MICTSETA

118699 CLOUD ADMINISTRATOR

EISA - MEMO

KNOWLEDGE

Cloud Computing Scenarios and Model Answers - A

1. Cloud Management Service

1.1 Cloud Infrastructure and Cloud-Related Services

StreamMedia Inc. is launching a video streaming platform (similar to YouTube) and expects 100,000 concurrent users immediately. They have identified four specific technical bottlenecks that standard web hosting cannot handle:

1. **Uploads:** Users upload raw 4K video files (up to 50GB each) which must be converted into 5 different formats (4K down to 360p). This process currently crashes their standard web servers.
2. **Global Latency:** Users in Asia and Europe report buffering because the servers are located in the US.
3. **Database Scale:** They expect millions of user comments and "likes" per hour, which is overwhelming their traditional SQL database.
4. **Recommendations:** They need to suggest new videos based on viewing history, which requires heavy mathematical processing.

Identify the specific infrastructure components required, focusing on the **Compute** types needed for video conversion vs. recommendations, and the **Network** components to solve the buffering issues.

Model Answer:

Cloud Infrastructure Components:

1. Compute Resources:

- **Video Transcoding Instances:** High-CPU Virtual Machines (e.g., 32 vCPU) dedicated to converting uploaded videos. These should use **Auto-Scaling** to handle bursts of uploads.
- **Recommendation Engine: GPU-enabled instances** specialised for machine learning/mathematical processing.
- **Application Servers:** Standard General Purpose instances for managing user logins and the website interface.

2. Storage Infrastructure:

- **Object Storage (Hot/Standard):** For storing the raw uploaded video files and the converted versions.
- **Block Storage (SSD):** High-performance storage attached directly to databases for fast transaction processing.

3. Network Infrastructure:

- **Content Delivery Network (CDN):** A network of edge servers distributed globally to cache video content. This solves the **Global Latency** issue by serving video from a server closest to the user (e.g., serving Asian users from a Tokyo node).
- **Load Balancers:** To distribute incoming traffic evenly across the application servers.

4. Database Infrastructure:

- **NoSQL Database:** Best suited for the "Likes" and "Comments" because it can scale horizontally to handle millions of simple writes per second.
- **Relational Database (SQL):** Used for structured data like User Accounts and Billing information.

5. Cloud-Related Services:

- **Media Encoding Service:** A managed service to automate the transcoding of video files.
- **AI/ML Service:** Managed Machine Learning to power the recommendation engine.

1.2 Cloud Management Service

MultiCloud Enterprises has acquired three smaller companies, resulting in a messy environment where they use three different cloud providers (Cloud A, Cloud B, and Cloud C). The CIO is facing a crisis:

- **Shadow IT:** Developers are using personal credit cards to spin up servers without permission.
- **Waste:** An audit found 50 servers that have been running for months with 0% CPU usage (orphan resources).
- **Security:** There is no central way to see which servers have open ports to the internet.

The CIO wants to implement a **Cloud Management Platform (CMP)**. Explain how this platform would solve the specific problems of **Cost Visibility**, **Governance**, and **Provisioning**.

Model Answer:

Cloud Management Service Implementation:

1. Unified Visibility (Solving the "Three Clouds" problem):

- The CMP connects to the APIs of all three providers (Cloud A, B, and C).
- It provides a **"Single Pane of Glass" dashboard** that aggregates costs and resources, allowing the CIO to see total spend and security posture in one view, rather than logging into three separate consoles.

2. Cost Management & Optimisation (Solving "Waste"):

- **Resource Inventory:** The service scans for "Zombie" or "Orphan" resources (servers with <5% utilisation) and flags them for termination.

- **Budget Alerts:** Global budgets are set, alerting the finance team if any department exceeds their spend limit across any cloud.

3. Governance and Security (Solving "Shadow IT"):

- **Service Catalog:** Developers are blocked from direct console access. Instead, they must provision resources through a **Self-Service Portal** within the CMP.
- **Policy Enforcement (Guardrails):** The CMP enforces rules automatically (e.g., "No server can be created with an open public IP address"). If a user tries to violate this, the CMP blocks the request.
- **Tagging Strategy:** All resources are automatically tagged with "Cost Center" and "Owner" at creation, ensuring accountability.

1.3 Determine Resources to Work With in a Given Situation

GameDev Studios is launching a competitive multiplayer shooter game. The technical requirements are strict:

1. **The Match:** 100 players interact in a shared virtual world. If a player shoots, the server must calculate the physics in **less than 50ms**.
2. **The Lobby:** Players sit here while waiting for a match. Latency doesn't matter, but the system must handle thousands of idle connections cheaply.
3. **Matchmaking:** A background process that scans the player list to find players of similar skill levels. It runs sporadically (only when players join).
4. **Replays:** After a match, a video file is saved for players to watch later.

Determine the resources for these three distinct workloads. Specifically, justify why you would use **High-Performance Compute** for the Match but different resources for the Lobby and Replay storage.

Model Answer:

Resource Allocation Strategy:

1. The Match (High Performance Compute):

- **Resource: Dedicated Compute Instances** (High Frequency CPU).
- **Justification:** The 50ms latency requirement implies heavy physics calculations. You cannot use shared/burstable CPU because "noisy neighbors" might cause lag spikes. You need consistent, dedicated raw power.
- **Protocol:** UDP (User Datagram Protocol) for speed.

2. The Lobby (Cost-Effective Compute):

- **Resource: General Purpose / Burstable Instances** or **Containers**.
- **Justification:** Maintaining idle connections is low-CPU work. Using expensive high-performance servers here would be a waste of money.

3. Matchmaking (Event-Driven):

- **Resource: Serverless Functions** (Function-as-a-Service).
- **Justification:** This process runs sporadically. Serverless is ideal because you only pay for the milliseconds the code runs while sorting players, and it scales to zero when no one is queueing.

4. Replays (Storage):

- **Resource: Object Storage (Standard/Cool Tier).**
- **Justification:** Video files are unstructured data. Object storage is the cheapest and most durable place to store these large files.

1.4 Containerisation and Orchestration

FinTech Solutions runs a mobile banking app. It is currently a **Monolithic Application** running on large Virtual Machines. They face two major operational issues:

1. **"The Update Nightmare":** To update just the "Login" feature, they have to restart the entire banking system, causing downtime for logged-in users.
2. **"It Works on My Machine":** Developers write code on Windows laptops, but the servers run Linux. Code often breaks in production because libraries are missing or different.

They want to move to **Containers** and **Orchestration (e.g., Kubernetes)**. Explain how this move solves these two specific problems.

Model Answer:

Containerisation Strategy:

1. Solving "It Works on My Machine" (Encapsulation):

- **Containers:** A container packages the application code *together* with all its dependencies, libraries, and the operating system settings into a single immutable artifact.
- **Result:** The container runs exactly the same way on the developer's laptop as it does on the production server. This eliminates "environmental drift" errors.

2. Solving "The Update Nightmare" (Microservices & Orchestration):

- **Decomposition:** The Monolith is broken into Microservices (Login Service, Balance Service, Transfer Service), each in its own container.
- **Rolling Updates:** The Orchestration platform (e.g., Kubernetes) allows you to update *only* the "Login Container" while the "Balance Container" keeps running.
- **Zero Downtime:** The Orchestrator spins up the new version of the container, checks it is healthy, and only *then* directs traffic to it, allowing updates without taking the system offline.

2. Monitor and Maintain Specific Elements of the Cloud

2.1 Effective Server Maintenance and Optimisation

GlobalLogistics runs a fleet of 200 Virtual Machines (VMs) that manage their trucking routes. They are facing a crisis because they have neglected maintenance for two years.

- **The Security Issue:** The VMs are running an operating system version that reached "End of Life" six months ago. They have not been patched because the Operations Manager is terrified that "rebooting for updates might crash the application."
- **The Performance Issue:** Drivers are complaining that the route app is crashing. An analysis shows the servers are constantly running at **99% RAM usage**, even though the CPU is only at 10%.
- **The Process Issue:** Currently, the team manually logs into each of the 200 servers once a month to clear out temporary files. This takes 40 hours of staff time.

The CTO has tasked you with formulating a new strategy that handles **Patching**, **Right-Sizing**, and **Automation** without causing downtime.

Model Answer:

Maintenance and Optimisation Strategy:

1. Patch Management (Solving the Security Issue):

- **Strategy:** Implement **Automated Rolling Updates**.
- **Mechanism:** Instead of patching all 200 servers at once (which risks total downtime), the management tool patches servers in small batches (e.g., 10% at a time).
- **Safety Check:** If a batch fails (e.g., the app doesn't restart), the system automatically **Rolls Back** the changes, preventing the bad patch from spreading to the rest of the fleet.
- **Result:** The "End of Life" OS is upgraded securely without taking the trucking system offline.

2. Right-Sizing (Solving the Performance Issue):

- **Diagnosis:** The servers are "Memory Bound" (High RAM, Low CPU).
- **Action:** Change the Instance Type from "General Purpose" (balanced CPU/RAM) to **"Memory Optimised"** (High RAM, Low CPU).
- **Benefit:** This provides the necessary RAM to stop the crashes, and since "Memory Optimised" instances often cost less per GB of RAM than adding more General Purpose servers, it optimises spend.

3. Operational Automation (Solving the Process Issue):

- **Action:** Deploy **Configuration Management Agents** (or use a Scripting Service) to automate the disk cleanup.

- **Task:** Create a scheduled script: Every Sunday at 2 AM, delete /tmp files older than 7 days.
- **Benefit:** This eliminates the 40 hours of manual labor, freeing the team to focus on the OS upgrade.

2.2 Effective Storage Management Policies

MediaArchive Corp stores 2 Petabytes (PB) of news footage. They pay a flat rate for "Standard Storage" (R0.30 per GB), resulting in a massive monthly bill. An analysis of their access logs shows a clear lifecycle:

1. **Breaking News (Days 0-30):** Footage is accessed constantly by editors. Speed is critical.
2. **Recent History (Days 31-90):** Footage is used occasionally for documentaries.
3. **Deep Archive (Day 90+):** Footage is almost never watched (maybe once a year) but must be kept for legal reasons.

Design a **Storage Lifecycle Policy** that moves data between specific tiers (Hot, Cool, Cold/Archive) based on these three timeframes to reduce the monthly bill by 65%.

Model Answer:

Storage Management Policy:

1. Classification & Tiers:

- **Hot Tier (Standard):** For data aged 0-30 days. Highest cost, but lowest latency and no retrieval fees.
- **Cool Tier (Infrequent Access):** For data aged 31-90 days. Lower storage cost (~50% cheaper), but slightly higher retrieval cost.
- **Cold/Archive Tier:** For data aged 90+ days. Lowest storage cost (~80-90% cheaper), but retrieval takes hours.

2. Lifecycle Policy Automation:

- Create an automated rule on the storage bucket:
  - IF Age > 30 Days THEN Move to Cool Tier
  - IF Age > 90 Days THEN Move to Cold Tier

3. Expected Outcome:

- By moving 90% of their 2PB data to the Cold Tier (which is vastly cheaper), the company will reduce their monthly storage bill by approximately **65-70%**, aligning the cost with the actual value of the data.

2.3 Performance Characteristics of Storage

ShopFast Inc. is experiencing website slowdowns. The DevOps team suspects the storage attached to the database is the bottleneck. Users report two different symptoms:

1. **Symptom A:** "The daily backup job takes 6 hours instead of 1, slowing down the whole system." (This job reads massive sequential files).
2. **Symptom B:** "During the Black Friday sale, the checkout page froze." (This involves thousands of tiny, random database writes per second).

Describe a monitoring strategy. Which specific metric would you track to diagnose Symptom A (**Throughput/Bandwidth**) versus Symptom B (**IOPS**)?

Model Answer:

Performance Monitoring Strategy:

1. Throughput (Bandwidth) Monitoring:

- **Definition:** Measures the volume of data transferred (Megabytes per second).
- **Diagnosis for Symptom A:** The backup job is a large sequential read. The team must monitor **Throughput (MB/s)**. If the graph hits the maximum limit (e.g., 500 MB/s), the "pipe" is too narrow for the backup size.

2. IOPS (Input/Output Operations Per Second) Monitoring:

- **Definition:** Measures the *number* of individual read/write requests.
- **Diagnosis for Symptom B:** Checkout transactions are small but numerous. The team must monitor **IOPS**. If the database is trying to do 5,000 IOPS but the disk is capped at 3,000 IOPS, requests will queue up and the checkout will freeze.

3. Latency Monitoring:

- **Definition:** The time it takes for a single disk operation to complete.
- **Trigger:** High latency is the ultimate symptom of both bottlenecks above. If Disk Latency exceeds 20ms, it confirms the storage is overwhelmed.

2.4 Cost Implications Related to Storage and Database Management

StartupTech provides a SaaS platform. Their monthly bill is **R35,000**. The CFO wants to reduce this. The infrastructure audit reveals:

- **Storage Waste:** They host a 4K video background (200MB) on their homepage. 10,000 daily visitors download this file directly from the main server (High Egress Cost: R2.00/GB).
- **Database Waste:** They run a massive "Production-Size" database for their "Test" environment, which is only used by developers from 9 AM to 5 PM.

Calculate the current daily Data Transfer cost. Then, explain how implementing a **Content Delivery Network (CDN)**, **Auto-Scheduling** and **Caching policies** would drastically reduce this specific line items.

Model Answer:

Cost Optimisation Analysis:

1. Data Transfer Savings (CDN Implementation):

- **Current State:** 10,000 users * 0.2GB = 2,000GB/day. @ R2.00/GB = **R4,000/day**.
- **Solution:** Move the video to a **Content Delivery Network (CDN)**. CDNs cache content and have much lower data transfer rates (e.g., R0.50/GB).
- **Implication:** This simple change reduces the data transfer bill by **75%**.

2. Database Savings (Right-Sizing & Scheduling):

- **Current State:** Test Database runs 24/7 (730 hours/month).
- **Solution:** Implement **Auto-Scheduling** to turn the Test DB off at night (5 PM to 9 AM) and on weekends.
- **Calculation:** Running only 8 hours/day (Mon-Fri) = ~160 hours/month.
- **Implication:** This reduces the Test Database runtime by **~78%**, directly reducing that line item on the bill.

2.5 Monitoring of User Activity Through the Use of Logs

SecureBank has a security policy that forbids "Shared Root Accounts," yet a recent incident revealed a database was deleted at 3 AM using the "root" login. There is no way to know *which* of the 5 admins did it.

Design a logging framework to ensure **Non-Repudiation** (proof of who did what) for future incidents.

Model Answer:

User Activity Monitoring Framework:

1. Identity Management (The "Who"):

- **Federated Identity:** Disable the shared "root" account. Issue unique Named Accounts to every admin (e.g., admin.sarah, admin.john).
- **Benefit:** Every action is now tied to a specific human identity.

2. Audit Logging (The "What"):

- **Service:** Enable the cloud provider's **Audit Trail Service** (logs control-plane API calls).
- **Capture:** Ensure logs capture: Actor (admin.sarah), Action (DeleteInstance), Resource (DB-Prod), Source IP, and Timestamp.

3. Log Integrity (The "Proof"):

- **WORM Storage:** Ship these logs to a **Write-Once-Read-Many** storage bucket.

- **Benefit:** This prevents a malicious admin from deleting the logs to cover their tracks. This ensures the logs stand up as evidence.

3. Maintain Security Protocols

3.1 Compliance and Governance

GlobalHealthcare is expanding to Europe. They collect patient data. A former patient, "Mr. Smith," has formally exercised his **GDPR "Right to be Forgotten."**

- **The Conflict:** Mr. Smith's data is in the active database, but also on 500 backup tapes stored in a vault. You cannot edit a backup tape without destroying it.
- **The Question:** How do you comply with the deletion request without destroying your disaster recovery backups?

Discuss the governance and technical challenges here. How do you ensure you are compliant with his deletion request without destroying the integrity of your backup?

Model Answer:

Compliance Strategy:

1. Active Deletion:

- Immediately hard-delete the user's data from all active databases and search indexes.

2. The "Putative Deletion" Strategy for Backups:

- Since backup tapes are immutable (cannot be changed), you create a **"Suppression List"** or "Blocklist" containing the IDs of deleted users.
- **Protocol:** If a disaster occurs and you must restore from backup, the restoration script checks the Blocklist. It restores all data *except* the IDs found on the list.
- **Result:** The user's data remains on the tape (technically) but is never restored to a live environment, satisfying the "Right to be Forgotten" principle in most jurisdictions.

3.2 Data Life Cycle Management

InvestCorp is under investigation. Regulators demand all emails from 2020. The IT Manager worries that a rogue employee might have altered old emails to hide illegal activity.

Explain how using **WORM (Write Once, Read Many)** storage technology would have protected the company from this risk.

Model Answer:

Data Lifecycle & Integrity:

1. WORM Technology Defined:

- WORM storage applies a "Retention Lock" to data. Once a file is written, it becomes **Immutable**.

2. Protection Mechanism:

- If InvestCorp had stored the 2020 emails in WORM storage with a 7-year retention policy, it would be **technically impossible** to overwrite, modify, or delete those files before the year 2027.
- Even the "Root Admin" cannot alter a WORM-locked file.

3. Outcome:

- This guarantees **Data Integrity**. The company can prove to the regulators that the emails retrieved are the exact, unaltered originals, as modification was systemically impossible.

3.3 Security Vulnerabilities and Mitigation Strategies

SaaS-App Inc. had a security audit that found three critical flaws:

1. **Vulnerability A:** The login page allows users to type code into the username box to bypass passwords (SQL Injection).
2. **Vulnerability B:** Database backups are stored in a public storage bucket.
3. **Vulnerability C:** Developers left "Secret Access Keys" hardcoded in the application text files.

For each flaw, provide the specific **Mitigation Strategy** (e.g., Input Validation/WAF, Encryption, Secrets Management).

Model Answer:

Vulnerability Mitigation:

- **For Vulnerability A (SQL Injection):**
  - **Mitigation:** Implement **Input Validation** and use **Parameterised Queries** in the code (treating input as text, not executable code). Additionally, deploy a **Web Application Firewall (WAF)** to block injection patterns.
- **For Vulnerability B (Public Backups):**
  - **Mitigation:** Immediately apply **Access Control Policies** to block public access. Enable **Encryption at Rest** so that even if the file is stolen, it is unreadable without the key.
- **For Vulnerability C (Hardcoded Keys):**
  - **Mitigation:** Remove keys from the code. Implement a **Secrets Management Service** (a secure vault) so the application retrieves keys programmatically at runtime, rather than storing them in plain text.

3.4 Recovery Methods and Business Continuity

GlobalFactory produces 500 cars a day.

- **The Constraint:** Downtime costs R100,000 per minute. They require **Zero Downtime**.

- **Current State:** They have a Disaster Recovery site that is "Active-Passive" (it sits cold and takes 4 hours to turn on).
- **The Problem:** A 4-hour recovery time costs R24 Million.

Explain the architecture required to achieve the Zero Downtime goal.

Model Answer:

Business Continuity Architecture:

1. Move to Active-Active (Hot-Hot):

- Instead of one site sleeping (Passive), both Site A and Site B run simultaneously, each handling 50% of the traffic.

2. Data Synchronisation:

- Implement **Bi-Directional (Master-Master) Replication**. Data written to Site A is instantly replicated to Site B, and vice-versa.

3. Failover Logic:

- If Site A floods, the Global Load Balancer detects the failure instantly. It routes 100% of the traffic to Site B.
- **Result:** Since Site B is already running and has the data, the switchover happens in seconds/milliseconds, achieving the "Zero Downtime" goal and saving the R24 Million potential loss.

Rubric

Criteria	Proficient (3)	Developing (2)	Beginning (1)	Exemplary (4)
Technical Accuracy	Uses mostly correct terminology. Identifies general resource types correctly.	Some technical errors or misuse of cloud terms. General understanding only.	Significant technical inaccuracies or inability to identify correct services.	Uses precise terminology (e.g., IOPS, WORM, CDN, Multi-AZ). Identifies correct resource types for specific workloads.
Problem Analysis	Addresses most bottlenecks but may miss minor secondary requirements.	Identifies the problems but offers vague or generic solutions.	Fails to address the specific problems outlined in the scenario.	Addresses every bottleneck or "pain point" mentioned in the scenario with a specific technical solution.
Justification & Logic	Provides basic reasoning for choices, though some logic may be thin.	Lists solutions without explaining the underlying reasoning or trade-offs.	Offers solutions that do not logically align with the scenario's constraints.	Provides clear, logical reasoning for why a resource was chosen (e.g., why GPU for ML vs. CPU for transcoding).

Criteria	Proficient (3)	Developing (2)	Beginning (1)	Exemplary (4)
Cost & Performance Optimisation	Generally chooses appropriate tiers but may miss specific cost-saving opportunities.	Suggests "over-provisioned" or expensive solutions where cheaper ones would suffice.	Shows little to no consideration for cost or resource efficiency.	Consistently selects the most cost-effective and performant tier (e.g., Cold storage for archives, Spot/Burstable for idle loads).
Security & Compliance	Addresses major security concerns but may overlook specific mechanisms like Log Integrity or Secrets Management.	Identifies a need for security but lacks specific mitigation strategies (e.g., says "secure it" without saying how).	Ignores security implications or suggests non-compliant practices.	Integrates "Security by Design" (Encryption, WAF, WORM, Least Privilege) and accurately addresses regulatory needs like GDPR.

Section-Specific "Critical Success Factors"

To achieve a "Proficient" or "Exemplary" score, a response must include these specific elements from the model answers:

1. **Infrastructure & Orchestration**
   1.3 (Resource Strategy): Must match Spot Instances to "Training" and Serverless to "Preprocessing" to show cost-awareness.
   1.4 (Kubernetes): Must explain Self-Healing (restarting failed containers) and Packing Density (improving utilisation from 25% to 70%).

2. **Monitoring & Maintenance**
   2.1 (Maintenance): Must propose Right-Sizing (downsizing from "Extra Large") and Automated Patching.
   2.2 (Storage Policy): Must define specific triggers (e.g., 90 days for Warm, 2 years for Cold) to demonstrate lifecycle automation.
   2.3 (Storage Metrics): Must link Throttling to Request Limits and Latency to image load speeds.

3. **Security & Business Continuity**
   3.1 (Compliance): Must identify that storing EU data in Texas violates Data Sovereignty and requires regional data centers.
   3.2 (DLM): Must include Cryptographic Deletion or physical overwriting as part of the "Disposal" stage.
   3.4 (Recovery): Must quantify the loss (R85,000/min) to justify a near-zero RPO for production databases.

MICTSETA | Media, Information And Communication Technologies Sector Education And Training Authority

SHAPING SKILLS, PIONEERING INDUSTRIES, EMPOWERING FUTURES