ViewLegacy.

レガシーデータベース連携において業務データをクラウドに保存しないことによるリスク低減効果

技術白書

発行: 合同会社ビューレガシー
著者: 増田朝陽
発行日: 2026年07月06日

本書は、業務データを必要時にプロバイダークラウド経由で中継しつつ、その業務データをクラウド側に永続保存しないレガシーデータベース連携方式が、どのようなリスクを低減し、どのようなリスクを残すのかを評価する。

1. エグゼクティブサマリー (1/2)

レガシーデータベース連携は、多くの場合、実務上の単純な要求から始まる。AIエージェント、SaaS、顧客ポータル、取引先システム、社内ツールに対して、選択された業務データへアクセスさせたいという要求である。設計上の問いは、そのアクセスのために、機密性の高い業務レコードの永続的なクラウド側コピーを作る必要があるかどうかである。

本書は、6つの方式を比較する。クラウドDBへの全量複製、データウェアハウスやデータレイクへのETL/ELT、iPaaSまたはSaaS同期、業務レコードをキャッシュするカスタムクラウドAPI、非永続クラウドパススルーゲートウェイ、オンプレミスエージェントを伴うケイパビリティベースの非永続パススルー方式である。

結論はバランスを取ったものである。プロバイダークラウドに業務データを保存しないことは、クラウド側侵害、バックアップ漏えい、ログ漏えい、保持設定ミス、下流分析への拡散によって露出する蓄積済み機密データの量を減らしうる。また、保持、削除、データ所在地、インシデント範囲特定の一部を単純化しうる。

1. エグゼクティブサマリー (2/2)

ただし、リスクは消えない。業務データは依然としてゲートウェイを通過する。オンプレミスエージェントはソースDBへアクセスする。運用メタデータも機微になりうる。この方式は、法律、契約、プライバシー、セキュリティ上の義務を自動的に解決するものではない。

非永続クラウドパススルー方式は、外部システムが現在の特定レコードや明示的な業務ケイパビリティを必要とし、DB全体の二次コピーを作る必要がない、読み取り中心の運用アクセスに特に適する。

一方で、分析、大規模BI、機械学習、履歴レポーティング、オフライン利用、クラウド側の永続データセットを必要とする用途では、データ複製、データウェアハウス、統制された分析基盤、SaaS同期の方が適切な場合がある。

最も強い設計は、単に「データを保存しない」ことではない。業務データをプロバイダークラウドに保存せず、payloadをログに残さず、生SQLを露出せず、スコープされた明示的ケイパビリティだけを公開することである。

2. 背景 (1/2)

レガシーシステムは、顧客、注文、在庫、会計、契約、従業員データの真のソースであり続けることが多い。多くは、現代的なAPIゲートウェイ、SaaS連携、AIエージェント、MCPクライアントを前提に設計されていない。

現代のアプリケーションは、このデータへのアクセスをますます必要としている。よくある解決策は「コピーする」ことである。DBを複製する、ETLでDWHへ送る、SaaSへ同期する、クラウドAPIバックエンドでキャッシュする、開発や運用の過程で必要以上のログやトレースを取得する。

業務データが別システムへコピーされると、追加の義務が生じる。アクセス制御、暗号化、鍵管理、オペレーターアクセスレビュー、バックアップ保持、削除対応、データ主体要求、インシデント対応、監査範囲、ベンダーレビュー、データ所在地レビュー、分析・ダッシュボード・サポートツール・エクスポート先の共有レビューである。

ログ、トレース、エラーメッセージ、キュー、オブジェクトストレージ、一時ファイル、ダッシュボードは、意図せず隠れたデータストアになりうる。

2. 背景 (2/2)

「クラウドに保存しない」という主張は、境界が明示されて初めて意味を持つ。本書では次の区分を用いる。

データ区分	例	評価対象のパススルー方式での扱い
業務データ	顧客、注文、請求、在庫、契約、従業員データ	リクエスト/レスポンス経路を通過しうるが、プロバイダークラウドに永続保存しない
運用メタデータ	timestamp、capability名、APIキー識別子、status、latency、error code、接続状態	ルーティング、認可、可観測性、監査のため保存しうる
ログ	gateway request log、control event、dashboard view	payload、生SQL、認証情報、機微なDBエラー、response bodyを除外する
トレース	distributed tracing span、request attribute、sampled payload	request/response payloadを取得しない
バックアップ	DB snapshot、object storage backup、log archive	非永続を主張するなら業務payload storeを含めない
一時ファイル	retry buffer、crash dump、export、debug file	provider cloudに業務payloadを含めない
ダッシュボード	provider operations UI、support tool、customer metrics	業務レコードではなくメタデータを表示する

3. 比較対象の連携パターン (1/10)

パターン1: クラウドDBへの全量複製

業務データをレガシーDBからクラウドDBへコピーする方式である。クラウド側には、広いテーブル、インデックス、バックアップ、レプリカ、管理ツールを含む永続的な二次コピーが作られる。

主な利点は、クラウド側で柔軟にアクセスできることである。AIエージェント、SaaS、BI、社内アプリは、毎回ソースシステムに触れずに管理されたデータセットを読める。

主なリスクは、蓄積された露出である。クラウドDB、バックアップ、ログ、エクスポート、サポートツールが侵害されると、広い二次データセットがインシデント範囲に入る。削除と保持は、ソース、クラウドコピー、インデックス、レプリカ、バックアップ、下流利用者を含めて扱う必要がある。

3. 比較対象の連携パターン (2/10)

パターン2: データウェアハウスまたはデータレイクへのETL/ELT

業務データを抽出、変換し、分析基盤へロードする方式である。通常、履歴データ、派生テーブル、集計、エクスポート、BIセマンティックモデルが保存される。

主な利点は分析である。DWHやデータレイクは、レポーティング、履歴トレンド分析、広いjoin、実験、統制されたBIに適する。

主なリスクは、分析用途での拡散である。データはstaging table、変換job、object storage、notebook、feature store、dashboard、query result cache、exported spreadsheetへコピーされうる。pipeline logやerror recordが行の値を保持することもある。

分析が主目的なら、この方式は適切になりうる。一方、現在の1件の運用レコードを狭く参照するだけなら、多くの場合過剰である。

3. 比較対象の連携パターン (3/10)

パターン3: iPaaSまたはSaaS同期

業務データをサードパーティの連携プラットフォームやSaaSへ同期する方式である。ベンダーは、レコード、connector state、mapping、retry payload、log、support-visible diagnostic dataを保存しうる。

主な利点は導入速度である。既製connectorやworkflow automationを使い、短期間でシステムを接続できる。

主なリスクは、ベンダーと下流の範囲である。保存場所、保持、バックアップ、sub-processor、support access、削除、ログ挙動は、プロバイダーのプラットフォームに依存する。顧客は、何が保存され、どこに保存され、誰がアクセスでき、どのように削除されるかを理解する必要がある。

SaaSが自サービス提供のために永続コピーを正当に必要とする場合、この方式は適切である。ただし、ベンダー管理だから低リスクだと仮定すべきではない。

3. 比較対象の連携パターン (4/10)

パターン4: 業務レコードをキャッシュまたは保存するカスタムクラウドAPI

カスタムクラウドAPIが外部クライアントに整理されたインターフェースを提供し、その内部で業務レコードをDB、cache、object store、queue、search indexなどに保存する方式である。

主な利点はプロダクト制御である。API設計、認証、性能調整、レガシーシステムの隠蔽を自社で制御できる。

主なリスクは、「cache」が実務上データストアになることである。cached recordは、backup、debug tool、dashboard、trace、replay queue、support workflowへ入りうる。保持と削除の方針は、意図した保存と偶発的な保存の両方を考慮する必要がある。

latency、offline動作、大量の反復アクセスのためにクラウド側状態が必要な場合には有用である。ただし、payload保存が技術的に防止され検証されていない限り、非永続とは呼ぶべきではない。

3. 比較対象の連携パターン (5/10)

パターン5: 非永続クラウドパススルーAPIゲートウェイ

クラウドゲートウェイがcaller認証、認可、routing、運用メタデータ記録を行う。業務データはrequest/response経路を通過するが、provider cloud database、object storage、queue、analytics store、dashboard、trace、logには保存しない。

主な利点は、蓄積されたクラウド側payload露出の低減である。プロバイダークラウドに永続的な業務payload storeがなければ、侵害、バックアップ、保持、削除、データ所在地に関する問いの一部は狭くなる。

主なリスクはrequest pathの機微性である。侵害されたgatewayは、in-flight dataを観察しうる。log、trace、queue、crash dump、一時ファイル、support toolがpayloadを取得すれば、非保存という主張は崩れる。

この方式は、クライアントが現在のデータを必要とするが分析コピーを必要としない運用読み取りアクセスに適する。

3. 比較対象の連携パターン (6/10)

パターン6: オンプレミスエージェントを伴うケイパビリティベース非永続パススルー

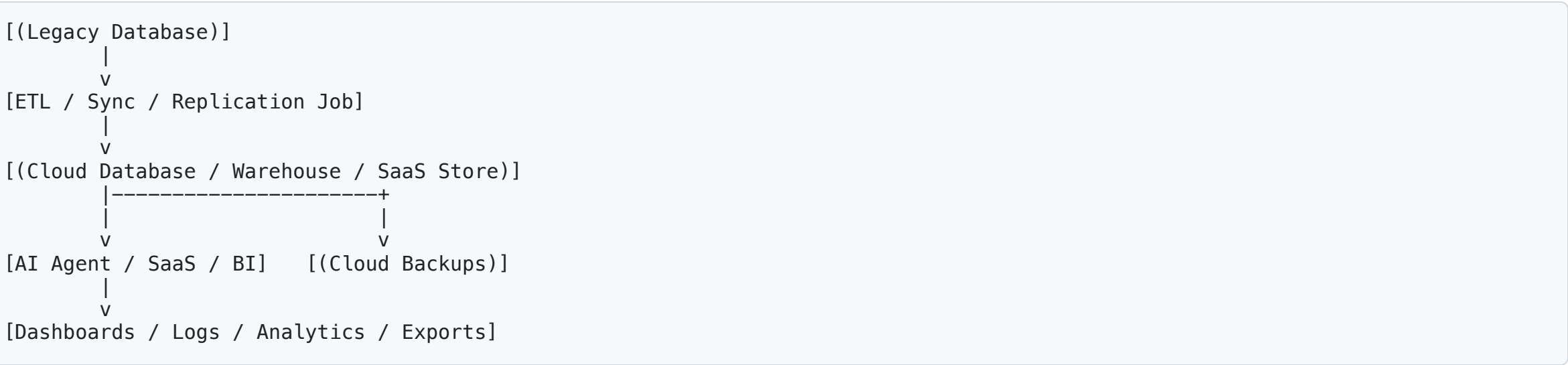
この方式は、agent-defined capability boundaryを追加する。プロバイダークラウドはREST APIとMCPを通じて選択された読み取り中心の操作を公開するが、生SQLやDB認証情報を知らない。APIキーは許可capabilityにスコープされる。OpenAPIとMCP tool definitionはcallerの認可scopeに基づいてフィルタされる。

オンプレミスエージェントは、SQL、DB認証情報、validation、execution policy、output filtering、row limit、business meaningを担う。`capability.yaml` が存在しうる操作を定義する。

主な利点は、discoverable/callable surfaceを狭めることである。クライアントが見るのは `get_invoice_status` のような業務capabilityであり、生テーブル、認証情報、広いSQL toolではない。

主なリスクは、capabilityとagent layerへの集中である。定義が広すぎる、output filterが弱い、agent hostが侵害される場合、機密性リスクは残る。

3. 比較対象の連携パターン (7/10)



このモデルでは、連携経路が意図的に永続的な二次データセットを作る。分析、性能、可用性には価値がある一方で、クラウド側の保存、バックアップ、ダッシュボード、サポート、インシデント対応の範囲を拡大する。

3. 比較対象の連携パターン (8/10)

```
[AI Agent / SaaS / MCP Client]
        |
        | HTTPS request: capability + params
        v
[Cloud Gateway: auth, routing, observability]
        |
        | outbound tunnel message
        v
[On-Prem Agent: validation, SQL, output filtering]
        |
        v
[(Legacy Database)]
        |
        v
[On-Prem Agent] -> [Cloud Gateway] -> [Client]

[Cloud Gateway] -. stores metadata only .-> [(Operational Metadata)]
```

業務データは依然としてgatewayを通過する。リスク低減は、provider cloud storage、log、trace、queue、backup、analytics store、dashboardに業務レコードを蓄積しないことから生じる。

3. 比較対象の連携パターン (9/10)

```
Provider Cloud
  [Gateway]
    |-- routing
    |-- identity
    |-- authorization
    |-- payload-free observability
    +--> [(Metadata Store)]

Customer Environment
  [On-Prem Agent]
    |-- capability.yaml
    |-- database credentials
    |-- prepared SQL
    |-- validation and output allow-list
    +--> [(Legacy Database)]

Boundary rule:
  Business payloads may transit the gateway.
  Business payloads must not be persisted in provider cloud systems.
```

この境界は実装で検証されなければならない。可観測性、support tool、retry、backupがpayloadを保持するなら、「保存しない」と言うだけでは足りない。

3. 比較対象の連携パターン (10/10)

パターン	クラウド側の永続業務データ	典型的な保存データ	主な利点	主なリスク	適した用途
全量DB複製	あり	table、index、backup、replica	柔軟なクラウドアクセス	広い二次データセット露出	クラウド側の永続運用コピー
ETL / ELT to DWH/lake	あり	履歴、派生table、export	分析とBI	分析用途での拡散と保持複雑性	統制されたBI、ML、広いjoin
iPaaS / SaaS同期	多くはあり	synced record、mapping、retry	迅速な連携	vendor storage、support、下流範囲	SaaS側recordが必要なworkflow
Cloud API cache	しばしばあり	cache、queue、debug data	性能と制御	cacheが別のdata storeになる	低latencyの反復アクセス
非永続pass-through	設計上なし	metadata、auth config、routing state	蓄積payloadの低減	transit、log、trace、queue leak	現在レコードの運用lookup
Capability-based pass-through	設計上なし	metadata、capability config、filtered catalog	discoveryと実行面を狭める	agent/capability統制が重要	AI、MCP、SaaS、社内toolの読み取り連携

4. 評価基準 (1/2)

評価軸	全量複製	ETL / DWH	iPaaS / SaaS sync	Cloud API cache	非永続pass-through	Capability-based pass-through
provider cloudでの業務データ永続化	高	高	高	中から高	設計上なし	設計上なし
蓄積機密データ量	高	履歴込みで高	中から高	cache次第	統制が効けば低	統制が効けば低
cloud側侵害blast radius	高	高	中から高	cache範囲	in-flight + metadata	in-flight + metadata + capability scope
backup/retention負荷	高	高	vendor依存	中	低め	低め
削除/DSR複雑性	高	高	高	中	低め	低め
log/trace漏えいリスク	payload logで高	pipelineで高	connectorで高	高	統制失敗で高	統制失敗で高
運用metadataの機微性	中	中	中	中	中	中
data residency/vendor review	高	高	高	中	storage reviewは狭い	storage reviewは狭い

4. 評価基準 (2/2)

評価軸	全量複製	ETL / DWH	iPaaS / SaaS sync	Cloud API cache	非永続pass-through	Capability-based pass-through
audit scope	広いcloud dataset	広いanalysis platform	vendorと下流tool	API + cache	gateway metadataとcontrol	gateway、metadata、capability、agent
recovery/incident scoping	複雑	複雑	vendor依存	中	payload storeがなければ狭い	capability logがcleanなら狭い
operational read access	良い	staleの可能性	sync済みなら良い	良い	良い	強い
MCP / AI agent access	広いと危険	間接的	依存	scope次第	scope次第	強い
analytics / BI	弱から中	強い	中	限定的	弱い	専用設計なしでは弱い
operational complexity	中	高	中	中	中	中から高
performance / latency	sync後は低	分析では低	sync後は低	warm cacheなら低	高め、source依存	高め、source依存

5. 評価方法 (1/2)

この評価は、実務的な例示モデルであり、形式的な定量リスクモデル、法律意見、コンプライアンス認証、セキュリティ監査の代替ではない。

モデルは、顧客、注文、請求、在庫、契約、従業員データを含む架空のレガシー業務システムを想定する。データは5種類に分類する。

- public metadata
- internal operational metadata
- confidential business records
- personal data
- sensitive personal or regulated data

各連携パターンについて、何が保存され、送信され、ログに入り、ダッシュボードに露出し、バックアップやretry経路に含まれるかを確認する。そのうえで、12個のシナリオを high risk、medium risk、lower risk、reduced if correctly implemented、depends on controls、out of scope といった定性的ラベルで評価する。

5. 評価方法 (2/2)

データ区分	Replication / ETL	iPaaS / SaaS Sync	Cloud API with cache	Non-persistent pass-through	Capability-based pass-through
raw business records	広く保存	vendor systemに保存	cacheされれば保存	設計上保存しない	設計上保存しない
SQL query results	保存またはquery cache	recordとして保存されうる	しばしばcache	transitのみ	output filtering後にtransit
request payloads	log/traceへ入りうる	connector logへ入りうる	API logへ入りうる	保存してはならない	保存してはならない
response bodies	log/dashboardへ入りうる	SaaS support toolへ入りうる	cache/logされうる	保存してはならない	保存してはならない
raw SQL	job/logにあることが多い	connectorに隠れることが多い	backendに存在しうる	gatewayは知らない	agent-sideのみ
DB credentials	connector/cloud側の可能性	connector/vendor管理	cloud backendに存在しうる	agent-sideが望ましい	agent-sideのみ
operational metadata	保存	保存	保存	保存	capability scope付きで保存
dashboard data	record表示の可能性	record表示の可能性	cache record表示の可能性	metadataのみ	metadataのみ
backups	business recordを含む	synced recordを含む	cacheがあれば含む	metadata/configのみ	metadata/configのみ

6. 評価結果 (1/4)

シナリオ	Full Replication	ETL / DWH	iPaaS / SaaS Sync	Cloud API Cache	Non-Persistent Pass-Through	Capability-Based Pass-Through
S1 provider cloud DB侵害	高: 二次dataset露出	高: 履歴dataset露出	高: synced record露出	中/高: cache範囲	payload storeなしなら低減	durable data riskは低い
S2 object storage backup設定ミス	高	高	中/高	中	payload backupなしなら低い	metadata/configのみなら低い
S3 logがresponse bodyを含む	高	高	高	高	高: 主張が崩れる	技術的blockなしなら高
S4 traceがpayloadを取得	高	中/高	中/高	高	高	redaction設計なしなら高
S5 queue/retryがpayload保持	中	高	高	中	高: storage化	retryがpayload除外なら低い
S6 support dashboardがrecord表示	高	高	中/高	高	metadata-onlyなら低い	metadata-only/read-onlyなら低い

6. 評価結果 (2/4)

シナリオ	Full Replication	ETL / DWH	iPaaS / SaaS Sync	Cloud API Cache	Non-Persistent Pass-Through	Capability-Based Pass-Through
S7 削除またはretention enforcement	複雑性高	複雑性高	複雑性高	中	payload storeなしなら低減	低減。ただしsourceとmetadataは残る
S8 data residency storage review	範囲広い	範囲広い	vendor範囲広い	中	storage reviewは狭い	metadataとagentもreview対象
S9 on-prem agent host侵害	source risk残存	source connector risk	connector risk	connector risk	高: source access	高。capability limitのみで低減
S10 gatewayがtransit中に侵害	高	高	高	中/高	中: in-flight data	中: scopeされるが消えない
S11 正当AIが現在record 1件を要求	動くが広い	freshなら動く	sync済みなら動く	cache懸念あり	適合	強く適合
S12 BIが履歴分析datasetを必要	統制下なら良い	強く適合	適合可能	限定的	不適	analytics設計なしでは不適

6. 評価結果 (3/4)

リスク	storage-based integration	non-persistent pass-through	capability-based pass-through
蓄積機密dataset	replica、cache、export、backupで増える	provider-cloud payload storeがなければ低減	narrow outputでpayload非保存なら低減
cloud側breach blast radius	durable business recordを含みやすい	metadata + in-flightに限定されやすい	metadata、tool catalog、capability config、in-flightに限定されやすい
backup leakage	backupがbusiness recordを含む	metadata/configのみなら低い	capability configにpayload exampleやrecordを含めない
retention/deletion complexity	copy、derived table、index、log、export対象	source systemとmetadata policyは残る	source、metadata、capability history、audit logは残る
log/trace leakage	多数systemでpayload logの可能性	payload logなら非保存主張が崩れる	tool-call argumentもreview対象
operator dashboard exposure	dashboardがrecord表示しうる	metadataのみ表示すべき	metadata、capability名、status、latencyのみ表示すべき
endpoint/source compromise	sourceに加えてcloud copyも存在	agentとsourceが重要	agent、`capability.yaml`、output filter、credentialが重要
analytics suitability	分析には強い	弱い	別の統制済み分析経路なしでは弱い

6. 評価結果 (4/4)

評価から分かることは、業務データをプロバイダークラウドに保存しないことが、主に「蓄積データリスク」を下げるという点である。永続的なクラウド側露出を減らし、特定条件下で侵害調査範囲を狭め、保持、削除、バックアップ、データ所在地に関する一部の問いを単純化しうる。

しかし、リスクは消えない。移動する。

storage-based integrationでは、リスクの多くは永続cloud dataset、backup、transformation、downstream consumerに集中する。non-persistent pass-throughでは、重要な統制はrequest-path security、agent security、payload-free logging、trace redaction、retry design、dashboard design、support workflow、authorization scope、source-system protectionへ移る。

operational metadataも機微である。capability名、caller identity、timing、frequency、status code、error code、latencyは、business recordを保存しなくても業務活動を推測させうる。

7. 考察 (1/3)

業務データをプロバイダークラウドに保存しないことは、durable secondary-copy riskという1種類のリスクを下げる。実装が本当にpayloadを保存しないなら、「provider cloudにどのbusiness recordが保存されているか」という問いに答えやすくなる。

一方で、この方式はrequest-path controlの重要性を高める。gatewayはcallerを認証し、capabilityを認可し、過剰アクセスを拒否し、payload永続化を避け、telemetryを保護し、エラー処理でデータを漏らしてはならない。agentは入力を検証し、policyを適用し、prepared SQLを使い、row/byte limitを適用し、出力をfilterし、DB credentialを保護する必要がある。

log、trace、queue、error report、dashboard、temporary file、support tool、backupがpayloadを取得するなら、「非保存」という主張は実務上 false になる。

7. 考察 (2/3)

non-persistent pass-throughは、分析担当者が柔軟なad hoc exploration、多数tableにまたがる広いanalytical join、大量履歴dataset、ML training data、offline availability、低latencyの反復大量アクセスを必要とする場合には最適ではない。

durable datasetと広い分析が実際の要件なら、data warehouse、data lake、統制されたBI semantic layerの方が適切な場合がある。その場合の正しい対策は、保存を避けられるふりをすることではない。classification、access control、encryption、retention、deletion、lineage、logging、monitoring、incident responseを含む強いgovernanceである。

同様に、SaaSがサービス提供のために自らの永続record copyを正当に必要とする場合がある。重要なのは、保存が必要かつ統制されているかであって、すべての保存が悪いということではない。

7. 考察 (3/3)

「クラウドに業務データを保存しない」設計は、次の場合に失敗する。

- logがrequest/response payloadを取得する
- error reportがrecord valueやraw DB errorを含む
- distributed traceがrequest body、response body、header、sensitive attributeを取得する
- queue、retry buffer、dead-letter storeがbusiness dataを永続化する
- dashboardがprovider operatorにbusiness recordを表示する
- temporary file、crash dump、debug exportがcloud diskへ書かれる
- backupがpayload storeを含む
- support toolがoperatorへpayloadを露出する
- operational metadataを非機微として扱い、業務活動を推測させる
- on-premise agent hostが侵害される
- gatewayがrequest processing中に侵害される
- capability definitionが広すぎ、過剰なデータを返す

この方式は、これらの失敗モードに対して検証される必要がある。

8. 実務上の示唆 (1/2)

AIエージェント、SaaS、社内システムをレガシーDBへ接続する企業は、まずユースケースがoperational lookupなのかanalytical replicationなのかを決めるべきである。

operational lookupでは、可能な限りread-only capabilityから始める。明確な分析上または運用上の理由がない限り、full table copyを避ける。業務タスクに必要なfieldだけを返すcapabilityを定義する。

business dataをlog、trace、dashboard、metric、queue、error report、temporary file、support toolから除外する。provider cloudには最小限のoperational metadataだけを保存し、そのmetadataを潜在的に機微なものとして分類する。

capability-scoped authorizationを用いる。API keyは許可capabilityへscopeする。OpenAPIとMCP tool definitionはcallerのauthorization scopeでfilterし、discoveryが不要なoperationを露出しないようにする。

8. 実務上の示唆 (2/2)

security/architecture reviewでは、次を明示的に文書化すべきである。

- どのbusiness dataが保存されるか
- どのbusiness datがtransitするか
- どのbusiness datがnever persistedであるか
- どのoperational metadataが保存されるか
- logがpayloadを含むか
- traceがpayloadを含むか
- backupがpayload storeを含むか
- dashboardがrecordを表示するかmetadataのみか
- temporary file、retry、dead-letter queueがbusiness dataを含みうるか
- on-premise agentがどこで動き、credentialをどう保護するか
- capability definitionがinput、SQL、row、byte、outputをどう制約するか
- gateway compromise、metadata compromise、agent/source compromiseでincident responseがどう異なるか

logging/tracing設定をテストし、cloud backupを確認し、support dashboardを点検する。payload-free designは、方針だけでなくcodeとconfigurationで強制されている必要がある。

9. 制限事項と今後の課題 (1/2)

この評価は例示であり、完全なセキュリティ監査、法律レビュー、プライバシーレビュー、コンプライアンス認証、顧客固有の脅威モデルの代替ではない。

実際の義務は、法域、契約、データ区分、顧客ポリシー、provider terms、sub-processor、実装詳細、運用成熟度に依存する。

この比較は、latency、throughput、availability、costを深くbenchmarkしていない。また、payloadがlog、trace、queue、backup、dashboard、monitoring system、support tool、temporary file、crash artifactに保存されないよう厳格な統制があることを前提としている。

評価は主にread-oriented operational accessに焦点を当てる。write operationには、approval flow、idempotency、transaction design、conflict handling、change log、rollback procedure、より強いaudit、より明確な責任境界が必要である。

9. 制限事項と今後の課題 (2/2)

今後の課題には、次が含まれる。

- request/response bodyが取得されないことを証明する実際のlogging test
- payload redactionを確認するdistributed tracing test
- backupがbusiness payload storeを含まないことを確認するcloud backup verification
- metadata、routing path、customer environmentを含むdata residency analysis
- personal dataとoperational metadataに対するprivacy-impact analysis
- gateway、agent、support、observability workflowに対するred-team exercise
- replication-based approachとのbenchmark comparison
- write-capability control、approval workflow、audit requirementの評価
- API-key scopeごとのMCP/OpenAPI discovery filteringの検証

「クラウドに業務データを保存しない」という主張は、payload logging、payload tracing、raw SQL exposure、excessive capability output、provider support visibilityを同時に防ぐcontrol setとして評価されるべきである。

10. 参考文献 (1/2)

- OWASP Cheat Sheet Series, "Logging Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
- OWASP Top 10 Proactive Controls, "C9: Implement Security Logging and Monitoring." https://top10proactive.owasp.org/the-top-10/c9-security-logging-and-monitoring/
- OWASP API Security Top 10 2023, "API1:2023 Broken Object Level Authorization." https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/
- OWASP API Security Top 10 2023, "API3:2023 Broken Object Property Level Authorization." https://owasp.org/API-Security/editions/2023/en/0xa3-broken-object-property-level-authorization/
- NIST, "SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations." https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final
- NIST, "SP 800-92, Guide to Computer Security Log Management." https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf
- Model Context Protocol, "Tools." https://modelcontextprotocol.io/specification/2025-11-25/server/tools
- OpenAPI Initiative, "OpenAPI Specification v3.2.0." https://spec.openapis.org/oas/v3.2.0.html

10. 参考文献 (2/2)

- OpenAPI Documentation, "Describing API Security." https://learn.openapis.org/specification/security.html
- Amazon Web Services, "Shared Responsibility Model." https://aws.amazon.com/compliance/shared-responsibility-model/
- Microsoft Learn, "Shared responsibility in the cloud." https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility
- Regulation (EU) 2016/679, "Article 5: Principles relating to processing of personal data." https://gdpr-info.eu/art-5-gdpr/
- UK Information Commissioner's Office, "Data minimisation." https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/data-minimisation/
- UK Information Commissioner's Office, "Storage limitation." https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/storage-limitation/

補足: 評価アーティファクトの配布ファイル

本書で使った例示評価の補助ファイルは、PDFとあわせて次のzipファイルとして配布する。

```
src.zip
```

zipには、評価モデル、小さな生成スクリプト、生成済みのシナリオ結果表が含まれる。

```
src/evaluation-model.json
src/evaluate-scenarios.mjs
src/scenario-outcomes.md
```

これらのファイルは、本書の要約表の背後にあるscenario noteを記録する。定性的な比較補助であり、形式的な確率モデルではない。

最後に

本書は、非永続クラウドパススルー方式が常に優れていると主張するものではない。読み取り中心のレガシーシステム運用アクセスにおいて、業務データをクラウド側に永続保存しないことが、特定の重要なリスク群を下げうることを示すものである。

この設計は、payload-free logging、payload-free tracing、scoped capability、metadata discipline、agent-side enforcement、source-system securityを一体のcontrol setとして扱う場合に最も強くなる。

発行: 合同会社ビューレガシー
著者: 増田朝陽
発行日: 2026年07月06日