5. Connect to and consume Azure services and third-party services

└ **5.1 Implement API Management**

└ **5.1.1 Create an Azure API Management instance**

1. What is Azure API Management (APIM) and when should it be used?
2. What are the components of an APIM instance?
3. What are the pricing tiers of APIM and their core differences?
4. How do you provision an APIM instance using Azure CLI?
5. What are the networking options when creating an APIM instance (public vs. internal)?
6. How do you configure a custom domain during APIM creation?
7. What is the default management and developer portal behavior post-deployment?
8. What is the purpose of the publisher email and name fields?
9. What RBAC roles are relevant for managing APIM instances?
10. What are the prerequisites for deploying APIM in a VNET?

---

1. What is Azure API Management (APIM) and when should it be used?

A fully managed service to publish, secure, transform, and monitor APIs. Use APIM to expose internal or third-party services via a unified gateway with rate limiting, authentication, logging, and analytics.

---

2. What are the components of an APIM instance?

- API Gateway – Handles API calls, policies, and traffic.
- Developer Portal – Auto-generated site for API consumers.
- Management Plane – For configuring APIs and policies.
- Publisher Portal – Admin UI in Azure Portal.

---

3. What are the pricing tiers of APIM and their core differences?

- Developer – Non-production use, low-cost, no SLA.
- Basic – Entry-level prod tier, no VNET support.
- Standard – VNET support, SLA-backed, scalable.
- Premium – Multi-region, zone redundancy, higher throughput.
- Consumption – Serverless, pay-per-call, no custom domain/VNET.

---

4. How do you provision an APIM instance using Azure CLI?

```
az apim create \
  --name myapim \
  --resource-group myrg \
  --publisher-email admin@contoso.com \
  --publisher-name Contoso \
  --location eastus \
  --sku-name Developer
```

---

5. What are the networking options when creating an APIM instance (public vs. internal)?

- External (default): Public endpoint for API gateway and portals.
- Internal: Requires Premium tier, deploys into a VNET with private endpoints for secure access.

6. How do you configure a custom domain during APIM creation?
Use Azure CLI or Portal after deployment. Requires a valid certificate (Key Vault or PFX). Configure for gateway, developer portal, and management endpoints separately.

---

7. What is the default management and developer portal behavior post-deployment?
Both portals are enabled with default subdomains. Developer portal is public and customizable. Management portal is accessed via Azure Portal and not intended for external users.

---

8. What is the purpose of the publisher email and name fields?
Used in developer portal metadata and system-generated emails (e.g., confirmation, invitations). Must be valid to ensure email-based features work correctly.

---

9. What RBAC roles are relevant for managing APIM instances?

- API Management Service Contributor – Full management access.
- Reader – View-only access.
- Developer Portal Administrator – Limited to developer portal customization.

---

10. What are the prerequisites for deploying APIM in a VNET?

- Must use Premium tier.
- Requires subnet with enough IPs and correct NSG/UdR settings.
- DNS resolution must be in place for Azure services used by APIM.

5. Connect to and consume Azure services and third-party services

└ **5.1 Implement API Management**

└ **5.1.2 Create and document APIs**

1. How do you import an API into Azure API Management?
2. What API definition formats are supported for import?
3. What are the common ways to create an API manually in APIM?
4. What is the purpose of API operations and how are they defined?
5. How do you add or modify request/response parameters in APIM?
6. How does versioning work in APIM for APIs?
7. How do you document APIs using OpenAPI (Swagger) in APIM?
8. What is the role of the developer portal in API documentation?
9. How do you secure developer portal access?
10. What tools support automated API documentation publishing to APIM?

---

1. How do you import an API into Azure API Management?

Use the Azure Portal, CLI, or ARM/Bicep to import via OpenAPI, WSDL, GraphQL, or direct URL.

```
az apim api import \
  --resource-group myrg \
  --service-name myapim \
  --path myapi \
  --specification-format OpenApi \
  --specification-path ./openapi.json
```

---

2. What API definition formats are supported for import?

- OpenAPI 2.0/3.0 (JSON/YAML)
- WSDL (SOAP)
- GraphQL schemas
- Azure Functions or App Services (via App Insights tracing)

---

3. What are the common ways to create an API manually in APIM?

- Define operations manually in the Azure Portal
- Use Azure CLI or ARM templates
- Add operations under a base path, define HTTP verbs, request parameters, and response types

---

4. What is the purpose of API operations and how are they defined?

Operations define how the API is consumed (e.g., GET /users). Each includes:

- HTTP method
- URL template
- Request/response schema
- Optional policies (rate limits, auth)

---

5. How do you add or modify request/response parameters in APIM?

Via Portal or CLI:

- Define path/query/header parameters for requests
- Define expected status codes and response body schemas
- Example: Add header param in operation settings under Inbound Processing

6. How does versioning work in APIM for APIs?
Supported methods:

- Path-based (e.g., /v1/orders)
- Query string (e.g., ?api-version=1.0)
- Header-based (e.g., X-Version: 1.0)
  APIM supports version sets to group versions under a single API entity.

---

7. How do you document APIs using OpenAPI (Swagger) in APIM?
Import an OpenAPI file to auto-generate operations and documentation.
OpenAPI description appears in both the Azure Portal and developer portal.
Supports Swagger UI rendering in developer portal.

---

8. What is the role of the developer portal in API documentation?
Provides a public or private interface for users to:

- Discover and test APIs
- View request/response details
- Access API keys
- Read autogenerated documentation

---

9. How do you secure developer portal access?

- Enable require sign-in in portal settings
- Configure identity providers (AAD, Facebook, etc.)
- Assign users to products with subscription-based access

---

10. What tools support automated API documentation publishing to APIM?

- Swagger/OpenAPI CLI
- Postman → export to OpenAPI
- Azure DevOps/GitHub Actions → automate import via `az apim api import`
- Supports CI/CD for keeping API definitions in sync

5. Connect to and consume Azure services and third-party services

└ **5.1 Implement API Management**

└ **5.1.3 Configure access to APIs**

1. What authentication mechanisms are supported by Azure API Management (APIM)?
2. How do you secure APIs using subscription keys?
3. How do you configure OAuth 2.0 authentication with APIM?
4. How to configure a client application to call an APIM-secured API using a bearer token?
5. How do you restrict API access using IP filtering in APIM?
6. What is the role of policies in controlling access to APIs?
7. How can you enforce rate limits and quotas per subscription in APIM?
8. How do you enable CORS in API Management?
9. What is the difference between product-level and API-level access control?
10. How do you use managed identities to call APIs behind APIM securely?

---

1. What authentication mechanisms are supported by Azure API Management (APIM)?

- Subscription key
- OAuth 2.0 / OpenID Connect
- JWT validation
- Client certificates
- Managed identities

---

2. How do you secure APIs using subscription keys?

- Add APIs to a product.
- Require subscription on the product.
- Each caller must pass `Ocp-Apim-Subscription-Key` in header or query.

---

3. How do you configure OAuth 2.0 authentication with APIM?

- Register APIM as a client app in Microsoft Entra ID (or other provider).
- Configure OAuth 2.0 settings in APIM (under security tab).
- Set `validate-jwt` policy in inbound section of the API to enforce token validation.

---

4. What are the steps to configure a client application to call an APIM-secured API using a bearer token?

1. Register the client app in Entra ID.
2. Acquire token using MSAL or ADAL libraries.
3. Call the API with `Authorization: Bearer <token>` header.
4. Ensure APIM has a `validate-jwt` policy matching token settings.

---

5. How do you restrict API access using IP filtering in APIM?

- Use the `check-header` or `check-ip` policy in the inbound policy section.
- Example:

```
<check-header name="X-Forwarded-For" failed-check-httpcode="403" failed-check-error-message="Access denied">
  <value>203.0.113.1</value>
</check-header>
```

6. What is the role of policies in controlling access to APIs?

- Policies define request/response behavior at runtime.
- Used to enforce security (e.g., validate-jwt, check-header), rate limits, IP restrictions, CORS, etc.
- Applied at inbound, backend, outbound, or on error sections.

---

7. How can you enforce rate limits and quotas per subscription in APIM?

- Use built-in rate-limit and quota policies.
- Define policies in product or API scope.
- Example:

```
<rate-limit calls="10" renewal-period="60" />
<quota calls="1000" renewal-period="604800" />
```

---

8. How do you enable CORS in API Management?

- Add the cors policy in the inbound section.
- Example:

```
<cors allow-credentials="true">
  <allowed-origins><origin>*</origin></allowed-origins>
  <allowed-methods><method>GET</method></allowed-methods>
</cors>
```

---

9. What is the difference between product-level and API-level access control?

- Product-level: Controls who can access any API within the product using subscriptions.
- API-level: Policies or restrictions applied to individual APIs regardless of product membership.

---

10. How do you use managed identities to call APIs behind APIM securely?

- Enable system-assigned or user-assigned identity on APIM.
- Grant API backend (e.g., Azure Function) the necessary role (e.g., Function App Contributor).
- Use authentication-managed-identity policy in outbound call:

```
<authentication-managed-identity resource="https://<resource>" />
```

5. Connect to and consume Azure services and third-party services

└ **5.1 Implement API Management**

└ **5.1.4 Implement policies for APIs**

1. What are API Management policies and where can they be applied?
2. How do you use the set-header policy to manipulate request/response headers?
3. How does the validate-jwt policy work for token validation?
4. How do you apply rate limiting using the rate-limit policy?
5. What's the difference between quota and rate-limit policies?
6. How do you rewrite URLs using rewrite-uri or set-backend-service?
7. How do you handle conditional logic in policies using choose and when?
8. How can you transform request or response bodies using set-body?
9. What tools are used to author, test, and debug APIM policies?
10. What are best practices for policy organization and reuse?

---

1. What are API Management policies and where can they be applied?

- XML-based logic executed at runtime.
- Applied at four scopes: inbound, backend, outbound, and on-error.
- Can be configured at global, product, API, or operation levels.

---

2. How do you use the `set-header` policy to manipulate request/response headers?

- Adds or updates headers.
- Example:

```
<set-header name="X-Custom-Header" exists-action="override">
  <value>my-value</value>
</set-header>
```

---

3. How does the `validate-jwt` policy work for token validation?

- Validates JWT tokens in Authorization header.
- Requires configuration of issuer, audience, and signing keys.
- Example:

```
<validate-jwt header-name="Authorization" require-scheme="Bearer">
  <openid-config url="https://login.microsoftonline.com/<tenant>/v2.0/.well-known/openid-configuration" />
  <required-claims><claim name="aud"><value>api-client-id</value></claim></required-claims>
</validate-jwt>
```

---

4. How do you apply rate limiting using the `rate-limit` policy?

- Restricts number of calls in a time window.
- Example:

```
<rate-limit calls="100" renewal-period="60" />
```

  Limits client to 100 calls per minute.

---

5. What's the difference between `quota` and `rate-limit` policies?

- `rate-limit`: short-term throttle (e.g., per minute).
- `quota`: long-term usage limit (e.g., daily/weekly).
- Both can be used together for layered control.

6. How do you rewrite URLs using `rewrite-uri` or `set-backend-service`?

- `rewrite-uri`: Changes request path.
  Example:
  ```
  <rewrite-uri template="/v2/resource" />
  ```
- `set-backend-service`: Redirects to a different backend.
  Example:
  ```
  <set-backend-service base-url="https://api.contoso.com/v2" />
  ```

---

7. How do you handle conditional logic in policies using `choose` and `when`?

- Enables branching logic based on conditions.
  Example:
  ```
  <choose>
    <when condition="@(context.Request.Headers["x-version"] == "v2")">
      <set-backend-service base-url="https://api-v2.contoso.com" />
    </when>
    <otherwise>
      <set-backend-service base-url="https://api-v1.contoso.com" />
    </otherwise>
  </choose>
  ```

---

8. How can you transform request or response bodies using `set-body`?

- Overwrites the payload.
  Example (JSON):
  ```
  <set-body>@("{\"message\": \"Access denied\"}")</set-body>
  ```
- Can also use Liquid templates for dynamic content.

---

9. What tools are used to author, test, and debug APIM policies?

- Azure Portal policy editor with IntelliSense.
- Test console in Azure Portal.
- Developer portal (limited testing).
- Trace/debug by enabling "trace" and reviewing trace logs in the test console.

---

10. What are best practices for policy organization and reuse?

- Keep logic centralized at product/API level when possible.
- Use named values for reusable values.
- Document policy use with `<comment>` blocks.
- Avoid deep nesting for readability.

5. Connect to and consume Azure services and third-party services

└ 5.2 Develop event-based solutions

└ 5.2.1 Implement solutions that use Azure Event Grid

1. What is Azure Event Grid and what are its primary use cases?
2. What are the key components of Event Grid (events, topics, event subscriptions)?
3. How do you create a custom topic and event subscription via Azure CLI?
4. What event sources can trigger Event Grid?
5. What are the supported event handlers (destinations)?
6. How do you secure Event Grid with authentication and authorization?
7. How do you filter events in Event Grid subscriptions?
8. How do you enable dead-lettering for undeliverable events?
9. How do you validate Event Grid event schema in consumers?
10. How do you troubleshoot failed event deliveries?
11. Is Event Grid push or pull-based?
12. What delivery guarantees does Event Grid provide?
13. What is the default schema of an Event Grid event?
14. How do you choose between Event Grid, Event Hubs, and Service Bus?
15. How are retries and error handling managed in Event Grid?

---

1. What is Azure Event Grid and what are its primary use cases?

Azure Event Grid is a fully managed event routing service that enables reactive programming using events from Azure services or custom sources.
Use cases: Event-driven architecture, serverless automation, resource provisioning notifications, microservices decoupling.

---

2. What are the key components of Event Grid?

- Event Sources: Where events originate (e.g., Blob Storage, Resource Groups).
- Topics: Channels where events are sent.
- Event Subscriptions: Define how to handle events (e.g., send to Function, Webhook).
- Event Handlers: Endpoints that process events (e.g., Azure Function, Logic Apps).

---

3. How do you create a custom topic and event subscription via Azure CLI?

```
az eventgrid topic create --name mytopic --resource-group myrg --location eastus
az eventgrid event-subscription create --name mysub --source-resource-id
/subscriptions/<id>/resourceGroups/myrg/providers/Microsoft.EventGrid/topics/mytopic --endpoint <url>
```

---

4. What event sources can trigger Event Grid?

- Native: Blob Storage, Resource Groups, Event Hubs, IoT Hub, Media Services, etc.
- Custom sources: Via custom topics.
- Third-party: via webhook-compatible services.

---

5. What are the supported event handlers (destinations)?

- Azure Function
- Logic Apps
- Event Hubs
- Webhooks (HTTP/S endpoint)
- Service Bus Queue/Topic

6. How do you secure Event Grid with authentication and authorization?

- Event delivery: Signed with validationCode or Azure Active Directory token.
- Inbound auth: Use Azure RBAC for publishing to topics.
- Outbound auth: Use Webhook validation and shared access keys for endpoints.

---

7. How do you filter events in Event Grid subscriptions?

Use `--included-event-types` or advanced filters:

```
--advanced-filter data.subject StringBeginsWith "/blobServices/default/containers/images"
```

Filters reduce noise and only forward relevant events.

---

8. How do you enable dead-lettering for undeliverable events?

Set `--deadletter-destination` when creating a subscription, usually to a Blob Storage container:

```
--deadletter-destination blobcontainer:<storage-account>/<container>
```

This stores failed events for later review.

---

9. How do you validate Event Grid event schema in consumers?

Consumers must handle a **validationEvent** initially.

Respond to the `eventType = Microsoft.EventGrid.SubscriptionValidationEvent` by echoing back `data.validationCode`.

---

10. How do you troubleshoot failed event deliveries?

- Enable dead-lettering.
- Check metrics in Azure Monitor.
- Use `az eventgrid event-subscription show` to inspect subscription state.
- Inspect logs at the handler side (e.g., Azure Function failures).

---

11. Is Event Grid push or pull-based?

Event Grid is push-only — it pushes events to subscribers via HTTP POST; consumers must expose endpoints to receive events.

---

12. What delivery guarantees does Event Grid provide?

At-least-once delivery.

Events are retried with exponential backoff for up to 24 hours if the destination is unavailable.

---

13. What is the default schema of an Event Grid event?

Default schema fields include:

- `id`, `eventType`, `subject`, `data`, `eventTime`, `dataVersion`, `metadataVersion`.
  Custom topics use this schema unless configured for CloudEvents v1.0.

---

14. How do you choose between Event Grid, Event Hubs, and Service Bus?

- Event Grid: Discrete events, fan-out, serverless, push-only.
- Event Hubs: Telemetry, streaming data, high-throughput ingestion.
- Service Bus: Reliable message delivery with ordering and sessions.

---

15. How are retries and error handling managed in Event Grid?

- Retries: Exponential backoff for transient failures.
- Permanent failure: After 24 hours or persistent errors, events are dead-lettered if configured.
- Subscriber should return HTTP 2xx for success, otherwise it triggers retries.

5. Connect to and consume Azure services and third-party services
└ 5.2 Develop event-based solutions
└ 5.2.2 Implement solutions that use Azure Event Hub

1. What is Azure Event Hubs and what is it used for?
2. How does Event Hubs differ from Event Grid and Service Bus?
3. What are the key components of Event Hubs (namespace, event hub, partition, consumer group)?
4. How is data transmitted and consumed in Event Hubs (push/pull)?
5. What are partitions and why are they important in Event Hubs?
6. What is a consumer group and how is it used?
7. What are the differences between Standard, Basic, and Dedicated tiers?
8. How do you send events to Event Hubs using Azure SDK or Azure CLI?
9. How do you consume events from Event Hubs using EventProcessorClient?
10. What is checkpointing and why is it needed?
11. How do you implement checkpointing with Azure Blob Storage?
12. How do you authenticate and authorize access to Event Hubs (Shared Access Policies vs Azure AD)?
13. What are throughput units and how do they impact performance?
14. What are the retention and capture capabilities of Event Hubs?
15. How do Event Hubs integrate with Azure Stream Analytics and Azure Functions?

---

1. What is Azure Event Hubs and what is it used for?
Event Hubs is a data streaming platform and event ingestion service for high-throughput data pipelines.
Use cases: telemetry ingestion, app/event logging, IoT data streams.

---

2. How does Event Hubs differ from Event Grid and Service Bus?

- Event Hubs: Streaming, pull-based, low-latency, high-volume.
- Event Grid: Push-based, lightweight eventing.
- Service Bus: Reliable messaging with ordering and sessions.

---

3. What are the key components of Event Hubs?

- Namespace: Container for event hubs.
- Event Hub: Stream that holds data.
- Partition: Log stream shard for parallelism.
- Consumer Group: Independent view for multiple consumers.

---

4. How is data transmitted and consumed in Event Hubs?
Data is pushed by producers and pulled by consumers. Consumers use SDKs to poll events per partition.

---

5. What are partitions and why are they important?
Partitions enable horizontal scaling and parallel processing.
Events with the same partition key always go to the same partition to preserve order.

---

6. What is a consumer group and how is it used?
A consumer group is a view of an event hub.
Each consumer group has its own state, allowing multiple apps to read independently.

7. What are the differences between Standard, Basic, and Dedicated tiers?

- Basic: Single consumer group, limited features.
- Standard: Multiple consumer groups, capture, 1–20 throughput units.
- Dedicated: Reserved capacity, higher scale, 90+ MB/s ingress.

---

8. How do you send events to Event Hubs using Azure SDK or CLI?

CLI:

```
az eventhubs eventhub send --name <hub> --namespace-name <ns> --resource-group <rg> --message "event data"
```

.NET SDK:

```
await producer.SendAsync(new EventData(Encoding.UTF8.GetBytes("event data")));
```

---

9. How do you consume events from Event Hubs using EventProcessorClient?

Use Azure.Messaging.EventHubs.Processor:

```
var processor = new EventProcessorClient(blobContainerClient, "$Default", connStr, hubName);
processor.ProcessEventAsync += async args => { /* handle event */ };
await processor.StartProcessingAsync();
```

---

10. What is checkpointing and why is it needed?

Checkpointing stores the last successfully processed event offset.
It ensures events aren't reprocessed after restarts and supports fault tolerance.

---

11. How do you implement checkpointing with Azure Blob Storage?

Use EventProcessorClient with a Blob container as storage:

```
new EventProcessorClient(blobContainerClient, "$Default", eventHubConnectionString, eventHubName)
```

---

12. How do you authenticate and authorize access to Event Hubs?

- Shared Access Policies: Use connection string with Send, Listen, or Manage claims.
- Azure AD: Use Azure Identity SDK + RBAC on Event Hub namespace or resource.

---

13. What are throughput units and how do they impact performance?

Each Throughput Unit (TU) allows:

- 1 MB/s ingress or 1000 events/s
- 2 MB/s egress
  You can purchase 1–20 TUs in Standard tier to scale performance.

---

14. What are the retention and capture capabilities of Event Hubs?

- Retention: Configurable up to 7 days (Standard) or 90 days (Dedicated).
- Capture: Automatically stores data to Blob Storage or Data Lake in AVRO format.

---

15. How do Event Hubs integrate with Azure Stream Analytics and Azure Functions?

- Stream Analytics: Direct input via Event Hub; use for real-time queries and dashboards.
- Azure Functions: Use Event Hub trigger to react to events:

```
[EventHubTrigger("hubname", Connection = "EventHubConnection")] EventData[] events
```

5. Connect to and consume Azure services and third-party services
└ **5.3 Develop message-based solutions**
└ **5.3.1 Implement solutions that use Azure Service Bus**

1. What is Azure Service Bus and what are its use cases?
2. What is the difference between Service Bus, Event Hubs, and Event Grid?
3. What messaging models does Service Bus support?
4. What is the difference between standard and premium tiers in Service Bus?
5. What are message sessions and when are they needed?
6. What is dead-lettering and how is it configured?
7. How do you send messages using Azure SDK or CLI?
8. How do you receive messages from a queue?
9. What is peek-lock vs receive-and-delete mode?
10. How do you implement message deferral and why?
11. What is auto-forwarding in Service Bus?
12. How do you configure filters in topic subscriptions?
13. How do you authenticate and authorize access to Service Bus?
14. What are delivery and retry behaviors in Service Bus?
15. How do Service Bus and Azure Functions integrate?

---

1. What is Azure Service Bus and what are its use cases?
A fully managed message broker for enterprise apps.
Use cases: decoupled microservices, order processing, transactions, retries, delayed delivery.

---

2. What is the difference between Service Bus, Event Hubs, and Event Grid?
- **Service Bus**: Reliable messaging, ordering, sessions, dead-lettering.
- **Event Hubs**: High-throughput telemetry streaming.
- **Event Grid**: Lightweight, push-based eventing.

---

3. What messaging models does Service Bus support?
- **Queues**: Point-to-point (1 sender → 1 receiver).
- **Topics/Subscriptions**: Publish-subscribe (1 sender → multiple filtered subscribers).

---

4. What is the difference between standard and premium tiers in Service Bus?
- **Standard**: Basic features, shared resources, limited performance.
- **Premium**: Dedicated compute, faster, supports VNET, encryption, higher scale.

---

5. What are message sessions and when are they needed?
Sessions group related messages for ordered processing.
Required when strict message ordering per entity is needed (e.g., per user/cart).

---

6. What is dead-lettering and how is it configured?
Dead-letter queue (DLQ) stores undeliverable messages (e.g., max delivery attempts reached).
Enable via `EnableDeadLetteringOnMessageExpiration` or use `Abandon`/`DeadLetter` in SDK.

7. How do you send messages using Azure SDK or CLI?

.NET SDK:

```
await sender.SendMessageAsync(new ServiceBusMessage("Hello"));
```

CLI:

```
az servicebus message send --queue-name myqueue --namespace-name myns --resource-group myrg --body "msg"
```

---

8. How do you receive messages from a queue?

.NET SDK:

```
var msg = await receiver.ReceiveMessageAsync();
await receiver.CompleteMessageAsync(msg);
```

---

9. What is peek-lock vs receive-and-delete mode?

- **Peek-lock** (default): Two-phase – lock then explicitly complete.
- **Receive-and-delete**: One-shot; message is deleted on receive.

---

10. How do you implement message deferral and why?

Deferred messages are postponed for later retrieval by sequence number.
Useful when processing must be delayed:

```
var deferred = await receiver.ReceiveDeferredMessageAsync(sequenceNumber);
```

---

11. What is auto-forwarding in Service Bus?

Automatically forwards messages from one queue/topic to another.
Used for message routing, chaining processing steps.

---

12. How do you configure filters in topic subscriptions?

Use SQL-based filters:

```
az servicebus topic subscription rule create \
  --name highPriority \
  --subscription-name mysub \
  --topic-name mytopic \
  --filter-sql-expression "priority = 'high'"
```

---

13. How do you authenticate and authorize access to Service Bus?

- **Shared Access Signature (SAS)**: via policy keys and connection string.
- **Azure AD**: via RBAC roles like `Azure Service Bus Data Sender`.

---

14. What are delivery and retry behaviors in Service Bus?

- Default: 10 delivery attempts, exponential backoff.
- Messages that fail retry go to DLQ if enabled.
- Customize via `MaxDeliveryCount` and `LockDuration`.

---

15. How do Service Bus and Azure Functions integrate?

Use Service Bus trigger:

```
[ServiceBusTrigger("myqueue", Connection = "ServiceBusConnection")] string msg
```

5. Connect to and consume Azure services and third-party services
└ 5.3 Develop message-based solutions
└ 5.3.2 – Implement solutions that use Azure Queue Storage queues

1. What is Azure Queue Storage and what are its typical use cases?
2. How does Azure Queue Storage compare to Azure Service Bus queues?
3. What are the key components of an Azure Storage queue?
4. How are messages added to and retrieved from a queue?
5. What is the default behavior when retrieving messages from a queue?
6. What is message visibility timeout and how does it affect processing?
7. How do you handle message retries and poison messages in Azure Queue Storage?
8. How do you create and manage queues using Azure CLI or SDK?
9. How do you secure access to Azure Queue Storage (SAS tokens, RBAC, shared keys)?
10. What are the message size limits and encoding requirements?
11. How do you configure and use Base64 encoding with queue messages?
12. What are best practices for processing at scale with Queue Storage?
13. How do you integrate Azure Queue Storage with Azure Functions?
14. How do you monitor and troubleshoot Azure Queue Storage usage?
15. What are common limitations of Azure Queue Storage compared to other messaging services?

---

1. What is Azure Queue Storage and what are its typical use cases?
A simple, cloud-based message queueing service for decoupling components.
Use cases: task queues, background jobs, retry buffers.

---

2. How does Azure Queue Storage compare to Azure Service Bus queues?

- **Queue Storage**: Basic FIFO queues, lightweight, no ordering guarantees, no DLQ, no sessions.
- **Service Bus**: Enterprise-grade, supports DLQ, ordering, transactions, filtering.

---

3. What are the key components of an Azure Storage queue?

- **Queue**: A named message container.
- **Message**: Max 64 KB of Base64-encoded text.
- **Visibility timeout**: Temporary hide period after retrieval.

---

4. How are messages added to and retrieved from a queue?
Add: `PutMessageAsync()`
Get: `GetMessagesAsync()`
Delete: `DeleteMessageAsync()` after processing.

---

5. What is the default behavior when retrieving messages from a queue?
Messages are hidden for the visibility timeout duration.
If not deleted within that time, they become visible again for reprocessing.

---

6. What is message visibility timeout and how does it affect processing?
Defines how long a retrieved message stays hidden.
If not deleted in time, it is returned to the queue for another attempt.

7. How do you handle message retries and poison messages?

Track dequeue count using `DequeueCount`.

Manually delete or move messages to a custom "poison" queue if `DequeueCount` exceeds threshold.

---

8. How do you create and manage queues using Azure CLI or SDK?

CLI:

```
az storage queue create --name myqueue --account-name mystorage
```

.NET SDK:

```
await queueClient.CreateIfNotExistsAsync();
await queueClient.SendMessageAsync("my message");
```

---

9. How do you secure access to Azure Queue Storage?

- **Shared keys**: Full access via storage account key.
- **SAS tokens**: Scoped, time-limited access.
- **RBAC**: Role-based access via Azure AD (Blob/Queue Contributor roles).

---

10. What are the message size limits and encoding requirements?

- Max message size: **64 KB** (48 KB when Base64 encoded by default).
- Messages must be **text-based** (UTF-8 or Base64).

---

11. How do you configure and use Base64 encoding with queue messages?

By default, SDK encodes messages as Base64.

Can opt out using:

```
new QueueClient(..., new QueueClientOptions { MessageEncoding = QueueMessageEncoding.None })
```

---

12. What are best practices for processing at scale with Queue Storage?

- Use multiple consumer instances.
- Tune visibility timeout per job duration.
- Monitor queue length for autoscaling triggers.

---

13. How do you integrate Azure Queue Storage with Azure Functions?

Use the `QueueTrigger` binding:

```
[FunctionName("ProcessQueue")]
public void Run([QueueTrigger("myqueue", Connection = "StorageConnection")] string message)
```

---

14. How do you monitor and troubleshoot Azure Queue Storage usage?

- Enable diagnostics in Azure Monitor.
- Use metrics: `QueueLength`, `DequeueCount`, `Ingress`, `Egress`.
- View activity logs and errors in Log Analytics if enabled.

---

15. What are common limitations of Azure Queue Storage compared to other messaging services?

- No message ordering guarantee.
- No native dead-letter queues.
- No message filtering or pub/sub.
- Limited throughput and features compared to Service Bus.