (Functions, Logic Apps, Webhooks).
* **Validation handshake**: subscriber must echo validationCode on subscription creation.

**Event Hubs**

* High-throughput **telemetry ingestion** (IoT, logs, clicks).
* Millions of events/sec.
* **Partitions** → parallel processing.
* **Consumer groups** → multiple independent readers.
* At-least-once delivery, ordered **within partition only**.

**Service Bus**

* Enterprise-grade messaging.
* **Queues** → simple point-to-point.
* **Topics & Subscriptions** → pub/sub.
* Supports **FIFO** with *sessions*.
* Supports **dead-letter queues (DLQs)** for poison messages.
* At-least-once delivery, supports transactions.

**API Management (APIM)**

* Full API gateway.
* Features:
  + **Security** → validate-jwt, rate-limit, quota-by-key.
  + **Transformation** → rewrite-uri, set-header.
  + **Caching** → cache-lookup, cache-store.
  + **Developer portal** → auto-generated docs.
* Common policies tested:
  + validate-jwt → enforces token validation.
  + rate-limit-by-key → throttle by subscription key.
  + quota-by-key → enforce total quota.

**Logic Apps**

* Low-code workflow automation.
* Trigger types: Timer, Event Grid, Service Bus, HTTP, Blob events.
* Best for integration across **SaaS + Azure services**.
* Pricing → per action/connector execution.

**Azure Cache for Redis**

* In-memory caching.
* Reduces latency for high-traffic APIs.
* Supports pub/sub, session storage, caching layers.

**🛠 CLI / SDK Examples**

**Event Grid**

# Subscribe to blob storage events

az eventgrid event-subscription create \

--name mysub --source-resource-id <storage-id> \

--endpoint https://myfunc.azurewebsites.net/runtime/webhooks/EventGrid?functionName=MyFunc

**Service Bus**

# Create a Service Bus namespace & queue

az servicebus namespace create --resource-group myRG --name myns --location westeurope

az servicebus queue create --namespace-name myns --name myqueue --resource-group myRG

**APIM**

# Create an APIM instance

az apim create --name myapim --resource-group myRG --publisher-email admin@contoso.com \

--publisher-name "Contoso"

**⚠️ Gotchas & Exam Traps**

* **Event Grid** = reactive eventing (fan-out). **Event Hubs** = telemetry streaming. **Service Bus** = reliable enterprise messaging.
* **Service Bus sessions** → FIFO within a session, not across queue.
* **APIM validate-jwt** → check iss, aud, exp.
* **Dead-letter queues** exist in Service Bus, not Queue Storage.
* **Event Grid subscriber handshake** → must echo validationCode.
* **Logic Apps** → billed per action, not per runtime.
* **Redis Cache** → store session state or frequently accessed data.

**❓ Mini Practice Qs**

**Q1.** You must handle 1M telemetry events/sec from IoT devices. Which service?

* A) Event Grid
* B) Service Bus
* C) Event Hubs
* D) Queue Storage  
  ✅ **Answer**: C – Event Hubs.

**Q2.** Which Service Bus feature ensures ordered processing per customer?

* A) Dead-letter queue
* B) Duplicate detection
* C) Sessions
* D) FIFO queue  
  ✅ **Answer**: C – Sessions.

**Q3.** You’re publishing APIs for partners. Which APIM policy enforces JWT token checks?

* A) validate-jwt
* B) quota-by-key
* C) cache-lookup
* D) set-header  
  ✅ **Answer**: A – validate-jwt.

**Q4.** You must build an approval workflow that connects Outlook, SharePoint, and Teams with no custom code. Which service do you use?

* A) Azure Functions
* B) Azure Logic Apps
* C) Event Hubs
* D) AKS  
  ✅ **Answer**: B – Logic Apps.

**Q5.** Which caching service improves response times for a high-traffic API?

* A) Blob Storage
* B) Cosmos DB
* C) Azure Cache for Redis
* D) Event Grid  
  ✅ **Answer**: C – Redis Cache.

**AZ-204 Exam Cram Guide – Part 6 (IaaS & Third-Party Services)**

*(10–15% of the exam)*

**📚 Deep Review**

**Infrastructure as Code (IaC)**

* **ARM templates** = JSON declarative infra definitions.
* **Bicep** = higher-level DSL that compiles to ARM JSON.
* **Terraform** = 3rd-party tool, multi-cloud.

**Bicep features**:

* param → parameters (secureString / secureObject for secrets).
* output → expose values post-deployment.
* existing → reference existing resources.
* **Modules** → reusable definitions.
* **br: syntax** → reference modules in Bicep registries.

**Azure CLI IaC**

* Preview deployments without applying:
* az deployment group what-if --resource-group myRG --template-file main.bicep
* Build Bicep → ARM JSON:
* az bicep build --file main.bicep

**VM Extensions**

* Allow post-deployment configuration.

Common extensions:

* **Custom Script Extension** → run scripts (Linux/Windows).
* **DSC Extension** → enforce config compliance (Windows).
* **Azure Monitor Agent (AMA)** → telemetry collection.
* **cloud-init** → Linux VM initialization at boot.

**Third-Party Services & Containers**

* Deploy workloads that rely on **VMs, Docker, Kubernetes**.
* **VM Scale Sets (VMSS)** for horizontal scaling of IaaS.
* **AKS** for container orchestration, but exam focus = IaC & VM config.

**🛠 CLI / SDK Examples**

**Bicep Example**

param adminPassword secureString

param location string = resourceGroup().location

resource vm 'Microsoft.Compute/virtualMachines@2021-11-01' = {

name: 'myVM'

location: location

properties: {

osProfile: {

adminUsername: 'azureuser'

adminPassword: adminPassword

}

}

}

output vmId string = vm.id

**VM Extension Deployment**

az vm extension set \

--resource-group myRG \

--vm-name myVM \

--name CustomScriptExtension \

--publisher Microsoft.Azure.Extensions \

--settings '{"fileUris":["https://my.blob.core.windows.net/scripts/setup.sh"],"commandToExecute":"./setup.sh"}'

**⚠️ Gotchas & Exam Traps**

* **Bicep is not a runtime** → compiles to ARM JSON (executed by Resource Manager).
* **secureString / secureObject** = sensitive input → never logged.
* **what-if** = safe preview of changes.
* **cloud-init** = Linux bootstrapping (not Windows).
* **DSC extension** = Windows compliance configs.
* **Custom Script Extension** = ad-hoc scripts for any VM.
* Exam may give you JSON vs. Bicep side-by-side and ask which is easier to manage → **answer = Bicep**.

**❓ Mini Practice Qs**

**Q1.** You need to pass an admin password securely into a VM deployment using Bicep. Which parameter type should you use?

* A) string
* B) secureString
* C) object
* D) secureObject  
  ✅ **Answer**: B – secureString.

**Q2.** Which VM extension enforces configuration compliance on Windows VMs?

* A) Custom Script Extension
* B) DSC Extension
* C) Diagnostics
* D) Azure Monitor Agent  
  ✅ **Answer**: B – DSC Extension.

**Q3.** You want to preview infra changes before deploying a Bicep template. Which command do you use?

* A) az resource show
* B) az group create
* C) az deployment group what-if
* D) az bicep build  
  ✅ **Answer**: C – what-if.

**Q4.** Your Linux VM must install Nginx at first boot using IaC. Which tool should you use?

* A) Custom Script Extension
* B) cloud-init
* C) DSC
* D) Azure Policy  
  ✅ **Answer**: B – cloud-init.

**Q5.** Which Bicep keyword references an existing storage account without redeploying it?

* A) module
* B) existing
* C) resource
* D) output  
  ✅ **Answer**: B – existing.

**✅ Final Checklist for IaaS & Third-Party**

* Know **Bicep vs ARM** roles.
* Memorize parameter types (secureString, secureObject).
* Remember **what-if** vs **build** commands.
* Understand **Custom Script Extension vs DSC vs cloud-init**.
* Recognize when to use **VMSS** vs **AKS vs ACI**.

**AZ-204 Exam Cram Guide – Part 7 (API Development, Networking & Edge Services)**

(~5–10% of exam weight, but often the tricky curveball questions)

**📚 Deep Review**

**API Development & Management**

* **App Service API Hosting**
  + Host REST APIs in Web Apps or Functions.
  + Support OpenAPI (Swagger) docs → /swagger.json for discoverability.
* **API Management (APIM) Import**
  + Import APIs from:
    - OpenAPI/Swagger JSON
    - WSDL (SOAP)
    - Azure Functions / Logic Apps
  + Export APIs back out for client SDK generation.
* **APIM Versioning**
  + URL path (/v1/customers)
  + Query string (?api-version=1.0)
  + HTTP header (api-version: 1.0)
  + Policies can enforce backward compatibility.

**Networking & Security Integration**

* **VNET Integration**
  + App Service (Premium & above) supports VNET Integration (outbound).
  + Functions Premium plan supports VNET Integration.
  + ACA (Container Apps) supports both ingress & VNET.
* **Private Endpoints vs Service Endpoints**
  + **Private Endpoint** → Private IP in VNET, service accessible only privately.
  + **Service Endpoint** → Extends VNET to Azure service, public IP still used but restricted to subnet.
* **NSGs & Firewalls**
  + Key Vault, Storage, Cosmos DB → lock down with firewall rules or private endpoints.

**Deployment & DevOps**

* **App Service Deployment Options**
  + **Zip deploy**: Upload zipped artifacts.
  + **Run-from-package**: Deploy a zip mounted read-only → faster cold start.
  + **Local Git / GitHub Actions / Azure DevOps Pipelines**.
  + **Deployment slots**: Blue/green + slot settings (sticky vs non-sticky).
* **CI/CD Integration**
  + GitHub Actions: azure/webapps-deploy@v2 action.
  + Azure DevOps: AzureWebApp@1 task.
  + Functions: func azure functionapp publish <appname>.

**Niche Services Microsoft Tests**

* **Azure SignalR Service**
  + Real-time WebSockets at scale.
  + Functions integration → serverless real-time updates (chat, dashboards).
* **Azure Cognitive Services (edge mention)**
  + Prebuilt APIs (Vision, Text Analytics, Translator, Speech).
  + Authentication: API key or AAD.
* **Azure Front Door & CDN (light touch)**
  + Global load balancing + caching.
  + Integrates with App Service APIs.

**🛠 CLI / SDK Examples**

**APIM Import from Swagger**

az apim api import --service-name myapim --resource-group myRG \

--path myapi --specification-format OpenApi \

--specification-url https://contoso.com/swagger.json

**Private Endpoint for Storage**

az network private-endpoint create -g myRG -n myPE \

--vnet-name myVnet --subnet mySubnet \

--private-connection-resource-id <storage-id> \

--group-id blob --connection-name mystorageconn

**Zip Deploy to App Service**

az webapp deployment source config-zip \

--resource-group myRG --name myapp --src app.zip

**SignalR Create**

az signalr create --name mySignalR --resource-group myRG \

--sku Standard\_S1 --unit-count 1 --service-mode Default

**⚠️ Gotchas & Exam Traps**

* APIM versioning → Know path vs query vs header.
* Private Endpoint = fully private IP, Service Endpoint = still public.
* Functions Consumption plan **cannot** do VNET integration; must be Premium.
* Deployment slot settings: Mark secrets (connection strings) as slot-sticky.
* Zip deploy is read-only — can’t write to wwwroot.
* SignalR with Functions → must use negotiation endpoint pattern.
* Cognitive Services sometimes appear in exam but just at API-key basics.

**❓ Mini Practice Qs**

**Q1.** You need to deploy a new API version /v2/customers while keeping /v1/customers available. Which APIM versioning strategy fits?

* A) Query string
* B) Path
* C) Header
* D) Policy  
  ✅ **Answer: B – Path versioning**

**Q2.** Your Function (Consumption plan) must access a private Storage account via VNET. What’s required?

* A) Service Endpoint
* B) Private Endpoint
* C) Upgrade to Premium plan
* D) API Management  
  ✅ **Answer: C – Premium plan supports VNET Integration**

**Q3.** Which deployment method runs directly from a zip file, mounted read-only?

* A) Local Git
* B) Zip deploy
* C) Run-from-package
* D) Docker deploy  
  ✅ **Answer: C – Run-from-package**

**Q4.** Which service provides real-time WebSocket communication for a Functions-based chat app?

* A) Event Grid
* B) Service Bus
* C) SignalR Service
* D) Event Hubs  
  ✅ **Answer: C – SignalR Service**

**Q5.** In Key Vault, how do you restrict access so only your VNET can connect?

* A) RBAC
* B) Access Policy
* C) Private Endpoint
* D) Managed Identity  
  ✅ **Answer: C – Private Endpoint**