Project A: AWS Cloud Environment (AWS-like Cost Optimization Sandbox)

This document defines the first project: a AWS cloud environment that simulates an AWS-like infrastructure for cost optimization. The goal is to model the key AWS services that typically require manual FinOps work, store them in Supabase/Postgres tables, and drive continuous 'waste' via drift functions. A separate Agentic AI platform will later connect to this environment to discover resources, detect waste, and simulate optimization actions.

1. Design Goals

• Mirror the core AWS concepts relevant to cost optimization (compute, storage, databases, networking, observability, commitments, governance).

• Use Supabase/Postgres tables to represent AWS-like resources with realistic fields (IDs, types, env, state, metrics, cost).

• Provide a simple surface (APIs or direct queries) for an external Agentic AI to read state and apply actions.

• Support a Drift Engine that continuously introduces new resources, idle workloads, zombies, and growth, so the environment is never static.

2. Scope of Simulated AWS Services

• Compute: EC2-style instances, autoscaling groups, container/Kubernetes nodes, Lambda-style serverless functions.

• Storage: EBS-style block volumes, snapshots, S3-style buckets and tiered storage usage.

• Databases & Caches: RDS-style database instances, Redis/ElastiCache-style cache clusters.

• Managed Services: Generic managed search/streaming/feature-store clusters for consolidation scenarios.

• Streaming & Messaging: Kinesis/Kafka-style streams and managed messaging throughput.

• Networking & Traffic: Load balancers, Elastic IPs, data transfer metrics (cross-AZ/region, internet egress).

• Observability: Log groups and log volume metrics (CloudWatch-style).

• Commit Discounts: Reserved Instances / Savings Plans commitments and their utilization.

• Governance: Resource tags and ownership metadata to simulate FinOps accountability.

3. Core Schema Overview

The schema is organized by domain. All tables live in a single Supabase/Postgres project (the AWS cloud account).

Compute:

- • cloud_accounts
- • instances
- • autoscaling_groups
- • container_clusters
- • container_nodes
- • lambda_functions

Storage:

- • volumes
- • snapshots
- • s3_buckets
- • s3_bucket_usage_daily

Databases & Caches:

- • rds_instances
- • cache_clusters

Networking & Traffic:

- • load_balancers
- • data_transfer_daily

Observability:

- • log_groups
- • log_group_usage_daily

Commit Discounts:

- • commitments
- • commitment_utilization_daily

Governance & Metrics:

- • resource_tags
- • metrics_daily

4. Detailed Table Definitions

cloud_accounts

Purpose: Represents a logical cloud account or tenant in the fake environment, mirroring AWS account boundaries.

Supports optimization scenarios:

• Multi-account optimization, per-account reporting, future multi-tenant support.

Key fields:

• id (uuid (PK)) – Internal primary key.

• name (text) – Human-friendly account name (e.g. 'Prod Account', 'Dev Sandbox').

• provider (text) – Cloud provider identifier (e.g. 'aws-fake').

• created_at (timestamptz) – Timestamp of account creation in the sandbox.

instances

Purpose: Simulates EC2-style virtual machines (compute instances).

Supports optimization scenarios:

• Rightsizing compute (oversized instances).

• Idle instance shutdown and off-hours scheduling (dev/test).

• Preview environment cleanup.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Owning AWS cloud account.

• instance_id (text) – AWS-style instance identifier (e.g. 'i-0abc123').

• name (text) – Optional human-friendly name.

• instance_type (text) – Instance type (e.g. 't3.small', 'm5.large').

• gpu_hourly_cost (numeric) – Portion of hourly cost attributed specifically to GPUs (optional).

• gpu_count (integer) – Number of GPUs attached to this instance.

• has_gpu (boolean) – Whether this instance has attached GPU(s).

• env (text) – Environment: 'prod', 'dev', 'staging', 'preview', etc.

• region (text) – Region for this instance (e.g. 'us-east-1').

• state (text) – Lifecycle state: 'running', 'stopped', 'terminated'.

• autoscaling_group_id (uuid (nullable)) – Link to autoscaling_groups if applicable.

• launch_time (timestamptz) – When the instance was launched.

• hourly_cost (numeric) – Simulated hourly on-demand cost for this instance type.

• avg_cpu_7d (numeric) – Simulated average CPU utilization over last 7 days.

• avg_network_7d (numeric) – Simulated average network throughput over last 7 days.

• last_active_at (timestamptz) – Last time this instance did 'real work' in simulation.

• tags (jsonb) – Arbitrary key/value metadata (e.g. owner, team, service).

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

autoscaling_groups

Purpose: Simulates AWS Auto Scaling Groups controlling instance fleets.

Supports optimization scenarios:

• Autoscaling over-provisioning (high minimum capacity).

• Cluster right-sizing and scaling policy tuning.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Owning cloud account.

• name (text) – Autoscaling group name.

• min_size (integer) – Minimum number of instances to keep running.

• max_size (integer) – Maximum number of instances allowed.

• desired_capacity (integer) – Target instance count.

• instance_type (text) – Default instance type used in this group.

• env (text) – Environment label (prod/dev/etc.).

• region (text) – Region for this autoscaling group (e.g. 'us-east-1').

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

container_clusters

Purpose: Simulates ECS/EKS/Kubernetes clusters.

Supports optimization scenarios:

• Abandoned clusters (no active workloads).

• Container density optimization and cluster right-sizing.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• name (text) – Cluster name.

• env (text) – Environment (prod/dev/staging).

• region (text) – Region for this cluster (e.g. 'us-east-1').

• node_instance_type (text) – Default node type (e.g. 'm5.large').

• desired_nodes (integer) – Desired node count.

• min_nodes (integer) – Minimum node count.

• max_nodes (integer) – Maximum node count.

• estimated_hourly_cost (numeric) – Simulated total hourly cost of the cluster.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

container_nodes

Purpose: Represents individual worker nodes inside container_clusters (similar to EC2 backing nodes).

Supports optimization scenarios:

• Node-level utilization tracking.

• Packing containers per node (density) and identifying underutilized nodes.

Key fields:

• id (uuid (PK)) – Internal primary key.

• cluster_id (uuid (FK -> container_clusters.id)) – Owning cluster.

• instance_id (text) – Logical EC2-style identifier.

• instance_type (text) – Node instance type.

• region (text) – Region where this node is running (e.g. 'us-east-1').

• gpu_hourly_cost (numeric) – Portion of hourly cost attributed to GPUs on this node (optional).

• gpu_count (integer) – Number of GPUs on this node.

• has_gpu (boolean) – Whether this node has attached GPU(s).

• state (text) – Running or stopped.

• hourly_cost (numeric) – Simulated hourly cost for this node.

• avg_cpu_7d (numeric) – Average CPU utilization over last 7 days.

• avg_memory_7d (numeric) – Average memory utilization over last 7 days.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

lambda_functions

Purpose: Simulates AWS Lambda/serverless functions with configurable memory and concurrency.

Supports optimization scenarios:

• Lambda over-provisioning (excess memory).

• High concurrency caps driving unnecessary cost.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• name (text) – Function name.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region where this function is deployed (e.g. 'us-east-1').

• memory_mb (integer) – Configured memory size in MB.

• timeout_seconds (integer) – Configured timeout.

• provisioned_concurrency (integer) – Provisioned concurrency value.

• invocations_7d (integer) – Total invocations over last 7 days.

• avg_duration_ms_7d (numeric) – Average duration over last 7 days.

• estimated_monthly_cost (numeric) – Simulated computed cost based on usage and memory.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

volumes

Purpose: Simulates EBS-style block storage volumes.

Supports optimization scenarios:

• Zombie storage (unattached volumes).

• Oversized volumes and unused capacity.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• region (text) – Region where this volume resides (e.g. 'us-east-1').

• volume_id (text) – AWS-style volume identifier (e.g. 'vol-0abc123').

• size_gib (integer) – Allocated size in GiB.

• volume_type (text) – Type (e.g. 'gp2', 'gp3').

• state (text) – Volume state: 'in-use', 'available' (unattached).

• attached_instance_id (text (nullable)) – Linked instances.instance_id if attached.

• monthly_cost (numeric) – Simulated monthly cost for this volume.

• last_used_at (timestamptz) – Last time this volume saw simulated I/O.

• tags (jsonb) – Arbitrary key/value metadata.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

snapshots

Purpose: Simulates EBS snapshots and backups.

Supports optimization scenarios:

• Excess snapshot retention.

• Backup lifecycle optimization (pruning old backups).

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• region (text) – Region for this snapshot (e.g. 'us-east-1').

• snapshot_id (text) – Snapshot identifier.

• source_volume_id (text) – Originating volume_id, if any.

• size_gib (integer) – Logical snapshot size.

• created_at (timestamptz) – Snapshot creation time.

• retention_policy (text) – Optional policy label (e.g. '7d', '30d', '90d').

• monthly_cost (numeric) – Simulated monthly storage cost.

• tags (jsonb) – Metadata tags.

s3_buckets

Purpose: Simulates S3-style object storage buckets.

Supports optimization scenarios:

• Storage tiering (Standard vs IA vs Glacier).

• Cold data retention and lifecycle policies.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• name (text) – Bucket name.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region for this bucket (e.g. 'us-east-1').

• lifecycle_policy (jsonb) – Simulated lifecycle rules (e.g. move to IA after 30d).

• created_at (timestamptz) – Bucket creation time.

• tags (jsonb) – Metadata tags.

s3_bucket_usage_daily
Purpose: Daily aggregated usage metrics for S3 buckets.

Supports optimization scenarios:

• Detecting cold vs hot data patterns.

• Modeling cost impact of tiering and lifecycle changes.

Key fields:

• id (uuid (PK)) – Internal primary key.

• bucket_id (uuid (FK -> s3_buckets.id)) – Associated bucket.

• date (date) – Metric date.

• storage_gb_standard (numeric) – Simulated GB in Standard tier.

• storage_gb_ia (numeric) – Simulated GB in Infrequent Access tier.

• storage_gb_glacier (numeric) – Simulated GB in Glacier/Archive tiers.

• requests_count (bigint) – Total simulated object requests.

• estimated_storage_cost (numeric) – Simulated storage cost for the day.

• estimated_request_cost (numeric) – Simulated request cost for the day.

rds_instances
Purpose: Simulates RDS-style managed database instances.

Supports optimization scenarios:

• DB instance rightsizing (compute class).

• Storage over-provisioning and idle replicas.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• db_instance_id (text) – RDS-style instance identifier.

• engine (text) – DB engine (e.g. 'postgres', 'mysql').

• instance_class (text) – Size/class (e.g. 'db.m5.large').

• allocated_storage_gib (integer) – Allocated storage in GiB.

• env (text) – Environment.

• region (text) – Region for this DB instance (e.g. 'us-east-1').

• state (text) – Lifecycle state: 'available', 'stopped', etc.

• hourly_cost (numeric) – Simulated hourly cost.

• storage_monthly_cost (numeric) – Simulated monthly storage cost.

• avg_cpu_7d (numeric) – Average CPU utilization over last 7 days.

• avg_connections_7d (numeric) – Average connections over last 7 days.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

cache_clusters

Purpose: Simulates Redis/ElastiCache-style in-memory cache clusters.

Supports optimization scenarios:

• Idle or underutilized cache clusters.

• Oversized node types and unnecessary replicas.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• cluster_id (text) – Cache cluster identifier.

• engine (text) – Cache engine (e.g. 'redis').

• node_type (text) – Node instance type.

• num_nodes (integer) – Number of nodes in the cluster.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region for this cache cluster (e.g. 'us-east-1').

• hourly_cost (numeric) – Simulated hourly cost.

• avg_cpu_7d (numeric) – Average CPU utilization.

• avg_memory_7d (numeric) – Average memory utilization.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

load_balancers

Purpose: Simulates ELB/ALB/NLB-style load balancers.

Supports optimization scenarios:

• Orphaned load balancers with no active targets.

• Low-traffic LBs that may be consolidated or removed.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• lb_arn (text) – Load balancer identifier/ARN.

• name (text) – Load balancer name.

• type (text) – Type: 'application', 'network', etc.

• env (text) – Environment.

• region (text) – Region where this load balancer runs (e.g. 'us-east-1').

• hourly_cost (numeric) – Simulated hourly cost.

• avg_request_count_7d (numeric) – Average requests per day.

• created_at (timestamptz) – Row creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

elastic_ips

Purpose: Simulates Elastic IP addresses and their associations to instances or load balancers.

Supports optimization scenarios:

• Identifying unassociated (orphaned) Elastic IPs that continue to incur cost.

• Finding Elastic IPs attached to low-utilization resources.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Owning cloud account.

• allocation_id (text) – Elastic IP allocation identifier.

• public_ip (text) – Assigned public IP address.

• associated_instance_id (text (nullable)) – Linked instances.instance_id if attached to an instance.

• associated_lb_arn (text (nullable)) – Linked load_balancers.lb_arn if attached to a load balancer.

• state (text) – 'associated' or 'unassociated'.

• hourly_cost (numeric) – Simulated hourly cost for this Elastic IP.

• created_at (timestamptz) – Elastic IP allocation timestamp.

• updated_at (timestamptz) – Last update timestamp.

data_transfer_daily

Purpose: Aggregated data transfer metrics to simulate networking and egress cost.

Supports optimization scenarios:

• High cross-AZ/cross-region data transfer cost.

• Internet egress cost hotspots.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• date (date) – Metric date.

• source_region (text) – Source region (e.g. 'us-east-1').

• dest_region (text) – Destination region or 'internet'.

• direction (text) – 'intra-az', 'cross-az', 'cross-region', 'egress' etc.

• gb_transferred (numeric) – Simulated GB transferred.

• estimated_transfer_cost (numeric) – Simulated transfer cost for the day.

streaming_clusters

Purpose: Simulates Kinesis/Kafka-style streaming or messaging clusters.

Supports optimization scenarios:

• Reducing over-provisioned streaming throughput (excess shards/nodes).

• Identifying idle or low-traffic streams that can be consolidated or removed.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Owning cloud account.

• name (text) – Stream or cluster name.

• engine (text) – 'kinesis', 'kafka', or other engine identifier.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region for this streaming cluster (e.g. 'us-east-1').

• shard_count (integer) – Number of shards/partitions or equivalent.

• retention_hours (integer) – Data retention period in hours.

• provisioned_throughput_mbps (numeric) – Provisioned throughput.

• avg_usage_mbps_7d (numeric) – Average throughput usage over last 7 days.

• hourly_cost (numeric) – Simulated hourly cost for this streaming cluster.

• created_at (timestamptz) – Creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

log_groups

Purpose: Simulates CloudWatch log groups or equivalent logging sinks.

Supports optimization scenarios:

• Excess log retention and volume.

• Debug logging left enabled in non-dev environments.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• name (text) – Log group name.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region for this log group (e.g. 'us-east-1').

• retention_days (integer) – Configured retention period in days.

• created_at (timestamptz) – Log group creation timestamp.

• tags (jsonb) – Metadata tags.

log_group_usage_daily

Purpose: Daily aggregated log ingestion and storage per log group.

Supports optimization scenarios:

• Detecting noisy log groups and unnecessary retention.

• Estimating savings from log sampling or shorter retention.

Key fields:

• id (uuid (PK)) – Internal primary key.

• log_group_id (uuid (FK -> log_groups.id)) – Associated log group.

• date (date) – Metric date.

• ingested_gb (numeric) – Simulated GB of logs ingested.

• stored_gb (numeric) – Simulated GB of logs stored.

• estimated_ingestion_cost (numeric) – Simulated cost of ingestion.

• estimated_storage_cost (numeric) – Simulated cost of log storage.

commitments

Purpose: Simulates Reserved Instances and Savings Plans commitments.

Supports optimization scenarios:

• Detecting gaps where no commitments exist despite steady usage.

• Monitoring underutilized commitments.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• commitment_type (text) – 'reserved_instance' or 'savings_plan' etc.

• scope (text) – Scope description (e.g. region/service/instance family).

• term_months (integer) – Term length in months (e.g. 12, 36).

• hourly_commitment_amount (numeric) – Committed hourly spend or usage.

• start_date (date) – Commitment start date.

• end_date (date) – Commitment end date.

• created_at (timestamptz) – Row creation timestamp.

commitment_utilization_daily

Purpose: Daily utilization metrics for commitments.

Supports optimization scenarios:

• Finding underutilized commitments.

• Modeling potential for additional commitments.

Key fields:

• id (uuid (PK)) – Internal primary key.

• commitment_id (uuid (FK -> commitments.id)) – Associated commitment.

• date (date) – Metric date.

• actual_hourly_usage_equivalent (numeric) – Simulated average hourly usage applying to this commitment.

• utilization_percent (numeric) – Usage vs commitment percentage for the day.

• estimated_savings_vs_ondemand (numeric) – Simulated savings compared to on-demand for the day.

resource_tags

Purpose: Simulates AWS tagging: arbitrary key/value metadata for any resource type.

Supports optimization scenarios:

• Ownership and cost allocation governance.

• Identifying untagged/orphaned resources.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• resource_type (text) – Type of resource (e.g. 'instance', 'volume', 'rds_instance').

• resource_id (text) – Resource identifier within its table (e.g. instance_id, volume_id).

• key (text) – Tag key (e.g. 'owner', 'team', 'env').

• value (text) – Tag value (e.g. 'data-platform', 'billing').

• created_at (timestamptz) – Row creation timestamp.

metrics_daily

Purpose: Generic daily metric rollup for any resource type, for simplified reporting and dashboards.

Supports optimization scenarios:

• Cost and usage reporting across all resource types.

• Global optimization reporting and historical charts.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK)) – Owning cloud account.

• resource_type (text) – Resource type (e.g. 'instance', 'volume', 'rds_instance').

• resource_id (text) – Resource identifier within its type.

• date (date) – Metric date.

• metric_payload (jsonb) – Flexible metrics (CPU, storage, traffic, etc.).

• estimated_daily_cost (numeric) – Simulated total daily cost for this resource.

• created_at (timestamptz) – Row creation timestamp.

managed_services

Purpose: Simulates generic managed service clusters (e.g. search, analytics, feature stores) that may be over-provisioned or duplicated.

Supports optimization scenarios:

• Managed service consolidation (e.g. multiple small search clusters per team).

• Rightsizing node counts and SKUs for low-utilization managed services.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Owning cloud account.

• service_type (text) – Service type (e.g. 'opensearch', 'elasticsearch', 'feature_store').

• name (text) – Service/cluster name.

• env (text) – Environment (prod/dev/etc.).

• region (text) – Region for this managed service (e.g. 'us-east-1').

• instance_type (text) – Node instance type or SKU.

• node_count (integer) – Number of nodes in the cluster.

• hourly_cost (numeric) – Simulated hourly cost.

• avg_cpu_7d (numeric) – Average CPU utilization.

• avg_memory_7d (numeric) – Average memory utilization.

• created_at (timestamptz) – Creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

container_services

Purpose: Optional layer to represent logical services or deployments running on container_clusters.

Supports optimization scenarios:

• Container density and bin-packing at the service level (per deployment).

• Identifying services with large requested resources but low actual utilization.

Key fields:

• id (uuid (PK)) – Internal primary key.

• cluster_id (uuid (FK -> container_clusters.id)) – Owning cluster.

• name (text) – Service/deployment name.

• env (text) – Environment (prod/dev/staging/etc.).

• region (text) – Region where this service runs (e.g. 'us-east-1').

• requested_cpu (numeric) – Total requested CPU for all replicas.

• requested_memory_mb (integer) – Total requested memory for all replicas.

• replica_count (integer) – Number of replicas/pods.

• avg_cpu_7d (numeric) – Average CPU utilization across replicas.

• avg_memory_7d (numeric) – Average memory utilization across replicas.

• created_at (timestamptz) – Creation timestamp.

• updated_at (timestamptz) – Last update timestamp.

5. How This Project Will Be Used by the Agentic AI Platform

This AWS cloud environment is the data plane. Drift functions (Supabase Edge Functions or scheduled jobs) will continuously modify these tables to introduce new resources, create idle workloads, leave zombie storage, and grow usage. The Agentic AI platform (in a separate project) will connect via HTTP APIs or a Supabase client to:
- Discover resources across compute, storage, databases, networking, and observability.
- Detect waste based on utilization, environment, state, and cost fields.
- Propose and simulate optimization actions (stop, delete, resize, tier).
- Calculate savings using the cost fields and metrics_daily rollups.

This separation mirrors a real-world setup where a customer cloud account is managed by an external optimization platform like Heliozz.

6. API Access and IAM-style Single Endpoint

To connect the Agentic AI optimization platform (e.g. HNTKi/Heliozz) to this AWS cloud environment, we expose a single HTTP endpoint that behaves like an IAM-authenticated cloud API.

At a high level:

• The AWS cloud environment defines IAM-like API clients with access keys and scoped permissions.

• The Agentic AI platform makes all read/write calls (list, get, update, simulate) through one endpoint, passing its access key.

• Internally, the endpoint dispatches to the appropriate tables (instances, volumes, rds_instances, etc.) based on the requested resource_type and action.

api_clients

Purpose: Represents IAM-style API clients that can call the single AWS cloud endpoint.

Key fields:

• id (uuid (PK)) – Internal primary key.

• account_id (uuid (FK -> cloud_accounts.id)) – Cloud account this client belongs to.

• name (text) – Client name (e.g. 'hntki-platform').

• access_key_id (text) – Public API key identifier.

• secret_key_hash (text) – Hashed secret key for authentication.

• scopes (jsonb) – Allowed actions/resources (e.g. ['read:*', 'write:instances']).

• created_at (timestamptz) – Creation timestamp.

• last_used_at (timestamptz) – Last time this client was used.

Example endpoint shape (conceptual):

• URL: POST /fakecloud

• Headers: X-Access-Key and X-Signature (or similar) for authentication.

• Body: { "action": "list_resources", "resource_type": "instances", "filters": { "env": "prod" }, "limit": 100 }

The implementation details of the endpoint and signature format are up to the team, but from the Agentic AI perspective there is exactly one gateway into the AWS cloud environment, similar to using a single AWS IAM user with programmatic access.