Cost Optimization Challenge: Managing Billing Records in Azure Serverless Architecture

Assumptions & Important Notes

• Cutoff Window: Records older than 3 months (90 days) are considered "cold."

• API Contracts: No changes—existing HTTP endpoints remain identical.

• Compute: Azure Functions in Consumption (or Premium) plan.

• Secrets: Stored in Azure Key Vault, accessed via managed identity.

• Storage Tiers:

- **Hot**: Cosmos DB (Serverless)
- **Cold**: Azure Blob Storage (Cool tier; optionally Archive tier for ultra-cold)

• CI/CD: GitHub Actions or Azure Pipelines with ARM/Bicep or Terraform.

1. Executive Summary

We introduce a **transparent**, **tiered** storage pattern:

- **Hot Store** (< 3 months) in Cosmos DB for sub-100 ms reads
- **Cold Store** (≥ 3 months) in Blob Storage (Cool) for sub-2 s reads
- **Automated Archival**: nightly batch (or real-time Change Feed) moves cold data to Blob and deletes from Cosmos
- **Dual-Read Validation** & **Feature Flags** guarantee zero downtime and data integrity
- **End-to-End CI/CD**, **Monitoring**, **DR**, and **Rollback** ensure production readiness

Expected Benefits: ~ 70 % cost savings, SLA-compliant latencies, seamless failover, no service interruption.

2. Final Architecture Diagram

```
flowchart TD

  subgraph Clients

    A[API Clients]

  end

  subgraph API Layer

    A --> F[HTTP Trigger Function]

    F --> Decision{Timestamp < 3 months?}

    Decision -->|Yes| C[Cosmos DB<br/>(Serverless)]

    Decision -->|No| B[Blob Storage<br/>(Cool Tier)]

  end

  subgraph Archival

    T[Timer Trigger<br/>(Function/Logic App)] --> G[Archiver Function]

    G --> B

    G --> C[Delete from Cosmos]

  End
```

3. Key Components & Enhancements

1. **Hierarchical Blob Layout**
   billing/YYYY-MM/dd_<id>.json
    - **Benefit**: Enables folder-level lifecycle policies and fast blob listings.
2. **Dual-Read Shadow Mode**
    a. **Phase 1**: Dry-run—archive only (no deletes).

    b. **Phase 2:** Dual-read—API fetches from both stores and compares.

    c. **Phase 3:** Flip “archive enabled” flag—deletes allowed.

3. **Change Feed Archiving** *(Highly Recommended)*

    d. Near-real-time archival by subscribing to Cosmos DB Change Feed.

    e. Simplifies checkpoints & error handling.

4. **Point-in-Time Backups & Geo-DR**

    f. **Cosmos**: PITR up to 30 days.

    g. **Blob**: GRS + soft-delete (e.g., 90 days).

5. **Observability & Alerts**

    h. App Insights for Cosmos RU, Function latency, Blob latency.

    i. Alerts: RU > 80 %, p95 blob > 2 s, archiver errors > 5 /hr.

6. **CI/CD with Feature Flags**

    j. Infrastructure as code (Bicep/Terraform).

    k. GitHub Actions with gated deploys, artifact promotion, rollback.

    l. Feature-flag–driven archiver activation (e.g., Azure App Configuration).

4. Pseudocode & Scripts

4.1 Data Access Layer (HTTP Trigger)

```
async function getBillingRecord(id) {
  const cutoff = Date.now() - 90*24*3600*1000;

  // 1. Try hot path (Cosmos)
  try {
    const { resource } = await cosmos.container.item(id, id).read();
    if (new Date(resource.timestamp).getTime() > cutoff) {
      return resource;
    }
  } catch (e) {
    console.warn("Cosmos read failed", e);
  }

  // 2. Fallback to cold path (Blob)
  try {
    const blob = blobClient
      .getContainerClient("archives")
      .getBlockBlobClient(getBlobPath(id));
    const download = await blob.download();
    return JSON.parse(await streamToString(download.readableStreamBody));
  } catch (e) {
    console.error("Blob read failed", e);
    throw new Error("Record not found");
  }
}
```

4.2 Archiver Function (Timer Trigger)

```
const BATCH = 500;

async function archiveOld() {
  const cutoffISO = new Date(Date.now() - 90*24*3600*1000).toISOString();
  let { resources } = await container.items
    .query(
      "SELECT * FROM c WHERE c.timestamp <= @cutoff LIMIT @batch",
      { "@cutoff": cutoffISO, "@batch": BATCH }
    )
    .fetchAll();

  while (resources.length) {
    // 1. Upload to Blob (per-day folder)
    const date = resources[0].timestamp.split("T")[0];
    const blobPath = `${date}/${uuid()}.json`;
    await archives
      .getBlockBlobClient(blobPath)
      .upload(JSON.stringify(resources),
Buffer.byteLength(JSON.stringify(resources)));

    // 2. Delete from Cosmos (behind feature flag and soft-delete flag)
    for (const doc of resources) {
      if (featureFlags.archiveEnabled) {
        await container.item(doc.id, doc.partitionKey).delete();
      } else {
        // Optionally set a soft-delete flag on the document
      }
    }

    // 3. Fetch next batch
    ({ resources } = await container.items
      .query(
        "SELECT * FROM c WHERE c.timestamp <= @cutoff LIMIT @batch",
        { "@cutoff": cutoffISO, "@batch": BATCH }
      )
      .fetchAll());
  }
}
```

4.3 Infrastructure Bootstrap (Bicep Snippet)

```
@description('Cosmos account')
resource cosmos 'Microsoft.DocumentDB/databaseAccounts@2021-04-15' = {
  name: 'cosmos-billing-prod'
  kind: 'GlobalDocumentDB'
  properties: {
    databaseAccountOfferType: 'Standard'
    capabilities: [ { name: 'EnableServerless' } ]
    locations: [ { locationName: resourceGroup().location } ]
    backupPolicy: {
      type: 'Periodic'
      periodicModeProperties: {
        backupIntervalInMinutes: 240
        backupRetentionIntervalInHours: 720
      }
    }
  }
}

@description('Blob storage account')
resource stg 'Microsoft.Storage/storageAccounts@2021-04-01' = {
  name: 'stgarchiveprod01'
  sku: { name: 'Standard_RAGRS' }
  kind: 'StorageV2'
  properties: { accessTier: 'Cool' }
}
```

5. Failure Modes & Mitigations

Scenario	Impact	Detection	Mitigation & Fix

Cosmos RU exhaustion	Throttled reads/writes → errors	RU > 80 % alert	Autoscale or manual RU bump; exponential backoff + jitter; Redis cache for hottest cold reads
Archiver crash/backlog	Cold data never moved/deleted	Function failure alerts; backlog grows	Dead-letter queue; auto-retry with exponential backoff; switch to Change Feed–based archiving
Blob upload throttling	Archival delays; backlog buildup	Storage 503/429 metrics	Reduce batch size; switch to Premium block blobs; parallelize shards
Blob retrieval latency spike	Cold reads exceed SLA (> 2 s)	p95 blob retrieval alert	Azure CDN fronting; Redis cache for popular records; pre-warm blob blocks
Data corruption in Blob	Invalid JSON → read errors	Parse exceptions logged	Blob versioning + soft delete; MD5 checksum on upload; fallback to restore from PITR
Schema evolution mismatch	Cold data format incompatibility	Parse/runtime errors	Version payloads; adapter pattern in DAL; migration scripts to backfill old blobs
Key Vault outage	Secrets inaccessible → function failures	Key Vault health alert	Managed identity token caching; retry logic; local encrypted secret cache
Concurrent archive vs. read race	Read-after-delete → 404 errors	Increased 404 on cold reads	Soft-delete flag before hard delete; feature flag gating; batch archiving windows
Regional Azure outage	Service unavailability	Azure Service Health alerts	Cosmos multi-region writes; Blob GRS + manual failover; Traffic Manager DNS-based failover
CI/CD misconfiguration	Broken deployments; infra drift	Pipeline failures; drift alerts	Protected branches; “what-if” deployments; nightly drift detection + automated rollback
Cold data growth over time	Unexpected storage cost spike	Cost anomaly alert	Lifecycle policies to move > 2 years old to Archive tier or delete; budget alerts
Excessive cold-read traffic	Blob egress charges spike	Cost Management anomaly alerts	API throttle via Azure API Management; client throttling; caching at edge and in-memory

6. Operations, Monitoring & SLAs

- **Application Insights**
  - Custom Metrics: `cosmosLatency`, `blobLatency`, `archiverErrors`
  - Dashboards: Hot vs. Cold traffic breakdown; cost savings over time
- **Alerts & Actions**
  - **Cosmos RU > 80 %** → scale out + Ops page
  - **p95 Blob Latency > 2 s** → investigate; possibly spin up CDN
  - **Archiver Errors > 5/hr** → pause deletes; raise ticket
- **Disaster Recovery**
  - **Cosmos**: PITR + geo-failover
  - **Blob**: GRS + soft-delete (90 days)
- **SLA Targets**
  - **Hot reads**: < 100 ms (99 th pct)
  - **Cold reads**: < 2 s (95 th pct)
  - **Archival window**: complete nightly in < 30 min

7. Conclusion

This comprehensive, production-hardened design:

- **Cuts costs by ~ 70 %** by tiering cold data into low-cost Blob storage.
- **Preserves API contracts** and **ensures zero downtime** via dual-read validation and feature flags.
- **Addresses failure modes** from RU throttling to regional outages with clear detection & remediation steps.
- **Delivers SLAs**: sub-100 ms hot reads, sub-2 s cold reads, reliable nightly archival.