DUMPSARENA

Designing and Implementing a Microsoft Azure AI Solution

Microsoft AI-102

Version Demo

Total Demo Questions: 20

Total Premium Questions: 508

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

QUESTION NO: 1

You build a custom Form Recognizer model.

You receive sample files to use for training the model as shown in the following table.

Name	Type	Size
File1	PDF	20 MB
File2	MP4	100 MB
File3	JPG	20 MB
File4	PDF	100 MB
File5	GIF	1 MB
File6	JPG	40 MB

Which three files can you use to train the model? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. File1

B. File2

C. File3

D. File4

E. File5

F. File6

ANSWER: A C F

Explanation:

To train a custom Azure AI Document Intelligence (Form Recognizer) model, the training documents must be in a supported file format and within service limits. For custom model training, supported document types include PDF, JPEG/JPG, PNG, and TIFF. In addition, the file size must be within the service limit (commonly up to 50 MB per document), and the files must be accessible to the training process (for example, stored in Azure Blob Storage when using the service APIs).

Based on the table in the prompt, File1, File3, and File6 are the ones that meet the supported input format requirements (JPG/PNG/PDF/TIFF). The other files are in formats that aren't supported for training (for example, Office documents like DOCX/XLSX/PPTX or other non-image/non-PDF formats), so they can't be used directly as training inputs.

Why the others are wrong: even if they contain the same content, unsupported formats must be converted to a supported format (such as PDF or images) before they can be used for training.

References: Azure AI Document Intelligence overview, Custom model concepts and requirements.

QUESTION NO: 2

You build a chatbot that uses the Azure OpenAI GPT 3.5 model.

You need to improve the quality of the responses from the chatbot. The solution must minimize development effort.

What are two ways to achieve the goal? Each correct answer presents a complete solution. NOTE: Each correct answer is worth one point.

A. Fine-tune the model.

B. Provide grounding content.

C. Add sample request/response pairs.

D. Retrain the language model by using your own data.

E. Train a custom large language model (LLM).

ANSWER: B C

Explanation:

To improve response quality with minimal development effort, the best approaches are prompt-engineering and retrieval/grounding rather than training. **Providing grounding content** (often implemented as Retrieval Augmented Generation, or “on your data”) supplies the model with relevant, authoritative context at runtime, which reduces hallucinations and makes answers more accurate and consistent without changing the underlying model. This is typically done by retrieving passages from your documents (e.g., via Azure AI Search) and injecting them into the prompt.

Adding sample request/response pairs (few-shot examples) is another low-effort way to steer the model toward the desired tone, structure, and reasoning patterns. Examples in the prompt help the model imitate the format and level of detail you want, improving consistency without any training pipeline.

Fine-tuning can improve quality for narrow tasks, but it requires dataset preparation, evaluation, and ongoing maintenance—more effort than grounding and few-shot prompting. **Retraining** or **training a custom LLM** is not a practical or supported “minimal effort” approach for Azure OpenAI scenarios and is far more complex and costly.

References: Azure OpenAI on your data (grounding/RAG), Prompt engineering (including few-shot examples).

QUESTION NO: 3

You have an Azure subscription and 10,000 ASCII files.

You need to identify files that contain specific phrases. The solution must use cosine similarity. Which Azure OpenAI model should you use?

A. text-embedding-ada-002.

B. GPT-4.

C. GPT-35 Turbo.

D. GPT-4-32k.

ANSWER: A

Explanation:

To find files containing specific phrases using **cosine similarity**, you should use an **embeddings** model. Embeddings convert text into high-dimensional vectors where semantic similarity can be measured with distance metrics such as cosine similarity. You would typically embed (1) the target phrase(s) and (2) chunks of each ASCII file, then compute cosine similarity to retrieve the most relevant files/chunks.

Option A (text-embedding-ada-002) is an embeddings model designed exactly for this scenario (semantic search / similarity). In Azure OpenAI, embeddings are the recommended approach when you need vector representations and similarity scoring rather than text generation.

The other options (GPT-4, GPT-35 Turbo, GPT-4-32k) are **chat/completions** models optimized for generating and reasoning over text, not for producing embedding vectors for cosine similarity. While you could prompt them to "search," that would be inefficient, costly, and would not meet the explicit requirement to use cosine similarity over embeddings.

References: Azure OpenAI embeddings concepts, How to use embeddings in Azure OpenAI.

QUESTION NO: 4

You are developing an application that will use Azure Cognitive Search for internal documents.

You need to implement document-level filtering for Azure Cognitive Search.

Which three actions should you include in the solution? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Send Azure AD access tokens with the search request.

B. Retrieve all the groups.

C. Retrieve the group memberships of the user.

D. Add allowed groups to each index entry.

E. Create one index per group.

F. Supply the groups as a filter for the search requests.

ANSWER: C D F

Explanation:

To implement document-level filtering (security trimming) in Azure AI Search, you store authorization metadata in each document and then apply it as an OData filter at query time. First, you need to know which groups the caller belongs to (for example, Azure AD group object IDs). That's why retrieving the user's group memberships is required. Next, you must index that access control list per document (for example, a collection field like `allowedGroups` containing group IDs). Without this field in the index, there's nothing to filter on. Finally, every search request must include a filter that intersects the caller's groups with the document's allowed groups (for example, `allowedGroups/any(g: search.in(g, 'id1,id2'))`), ensuring unauthorized documents are excluded from results.

Sending Azure AD access tokens to the search endpoint is not, by itself, document-level filtering; it's about authenticating to the service (and many apps use query keys or managed identity instead). Creating one index per group is an anti-pattern that doesn't scale and complicates management. "Retrieve all the groups" is unnecessary; you only need the groups relevant to the current user/request.

References: Security trimming for Azure AI Search, Azure AI Search filters (OData)

QUESTION NO: 5

You are building a Conversational Language Understanding model.

You need to ensure that the model will support the following sample utterances: Set all the lights to on.

Turn off the lights in the living room.

What is the current thermostat temperature?

Lower the temperature of the thermostat by five degrees. Which three elements should you add to the model?

Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

A. a location Intent.

B. a change setting entity.

C. a device intent.

D. a change setting intent.

E. a query setting intent.

F. a device entity.

ANSWER: D E F

Explanation:

To support the four utterances, your CLU project needs to capture (1) what the user wants to do (intents) and (2) the key pieces of information needed to complete the action (entities). The utterances include both "change" actions (turning lights on/off; lowering thermostat) and a "query" action (asking for the current thermostat temperature). Therefore, you should include a **change setting intent** to route commands that modify a device state (for example, "Set all the lights to on" and "Lower the temperature…"), and a **query setting intent** to route informational requests (for example, "What is the current thermostat temperature?"). You also need a **device entity** to extract which device is being referenced (lights vs thermostat), which is essential for all sample utterances.

The **change setting entity** is not strictly required by the prompt because the "setting" can be modeled in different ways (for example, as separate entities like temperature/value, or as part of the utterance handled by downstream logic). Likewise, a **device intent** is not a good fit because "device" is typically an entity (a thing), while intents represent user goals (change vs query). A **location intent** is incorrect; "living room" is better modeled as a location entity if needed.

References: Conversational Language Understanding overview, Build a schema (intents and entities)

QUESTION NO: 6 - (HOTSPOT)

A1 You have an Azure OpenA1 resource named AH that hosts three deployments of the GPT 3.5 model. Each deployment is optimized for a unique workload.

You plan to deploy three apps. Each app will access AM by using the REST API and will use the deployment that was optimized for the apps intended workload.

You need to provide each app with access to AH and the appropriate deployment. The solution must ensure that only the apps can access AM.

What should you use to provide access to AM. and what should each app use to connect to its appropriate deployment? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

ANSWER:

Explanation:

To lock down an Azure OpenAI resource so that *only your applications* can call it, you want an authentication method that can be tied to an app identity and governed by Azure RBAC. API keys are shared secrets: once a key is copied, anything that has the key can call the resource, which doesn't really meet the "only the apps can access it" requirement. Azure OpenAI supports Microsoft Entra ID (Azure AD) authentication, where each app uses its own identity (service principal or managed identity) to request an access token and then calls the REST API with an `Authorization: Bearer <token>` header. That approach lets you grant only the required apps the proper role assignments on the Azure OpenAI resource and revoke access per app if needed.

For selecting the correct optimized model deployment, Azure OpenAI REST APIs route requests to a deployment by including the **deployment name** in the request URL path (for example, `/openai/deployments/{deployment-`

`name}/chat/completions`). The resource has a single endpoint, and the deployment is chosen by name, not by a separate per-deployment endpoint or “deployment type.”

References: Use managed identities with Azure OpenAI and Azure OpenAI REST API reference (deployment in URL).

QUESTION NO: 7

You have an Azure Cognitive Search solution and an enrichment pipeline that performs Sentiment Analysis on social media posts.

You need to define a knowledge store that will include the social media posts and the Sentiment Analysis results.

Which two fields should you include in the definition? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. storageContainer.

B. tables.

C. storaeeConnectionString.

D. files.

E. objects.

ANSWER: C E

Explanation:

In Azure Cognitive Search, a knowledge store is configured on a skillset and persists enriched content to Azure Storage in one or more projections. The knowledge store definition must include a **storageConnectionString** so the indexer/skillset can write to the target storage account. Without it, nothing can be persisted.

To store both the original social media posts and the sentiment analysis output, you typically project the enriched “document” shape into **objects** (JSON documents in Blob storage). Object projections are designed to capture the enriched tree (for example, the post text plus the sentiment score/label) in a structured JSON form that’s easy to consume downstream.

Options like **tables** and **files** are also valid projection types in general, but they are not required for this scenario as stated. **storageContainer** is not a top-level knowledge store field; containers are specified within projections (for example, an object projection can specify a container name). Therefore, the two key fields to include are the connection string and an objects projection.

References: Microsoft Docs: Knowledge store in Azure AI Search, Microsoft Docs: Knowledge store projections

QUESTION NO: 8

You are processing text by using the Azure AI Language service.

You need to identify music band names in the text. The solution must minimize development effort. What should you use?

A. Key phrase extraction

B. Conversational Language Understanding (CLU)

C. Entity linking

D. Custom named entity recognition (NER)

ANSWER: C

Explanation:

To identify music band names with minimal development effort, use **Entity linking**. Entity linking in Azure AI Language detects named entities in text and then disambiguates and links them to well-known entries in a knowledge base (for example, Wikipedia). For band names like “Coldplay” or “Metallica,” this often works out-of-the-box because these are commonly known entities, and the service can return both the mention and a stable identifier/URL for the linked entity. That means you avoid building and training a custom model and can still get high-quality, normalized results.

Why the other options are wrong: Key phrase extraction returns important terms but doesn’t reliably identify proper-noun entities or classify them as bands. CLU is intended for intent classification and entity extraction in conversational apps and typically requires authoring/training, which is more effort than needed here. Custom NER could be made to detect “BandName” entities, but it requires labeling data and training/deploying a custom model—higher effort than entity linking for common bands.

References: Azure AI Language entity linking overview, Named entity recognition overview.

QUESTION NO: 9

You plan to provision a QnA Maker service in a new resource group named RG1.

In RG1, you create an App Service plan named AP1.

Which two Azure resources are automatically created in RG1 when you provision the QnA Maker service? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Language Understanding

B. Azure SQL Database

C. Azure Storage

D. Azure Cognitive Search

E. Azure App Service

ANSWER: D E

Explanation:

When you provision a QnA Maker resource (the “QnA Maker” Cognitive Service), Azure deploys supporting resources into your subscription to host both the knowledge base content and the runtime endpoint. Specifically, QnA Maker uses **Azure Cognitive Search** to index the knowledge base and enable fast retrieval at query time. In addition, QnA Maker deploys an

Azure App Service (web app) to host the runtime that serves your published knowledge base over HTTPS. Even if you pre-create an App Service plan (AP1), the QnA Maker provisioning still creates the App Service resource (the web app) that runs on a plan (often the one you select/provide), so the App Service is still an automatically created resource in the resource group.

The other options are not automatically created as part of standard QnA Maker provisioning: **Language Understanding (LUIS)** is a separate service you may integrate with but isn't created by default. **Azure SQL Database** isn't used for QnA Maker knowledge base storage in this model (Search is the primary index). **Azure Storage** may be used in some architectures, but it is not one of the core resources automatically created for QnA Maker in the typical provisioning flow described in Microsoft documentation.

References: Microsoft Docs: Set up QnA Maker service in Azure, Microsoft Docs: QnA Maker overview.

QUESTION NO: 10

What is used to define a query in a stream processing jobs in Azure Stream Analytics?

A. SQL.

B. XML.

C. YAML.

D. KOL.

ANSWER: A

Explanation:

In Azure Stream Analytics (ASA), the transformation logic for a stream processing job is defined using the Stream Analytics Query Language, which is a SQL-like language. You write a query (for example, using `SELECT`, `FROM`, `WHERE`, windowing functions like `TumblingWindow`, and joins) to read from one or more inputs and write the results to an output. This is why **SQL** is the correct choice: ASA jobs are configured around inputs/outputs plus a SQL-based query that performs the real-time processing.

The other options don't apply to defining ASA queries. **XML** and **YAML** might be used in other contexts (configuration files, pipelines, IaC templates), but they are not the language used to author the ASA streaming query itself. **KQL** (often written as KQL, not "KOL") is the Kusto Query Language used primarily with Azure Data Explorer and Log Analytics; it is not the query language for Azure Stream Analytics jobs.

References: Stream Analytics query language reference, Introduction to Azure Stream Analytics.

QUESTION NO: 11

You have a text-based chatbot.

You need to enable content moderation by using the Text Moderation API of Content Moderator. Which two service responses should you use? Each correct answer presents part of the solution NOTE: Each correct selection is worth one point.

A. the adult classification score

B. optical character recognition (OCR)

C. personal data

D. text classification

E. the racy classification score

ANSWER: A E

Explanation:

For moderating a text-based chatbot with the Content Moderator *Text Moderation* API, the key responses you typically use are the classification scores that indicate whether the submitted text contains adult or racy content. The API returns these as part of the classification result (often under a “Classification” section), and they’re designed to be used as thresholds in your app logic (for example, block/flag when the score exceeds a configured value). Therefore, the **adult classification score** and the **racy classification score** are the two most relevant service responses for content moderation in a text chat scenario.

OCR is not applicable here because OCR is used to extract text from *images*, not to moderate already-textual chat messages. “Personal data” detection can be useful for privacy/PII scenarios, but it’s not one of the core moderation classification scores asked for in this question, and it isn’t the primary mechanism for adult/racy content moderation. “Text classification” is too generic; the actionable outputs for moderation are specifically the adult and racy scores returned by the service.

References: Text moderation API (Content Moderator), Azure AI Content Moderator documentation.

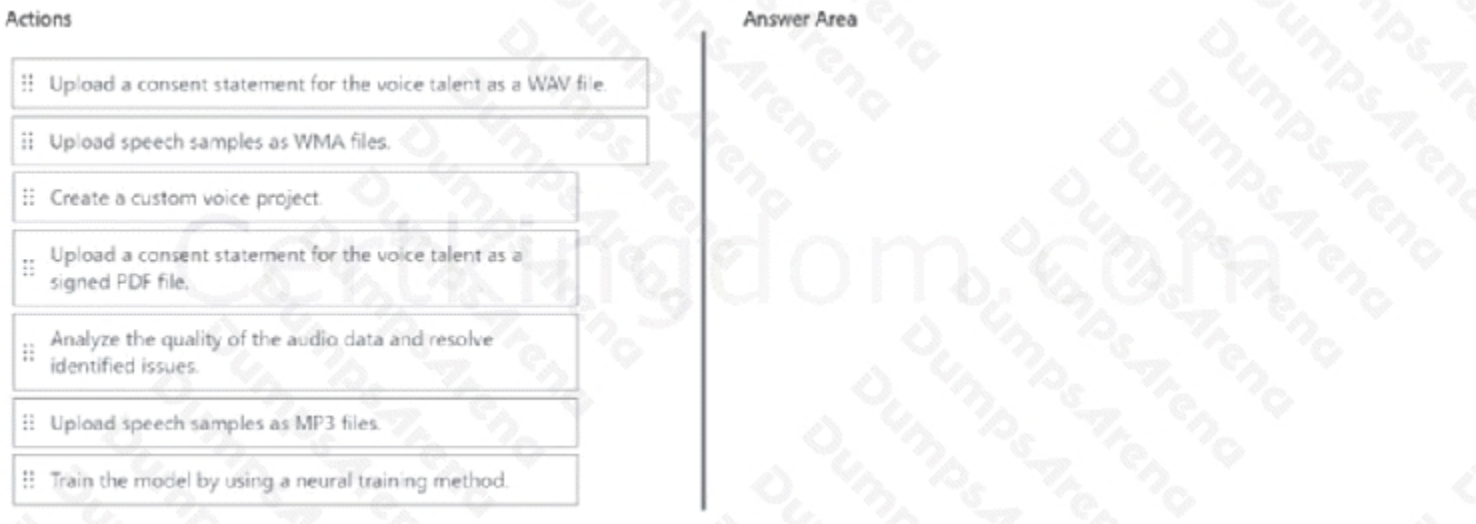
QUESTION NO: 12 - (DRAG DROP)

You are building a phone call handling solution that will use the Azure AI Speech service and a custom neural voice.

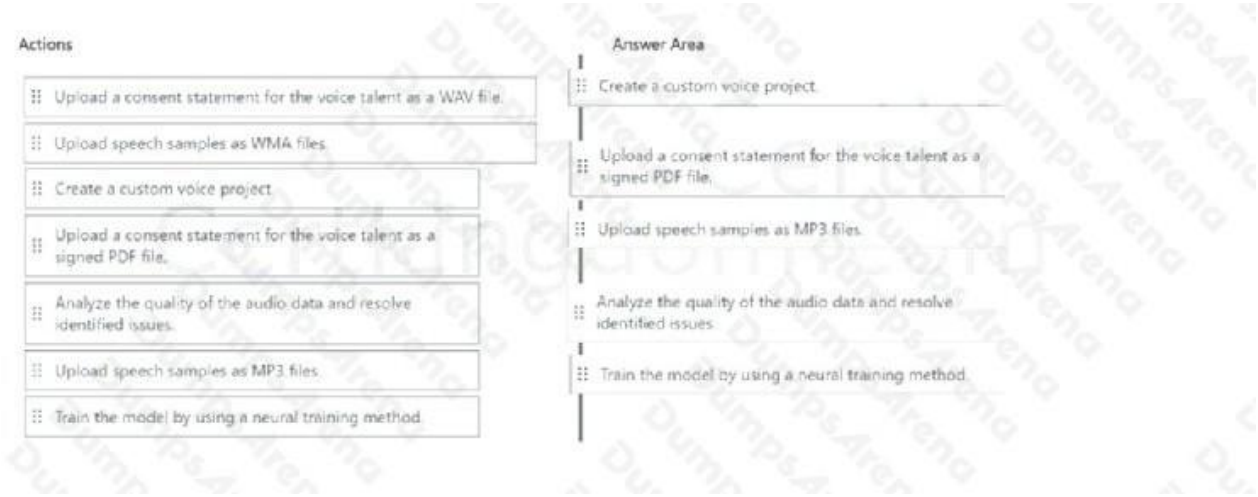
You need to create a custom speech model.

Which five actions should you perform in sequence from Speech Studio? To answer, move the

appropriate actions from the list of actions to the answer area and arrange them in the correct order.



ANSWER:



Explanation:

In Speech Studio, creating a Custom Neural Voice follows a pretty standard workflow: you set up the project container first, satisfy the compliance requirement (voice talent consent), bring in your training data, validate it, and then train. The first step is to **create a custom voice project** because everything else (consent, datasets, training jobs) is attached to a specific project in Speech Studio.

Next, Custom Neural Voice has strict responsible AI and legal requirements. You must provide proof that the voice talent agreed to have their voice used to build a synthetic voice. In Speech Studio this is handled by uploading a **signed consent statement**, which is typically provided as a **signed PDF**. A “consent statement as a WAV file” is not the correct compliance artifact for CNV, so that option is a distractor.

After consent is in place, you upload the **speech samples** that will be used for training. In this question, the correct choice is **MP3** (not WMA). WMA is not a standard/expected training upload format in Speech Studio custom voice workflows, and exam items commonly test that you choose the supported/expected audio format option rather than legacy formats.

Once the audio is uploaded, you should **analyze the quality of the audio data and resolve identified issues**. Speech Studio provides dataset validation/analysis (for example, checking audio quality, alignment, and other issues) and you’re expected to fix problems before training to avoid poor model quality or failed training runs.

Finally, you **train the model by using a neural training method**, which is the step that actually produces the custom neural voice model you can later deploy and use.

References: Custom Neural Voice documentation and Speech Studio overview.

QUESTION NO: 13

You need to enable speech capabilities for a chatbot.

Which three actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Enable WebSockets for the chatbot app.

B. Create a Speech service.

C. Register a Direct Line Speech channel.

D. Register a Cortana channel.

E. Enable CORS for the chatbot app.

F. Create a Language Understanding service.

ANSWER: A B C

Explanation:

To add speech capabilities to a bot using Azure AI Speech, you typically “voice-enable” it through the Direct Line Speech channel. That approach requires (1) an Azure AI Speech resource to provide speech-to-text and text-to-speech, and (2) enabling the Direct Line Speech channel on the bot so the Speech SDK can connect and stream audio. In addition, the bot must support WebSocket connections because Direct Line Speech uses WebSockets for real-time, low-latency audio and message streaming; in Azure App Service this means turning on the WebSockets setting for the bot’s web app.

Registering a Cortana channel isn’t required for generic speech enablement; Cortana integration is a separate, legacy-style channel choice and not the standard path for Speech SDK/Direct Line Speech. Enabling CORS is not a required step for Direct Line Speech (the Speech SDK connects to the Speech service and Direct Line Speech endpoint rather than relying on browser cross-origin calls to your bot). Finally, Language Understanding (LUIS/CLU) can improve intent recognition, but it’s optional and not required just to add speech input/output.

References: Voice-enable your bot with the Speech SDK (Direct Line Speech), Azure App Service settings: WebSockets

QUESTION NO: 14

You are building a Language Understanding model for an e-commerce platform.

You need to construct an entity to capture billing addresses.

Which entity type should you use for the billing address?

A. machine learned

B. Regex

C. geographyV2

D. Pattern.any

E. list

ANSWER: A

Explanation:

Use a **machine learned** entity for a billing address. Addresses are typically variable, multi-part, and highly dependent on context (street, unit, city, state/province, postal code, country). In LUIS, machine learned entities are designed to learn from labeled examples and can be composed as a parent entity with sub-entities (for example, Street, City, Region, PostalCode, Country). This approach is more robust than trying to force all possible address formats into a single pattern.

A **Regex** entity is best when the text follows a predictable, consistent format (like an order number or SKU). Real-world addresses vary too much across countries and user input styles, so regex tends to be brittle and will miss many valid inputs. **geographyV2** is a prebuilt entity for locations (cities, countries, etc.) but it won't reliably capture a full billing address string. **Pattern.any** is only a placeholder used inside pattern templates, not a general-purpose address extractor. A **list** entity is for a closed set of known values and synonyms, which doesn't fit addresses.

References: LUIS entity types, Machine learned entities.

QUESTION NO: 15

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You create a web app named app1 that runs on an Azure virtual machine named vm1. Vm1 is on an Azure virtual network named vnet1.

You plan to create a new Azure Cognitive Search service named service1.

You need to ensure that app1 can connect directly to service1 without routing traffic over the public internet.

Solution: You deploy service1 and a private endpoint to vnet1.

Does this meet the goal?

A. Yes

B. No

ANSWER: A

Explanation:

Yes. Azure Cognitive Search supports Azure Private Link, which lets you create a *private endpoint* for the search service in a virtual network subnet. When you deploy service1 and create a private endpoint in vnet1, vm1 (and therefore app1) can reach the search service using a private IP address from vnet1. This keeps traffic on the Microsoft backbone and avoids routing over the public internet, meeting the requirement for direct, private connectivity.

In practice, you also need correct DNS resolution so that the search service's public FQDN resolves to the private endpoint IP from within vnet1 (typically via an Azure Private DNS zone such as `privatelink.search.windows.net` linked to the VNet). Additionally, you would usually restrict public network access on the search service to ensure clients must use the private endpoint. But the core solution described—deploying the service and a private endpoint to vnet1—is the right approach to achieve private connectivity.

Option B is incorrect because a private endpoint is specifically designed to avoid public internet routing for supported Azure PaaS services, including Azure Cognitive Search.

References: Azure AI Search private endpoints (Private Link), Azure Private Endpoint overview.

QUESTION NO: 16

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets might have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You create a web app named app1 that runs on an Azure virtual machine named vm1. Vm1 is on an Azure virtual network named vnet1.

You plan to create a new Azure Cognitive Search service named service1.

You need to ensure that app1 can connect directly to service1 without routing traffic over the public internet.

Solution: You deploy service1 and a public endpoint, and you configure an IP firewall rule.

Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No. Using a **public endpoint** for Azure Cognitive Search means the service is still reachable via the public internet, even if you restrict access with an IP firewall rule. An IP firewall rule limits which public source IPs can connect, but the traffic still targets the public endpoint and traverses public routing. That does not satisfy the requirement to connect “directly … without routing traffic over the public internet.”

To meet the goal, you should use **Azure Private Link (private endpoint)** for Azure Cognitive Search. A private endpoint assigns a private IP address from your virtual network (or a peered VNet) to the search service, so app1 on vm1 can resolve and reach service1 over the Azure backbone using private addressing. This keeps traffic off the public internet and allows you to disable/limit public network access as needed.

Option A is incorrect because firewalling a public endpoint is not the same as private connectivity. Option B is correct because the proposed solution does not eliminate public internet routing.

References: Use a private endpoint for Azure AI Search, Azure Private Link overview.

QUESTION NO: 17

You have an Azure subscription that contains an Azure AI Document Intelligence resource named Aldoc1 in the SO tier.

You have the lies shown in the following table.

Name	Format	Password-locked	Size (MB)
File1	JPG	No	400
File2	PDF	No	250
File3	PNG	Yes	180
File4	XLSX	No	900
File5	PDF	Yes	160

You need to train a custom extraction model by using AIdoc1. Which files can you upload to Document Intelligence Studio?

A. File1, and File2 only.

B. File2, File4, and File5 only.

C. File1, File2, and File4 only.

D. File1, and File5 only.

E. File1, File2. File3, File4, and File5.

ANSWER: C

Explanation:

To train a **custom extraction** model in Azure AI Document Intelligence (Form Recognizer) using **Document Intelligence Studio**, the training data you upload must be in a supported document format for custom models. For custom extraction, Studio supports common document/image types such as **PDF** and image formats like **JPG/JPEG/PNG/TIFF** (and related image types). Files that are not supported for custom training uploads (for example, Office documents like Word/Excel/PowerPoint, or other non-supported formats) can't be used directly in Studio; they must be converted to a supported format (typically PDF or images) first.

Based on the table in the prompt, **File1, File2, and File4** are the ones that match supported upload types for custom extraction training in Studio, so option **C** is correct.

The other options are incorrect because they either omit one of the supported files (A, D), include files that are not supported for custom model training uploads (B, E), or claim all files are uploadable when at least one is not (E).

References: Document Intelligence Studio, Custom models in Document Intelligence (supported inputs)

QUESTION NO: 18 - (HOTSPOT)

HOTSPOT

You are developing a streaming Speech to Text solution that will use the Speech SDK and MP3 encoding.

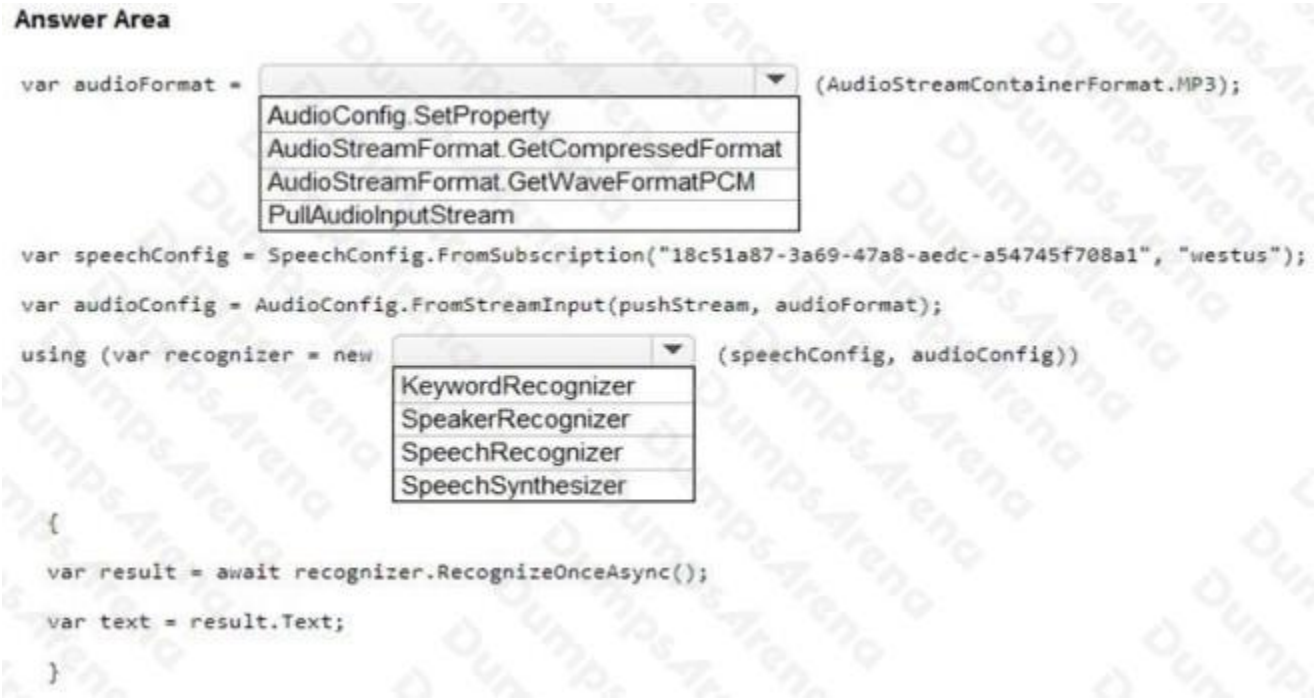
You need to develop a method to convert speech to text for streaming MP3 data.

How should you complete the code? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

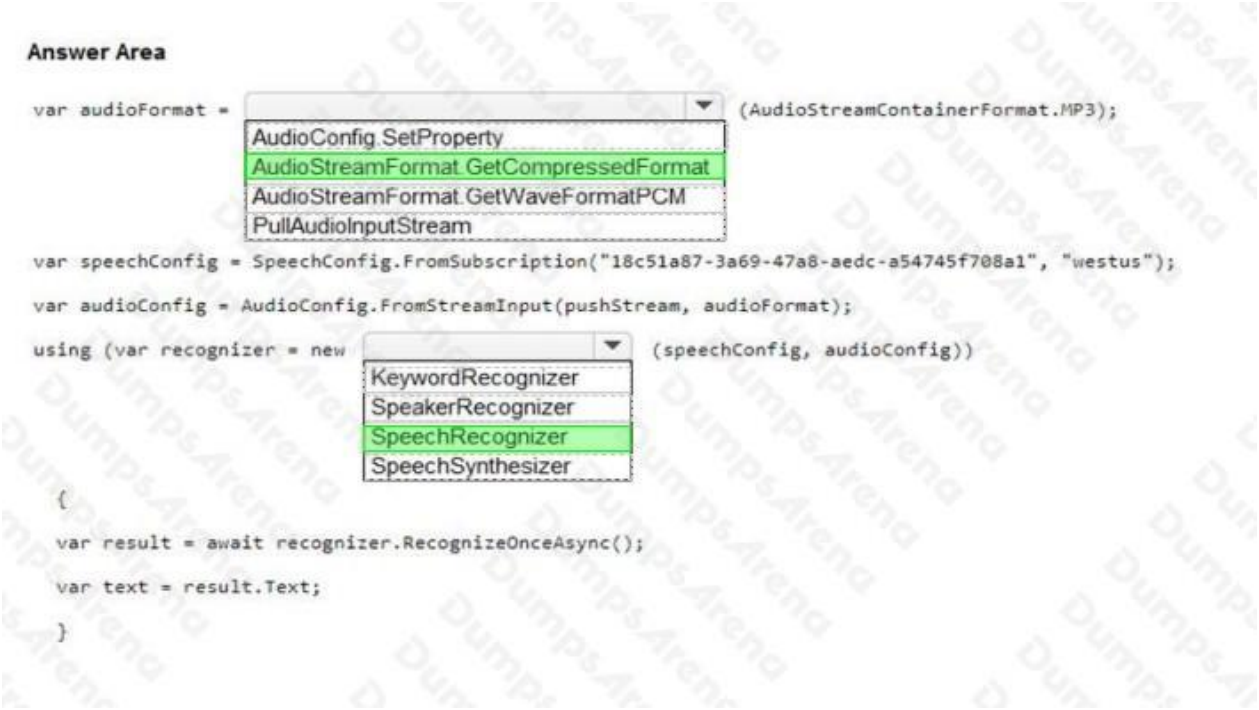
Hot Area:

Answer Area



ANSWER:

Answer Area



Explanation:

To stream MP3 audio into the Speech SDK, you need to tell the SDK that the incoming bytes are a *compressed/container* format (MP3), and then use the standard speech-to-text recognizer class.

For the first dropdown, the key detail is the method signature shown in the snippet: it passes `(AudioStreamContainerFormat.MP3)` into the selected method. In the Speech SDK, that pattern corresponds to `AudioStreamFormat.GetCompressedFormat(AudioStreamContainerFormat.MP3)`. This is the supported way to describe an MP3 (or other container/codec) stream when you're pushing bytes via a stream (for example, with a `PushAudioInputStream`). Options like `AudioStreamFormat.GetWaveFormatPCM` are specifically for raw PCM (uncompressed) audio and don't apply to MP3 container data.

For the second dropdown, the code is creating a recognizer with `(speechConfig, audioConfig)` and then calling `RecognizeOnceAsync()` to produce text. That is the standard Speech-to-Text flow, which uses `SpeechRecognizer`. The other choices are for different workloads: `SpeechSynthesizer` is text-to-speech, `SpeakerRecognizer` is for speaker identification/verification, and `KeywordRecognizer` is for keyword spotting.

Microsoft's guidance for codec/compressed audio input streams shows this exact approach: define the compressed stream format with `GetCompressedFormat` and then run recognition with `SpeechRecognizer`. See Use codec compressed audio input streams (Speech SDK) and the SpeechRecognizer class reference.

QUESTION NO: 19

You are examining the Language service output of an application.

The text analyzed is: Our tour guide took us up the Space Needle during our trip to Seattle last week.

The response contains the data shown in the following table.

Text	Category	ConfidenceScore
Tour guide	PersonType	0.45
Space Needle	Location	0.38
Trip	Event	0.78
Seattle	Location	0.78
Last week	DateTime	0.80

Which Language service API is used 10 analyze the Text?

A. Entity Linking.

B. Named Entity Recognition.

C. Key Phrase Extraction.

D. Sentiment Analysis.

ANSWER: B

Explanation:

The correct API is **Named Entity Recognition (NER)**. In Azure AI Language, NER returns a list of entities detected in the text (for example, *Space Needle* as a location/landmark and *Seattle* as a location), along with metadata such as the entity *category*, *offset*, *length*, and often a *confidence score*. That kind of structured output (entities + categories + positions in the text) matches what you'd expect from the NER feature.

Entity Linking is different: it not only detects entities but also links them to a knowledge base entry (typically including a data source like Wikipedia, an ID, and a URL). If the table/output doesn't show linked IDs/URLs, it's not entity linking. **Key**

Phrase Extraction returns key phrases (strings) without entity categories and offsets in the same way. **Sentiment Analysis** returns sentiment labels/scores (positive/neutral/negative) at document/sentence level, not entity lists.

References: Named entity recognition in Azure AI Language, Entity linking in Azure AI Language.

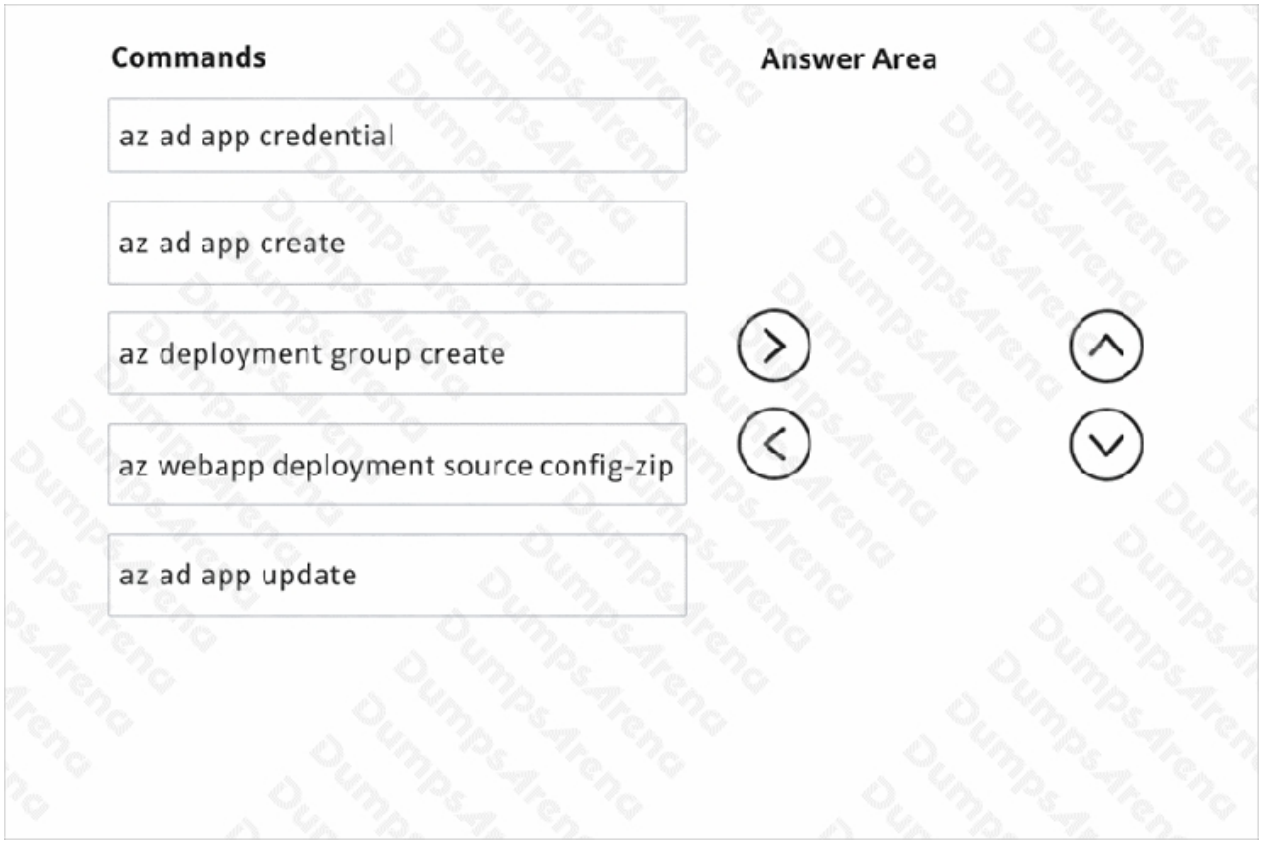
QUESTION NO: 20 - (DRAG DROP)

You Build a bot in JavaScript.

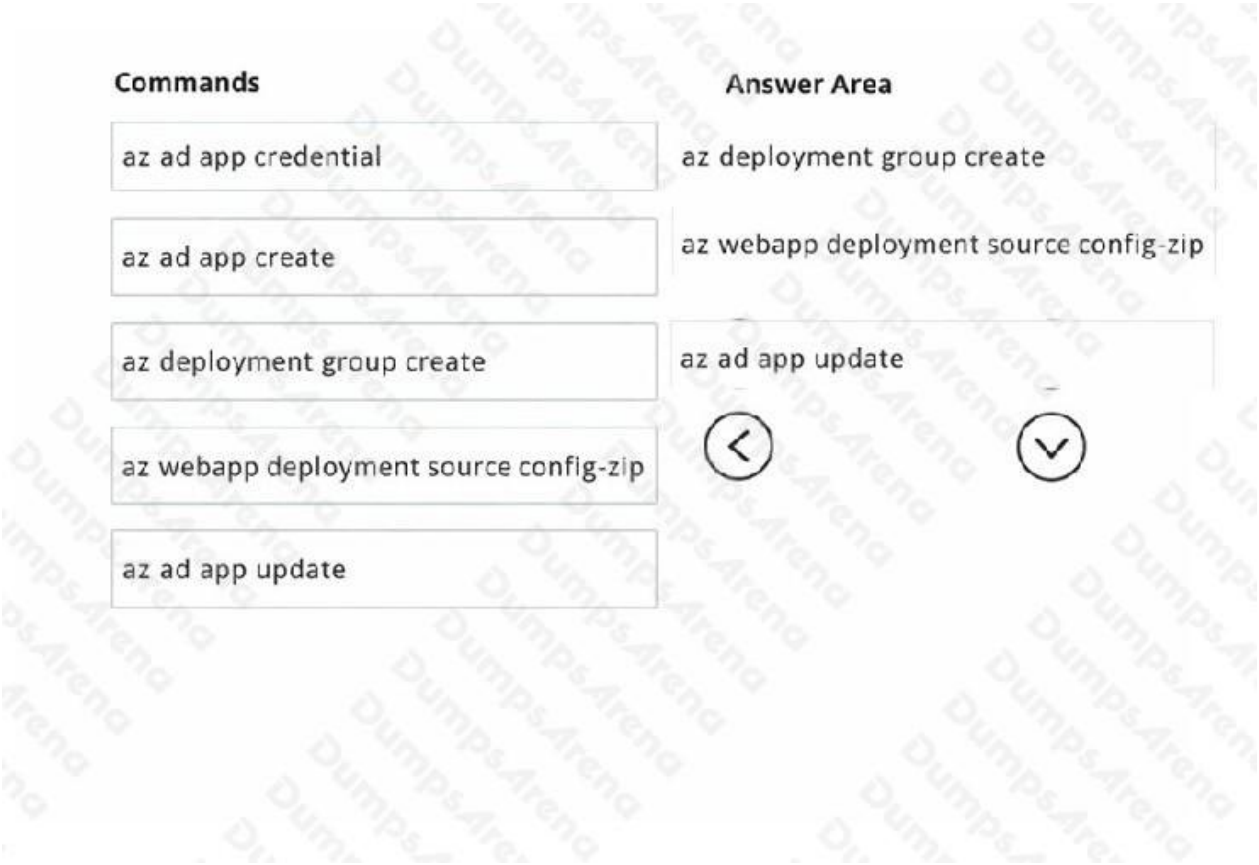
From the Azure Command-Line interface (CLI), you run the following command. az bot prepare-deploy

You need to deploy the bot to Azure.

Which three Azure CLI commands should you run in sequence? To answer, move the appropriate commands from the list of commands to the answer area and arrange them in the client order.



ANSWER:



Explanation:

Because you already ran `az bot prepare-deploy`, you've done the "packaging" step that generates the deployment assets (most importantly an ARM template and parameter files under a deploymentTemplates folder, and a zip-ready structure for publishing). The next steps are the classic Azure sequence: first create the Azure resources, then publish the code, then finalize any app registration settings that must match the deployed endpoint.

The first command should be `az deployment group create` because that's the Azure CLI command used to deploy an ARM template into a resource group. In the Bot Framework workflow, this provisions the App Service plan, Web App, and bot-related resources defined by the template produced by prepare-deploy. Microsoft documents ARM template deployment with this command here: Deploy resources with ARM templates and Azure CLI.

After the infrastructure exists, you deploy your bot code to the created Web App. The correct command for zip deployment to App Service is `az webapp deployment source config-zip`. This pushes your packaged bot code to the web app so the runtime can serve the bot endpoint. Microsoft documents zip deployment for web apps here: Deploy to Azure App Service by using ZIP deploy.

Finally, you may need to update the Azure AD app registration settings so they align with the deployed endpoint/URLs (for example, reply URLs/redirect URIs depending on the bot registration/auth scenario). From the provided options, `az ad app update` is the command that performs that configuration update. See: az ad app update. Putting these together yields the required three commands in the correct order.