ID: .....

Name: .....

AY2022-23 - Winter – 2nd In-Semester Examination

IT457 – Cloud Computing

Max. Marks: 20	Time: 1 hour

Instructions:

1. *This exam has 13 questions in total. All questions are compulsory.*
2. *All questions are to be answered in the question paper itself.*
3. *For objective type questions,* ***circle (like, ©) the choice(s)*** *to mark your answer.*
4. *There could be* ***one or more*** *correct answers to the objective type questions. Marks will be awarded only if all correct answers are circled.*
5. *Write your ID on* ***all pages*** *of Question paper.*
6. *Q1 to Q10 are of 1 mark each. Q11 and Q12 are of 3 marks each, and Q13 is of 4 marks.*

---

1. To decide appropriate AWS EFS throughput modes, it is important to know average-to-peak ratio of the EFS. How can this average-to-peak ratio be found/calculated?
   i. It is pre-set when AWS EFS is configured
   ii. It is calculated by comparing MeteredIOBytes metric to PermittedThroughput metric
   iii. It is calculated using AWS Cloudwatch metrics
   iv. It is actually a constant value of 5% ensured by AWS EFS

2. The annualized-failure rate of Azure managed disks is
   i. 0.1%
   ii. 0%
   iii. 0.01%
   iv. 0.001%

3. The ask is to upload a large file, say a 4k video file of size 2TB, to Azure block blob. Which of the following ways it can be achieved?
   i. Use PutBlob to upload the file in one go, assuming 2TB is within the service limit. No further action needed as it is committed automatically.
   ii. Split the file into smaller chunks, upload all chunks using PutBlock in parallel and then use PutBlockList at last to commit the uploaded blocks by providing the list of block IDs.
   iii. Split the file into smaller chunks, using MD5 hash to track IDs of the blocks. Use PutBlock to upload the blocks in parallel and use PutBlockList at last to provide list of blocks to commit to blob storage.

iv. Split the file into smaller chunks, using MD5 has to track IDs of the blocks. Use PutBlock and PutBlob in parallel to commit the blocks in parallel and avoid PutBlockList call which could be a bottleneck.

4. AWS Block Public Access feature of S3 storage helps control the access to S3 buckets by applying access policies at bucket level.
   Suppose that a bucket owned by "Account-1" has a policy that contains the following:
   A statement that grants access to AWS CloudTrail (which is an AWS service principal)
   A statement that grants access to account "Account-2"
   A statement that grants access to the public, for example by specifying "Principal": "*" with no limiting Condition.
   With this policy in place and RestrictPublicBuckets setting of Block Public Access feature enabled,
   i. AWS S3 allows public access to the bucket owned by “Account-1” because of the 3rd statement.
   ii. AWS S3 allows access only by AWS CloudTrail because of the 3rd statement and RestrictPublicBuckets being enabled
   iii. AWS S3 allows access only by “Account-2” because of the 2nd statement that grants access to “Account-2” explicitly
   iv. AWS S3 allows access to AWS CloudTrail and “Account-2” because of statement 1st and 2nd

5. In Azure, multi-region deployment will result in storage redundancy of different types. If you want to redirect, GET REST requests to secondary region and remaining requests to primary region, this can be achieved using which of the following?
   i. LRS
   ii. RA-GRS
   iii. GRS
   iv. ZRS

6. How does AWS EBS + AWS EC2 combination provides encryption for data-at-rest and data-in-transit?
   i. Create EBS as encrypted volume providing automatic encryption for data-at-rest. When it is attach it to EC2, the encryption occurs on server that hosts EC2 instance providing encryption for data-in-transit from EC2 to EBS volume.
   ii. All EBS volumes are encrypted by default, hence AWS EBS provides zero config encryption of data-at-rest. Attaching the EBS to EC2 instance allows EC2 to write data to encryption-enable volume ensuring data-in-transit is encrypted.
   iii. AWS EBS and EC2 both support encryption by default and hence both data-in-rest and data-in-transit are encrypted with the user having to worry about it.
   iv. AWS EBS and EC2 do not support encryption and hence the user needs to provide custom encryption code to ensure data-at-rest and data-in-transit are encrypted.

7. A CDN network has the ability to cache files closer to end users to speed up delivery of static files. However, with dynamic web applications, caching that content in edge locations isn't possible because the server generates the content in response to user behavior. How can Azure CDN be configured for this dynamic content delivery?

i. Enable Dynamic Site Acceleration feature in Azure CDN using the probe-path route optimization.
ii. Use Azure CDN from Akamai and configure Dynamic Site Acceleration.
iii. Use Azure Front Door service from Microsoft for Dynamic Site Acceleration
iv. Azure CDN is not suitable for dynamic website content, we should switch to more capable service like Azure Blob Storage that can directly connect with the web server.

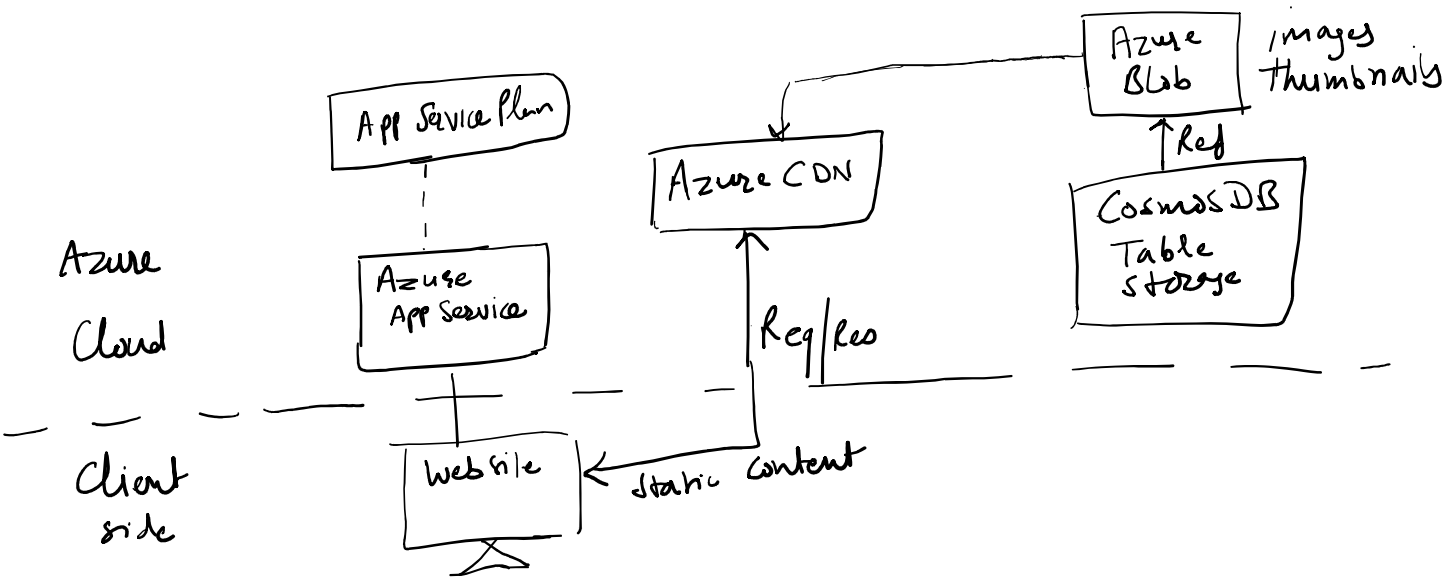
8. Which of the following is/are SQL database services?
   i. AWS RDS
   ii. Azure CosmosDB
   iii. AWS Aurora
   iv. AWS DynamoDB

9. For a SaaS application (like Gmail) where there is a database created for each user, but the usage pattern for the DB is unpredictable, with infrequent spikes (while receiving emails with attachments, or some similar activity during the day). If you were to use Azure SQL Database PaaS service for this application, how will you configure it?
   i. Single database per user deployed as single DB instance. Because of unpredictability, it is safe to use single database per user.
   ii. This scenario is fit for a NoSQL solution, hence use Managed instance DB for the app and migrate to Azure CosmosDB in the next upgrade cycle.
   iii. Managed instance DB per user because of the unpredictability and occasionally large loads.
   iv. Single database per user provisioned through an elastic pool. Unpredictability and occasional spike pattern suites elastic pool usage.

10. The requirement is to store user profile data for a web site into a NoSQL option. User profile contains name, mobile number, address, and a comma separated list two or more emails. Name of the user may or may not contain middle name. The sub fields in address could also be varied. Which NoSQL DB type is most appropriate for this kind of requirement?
    i. Key-Value pair table
    ii. Document DB
    iii. Wide-column Store
    iv. Graph Data

11. A website needs to be developed for cloud. The primary use case for this website is to show pleasing and beautiful pictures/photos of various categories to users. Categories are – Nature, Space, Instruments, Society, Cars, History. The home page is beautifully laid out static page with some pictures thumbnails from various categories. User can click on a single picture to see high resolution version of that picture in a separate window. User can click a category and then all pictures under that category are shown in a grid of thumbnails. Again, user can click a thumbnail and see its high-resolution picture. Though there could be other scenarios, but for this question let us limit it to this only – home page with categories and some thumbnails under each category, click on a thumbnail, selection of a category, show pictures under that

category as grid. Describe how would you have such static content service website on cloud (assume Azure) using the most appropriate solution.



App Service – will host the website that serves initial content.

App Service Plan – required for creating an app service

The client side will have URLs that point to the closest CDN. The images are uploaded to the Blob strorage service and their URLs are stored in Azure Table Storage in CosmosDB service.

Azure Table Storage tables:

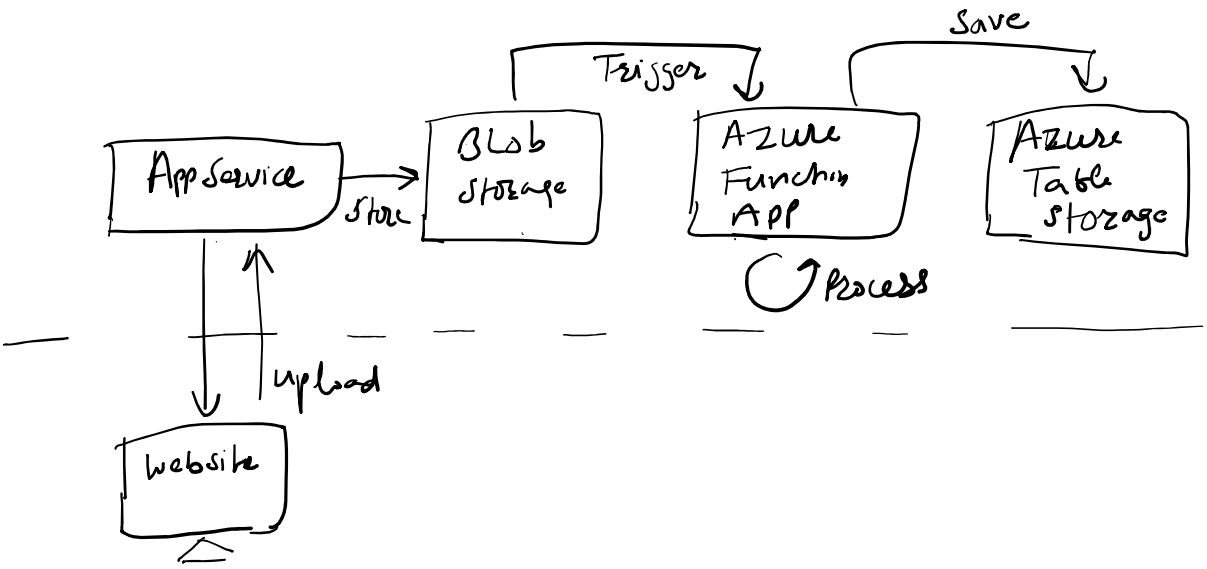
Table name: Images

Partition Key: Category

Row Key: ImageID_HiRes (or Thumb)

ID: .....

12. A website is going to be used by a company for its field agents who will upload large excel reports from their devices/laptops. These excel files will be processed as batch operations on the server side to extract data and store in a db. Rest of the website is simple single page app that lets the field agents login and do their work related activities. Describe an architecture suitable to deploy this type of web application. The data is stored in DB, but you do not need to go into further details of the db itself (like, SQL or NoSQL and schema etc).



App service – hosts website and serves initial pages.

Users upload data to blob storage using the API exposed by the App Service. The controller/back-end code stores the excel files in a blob storage and puts their URLs in Azure Table Storage. Azure Function app is the back-end processor that works on the excel files uploaded to blob storage. As a file is uploaded, the Azure Function app instance is triggered. It fetches the excel file from blob and processes it. It then stores the processed/extracted data into Azure Table Storage.

13. Design an Azure Table Storage database solution for an application called “Collaborators” that covers following scenarios of this social media related app.
    In this app the users register by providing their email, phone number and country as inputs. Then user can create her/his profile where artifacts like photos, certificates, awards etc. can be uploaded in addition to textual information about the profile like Name, Work, Education etc. Then, there is set of interests that a user can opt in. Eg. Automobiles, IT Technology, Gardening, Music, Movies, Games – the list of interests is pre-baked, and user only chooses from the given interests. Should new categories of interests need be added to app, it is desired that app should not require an update on users’ device. Based on common interests, a user is shown list of potential collaborators. From this list user can send connection request to other users and connect with each other as collaborator pairs after the other person accepts the request. A user, as you can expect, can see list of collaborator request (and accept/reject them) as well as current collaborators.

Note: The database design should be able to handle the above scenarios only. Do not imagine more scenarios and do not complicate mentioned scenarios. Eg. You do not need to handle how user will delete or ‘un-connect’ with collaborators. You need to design only the DB solution, not the whole app.

Table: Users

Partition-key: Country

Row-key: email

Data/Value: JSON of values such as name, phone number, URLs to docs and pics

Table: Interests

Partition-key: Category

Row-key: email

Table: ConnectionRequests

Patition-key: email (ie user ID)

Row-key: email

Table: Connections

Partition-key: Interest Category

Row-key: email - email

Use graph db for storing the connections is also possible

Table: IDs

ID: .....

Partition-key: IDs

Row-key: email

Data: password