ANTHROPIC CLAUDE & WORKSPACE STUDIOS • FIELD GUIDE

Claude Managed Agents & MCP Connectors

Architecting long-lived autonomous agents without DIY loop plumbing—decoupling the reasoning brain and PostgreSQL session store from disposable sandboxes, OAuth vaults, and remote HTTPS/SSE `/mcp` servers.

Brain / Hands Decoupling | Self-Hosted `cma-server` | Remote HTTPS/SSE `/mcp` | OAuth 2.1 & Secret Vaults | Bedrock • Vertex • Foundry | Zero-Secret Sandboxes



Zero DIY Harness Plumbing
Skip fragile custom retry loops—Managed Agents handle tool turns, streaming checkpoints, and durable state recovery out of the box.

Vault-Isolated Credentials
Store OAuth tokens and API keys inside server-side Vaults so sandboxed bash containers never touch raw secrets.

Cloud or VPC Sovereignty
Run on Anthropic's managed cloud or self-host `cma-server` with PostgreSQL and route inference to Bedrock, Vertex, or Foundry.

Decoupling the Agent Brain From the Hands

Traditional agent scripts tangle LLM reasoning, state persistence, untrusted shell commands, and raw API keys inside one brittle process. **Claude Managed Agents** separate the stateful reasoning loop (“brain”) from disposable execution containers and authenticated remote `/mcp` connectors (“hands”).

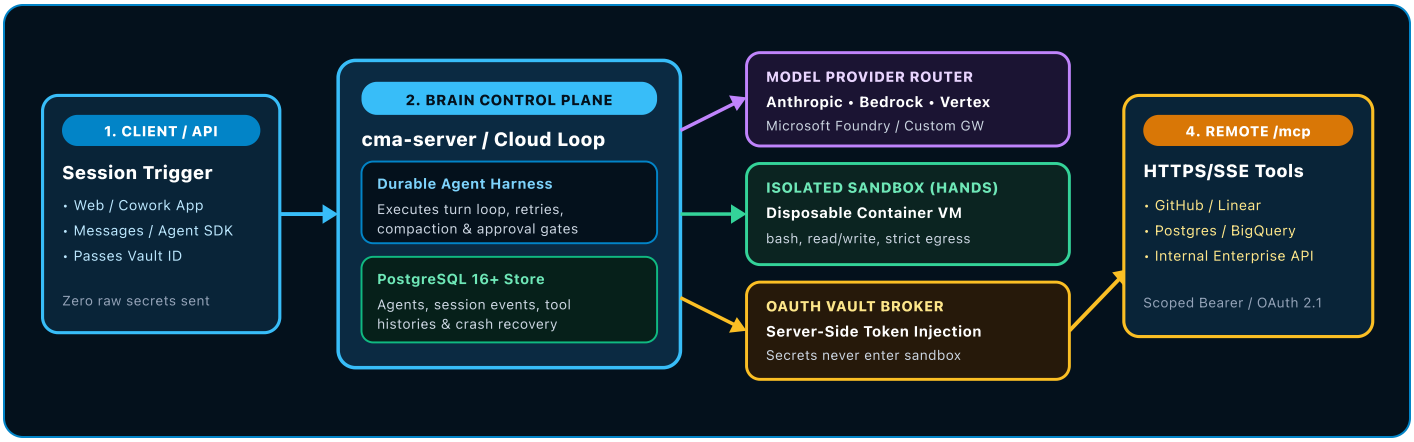
Quick Navigation

1. Decoupled Brain, Store & Sandbox Model p. 2
2. Prerequisites & Deployment Topology Matrix p. 3
3. Annotated Control Plane & Connector UI p. 4
4. Why Managed Loops Beat DIY Harnesses p. 5
5. Color-Coded Config & Schema Anatomy p. 6
6. Remote HTTPS/SSE MCP & OAuth Vaults p. 7
7. Self-Hosting `cma-server` in Your VPC p. 8
8. 5 Copy-Paste Agent & MCP Blueprints p. 9
9. Blast-Radius Containment & Schema Hygiene p. 10
10. Skills + `cma-server` + MCP Pipeline p. 11
11. Troubleshooting Clinic & Pre-Flight p. 12
Golden Rule: Never Put Secrets in Sandboxes All

Core Mental Model: Four Independent Planes

An agent definition declares *instructions* and *tool URLs*—never credentials. When a session starts, the control plane (Anthropic Cloud or your self-hosted `cma-server`) persists every turn in PostgreSQL, routes model calls to your chosen inference provider, mounts a clean filesystem container for code execution, and brokers remote `/mcp` calls through a server-side OAuth Vault.

Architecture: End-to-End Request & Isolation Flow



Key Takeaway: Sandboxes Execute Code, Vaults Call Remote MCP

Inside Claude Managed Agents, file and shell tools run inside your isolated container environment, while **remote `/mcp` servers are called server-side** through the Vault proxy. Even if untrusted code runs in bash, it cannot read your GitHub or Stripe OAuth tokens.

Prerequisites & Deployment Topology Matrix

Not every workflow needs a self-hosted cluster, and not every connector belongs on a developer laptop. Use the topology matrix below to pick the exact runtime and MCP transport that matches your security, reachability, and residency requirements.

1 Baseline Prerequisites

- **Remote Connectors:** A TLS-terminated HTTPS endpoint exposing `/mcp` (Streamable HTTP or SSE) plus OAuth 2.1 metadata or static Bearer headers.
- **Cloud Managed Agents:** Anthropic Console API key, declared Environment ID, and Vault IDs for authenticated MCP calls.
- **Self-Hosted `cma-server`:** Container host (Kubernetes or Docker), **PostgreSQL 16+**, and an upstream model credential.

2 Supported Inference Providers

- **Anthropic API:** Direct first-party access to the latest Claude Opus, Sonnet, and Haiku releases with native Vault integration.
- **Amazon Bedrock:** Route `cma-server` turns via AWS IAM roles to keep inference traffic inside your AWS account.
- **Google Cloud Vertex AI & Microsoft Foundry:** Enterprise billing, VPC Service Controls, and Azure private-link routing without changing agent code.

Deployment Topology Decision Matrix

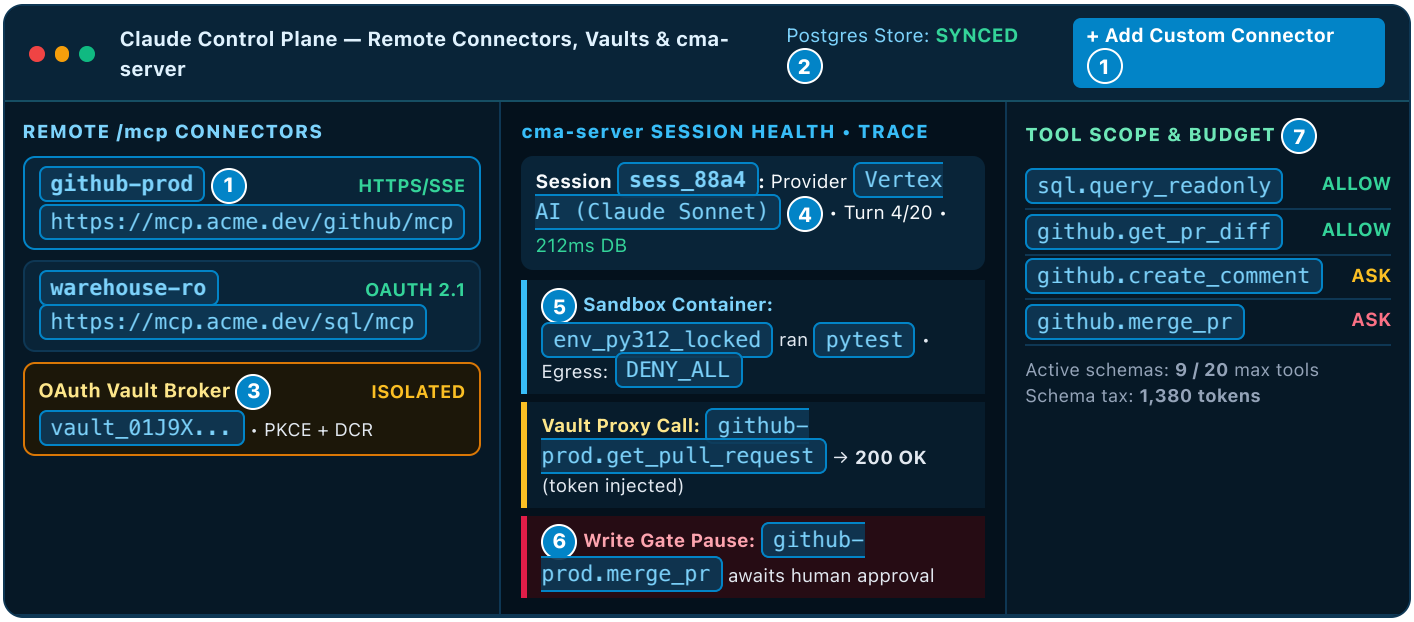
Topology	Runtime & State	Network Reachability	Secrets Isolation	Best Fit & Providers
Local `stdio` MCP Claude Desktop / Code	Local child process on user laptop; ephemeral session.	Localhost & VPN; no public URL required.	Local env vars or OS keychain on developer machine.	Individual dev tools, local Git/SQLite; Anthropic API or Bedrock CLI auth.
Remote HTTPS/SSE Custom Connector Web, Mobile, Cowork	Cloud-connected; stateless or MCP server-managed.	Public HTTPS `/mcp` (or allowlisted Anthropic IPs).	OAuth 2.1 PKCE, DCR, or enterprise IdP token exchange.	Shared SaaS tools (Linear, Notion, internal APIs) across Web & Mobile.
Cloud Claude Managed Agents Anthropic Control Plane	Fully managed loop + persistent session store + cloud VM.	Cloud-to-cloud HTTPS `/mcp`; sandbox egress rules.	**Vaults:** tokens injected server-side; invisible to sandbox.	Turnkey async bots, triage agents, & background research on Anthropic API.
Self-Hosted `cma-server` Your VPC / On-Prem	Your container fleet + **PostgreSQL 16+** persistence.	Private VPC mesh; reaches internal `/mcp` without public exposure.	K8s/HashiCorp Vault + private gateway auth; zero external data store.	Regulated finance/health, private repos; **Anthropic, Bedrock, Vertex, Foundry**.

Rule of Thumb: Move From Local to Managed When Sessions Outlive Your Laptop

If an agent needs to survive browser disconnects, run unattended overnight, share across Web/Mobile teammates, or keep internal API keys out of untrusted code-execution containers, graduate from local `stdio` to **Remote `/mcp` + Managed Agents**.

Annotated Tour: Connectors, Vaults & Session Monitor

Whether you register connectors in Claude Settings or orchestrate headless sessions via `cma-server`, four unified panels govern every run: the Remote `/mcp` registry, live session loop telemetry, the OAuth Vault broker, and per-tool read/write gates.



Control Plane & Connector Callout Legend

1 **Remote `/mcp` Registry:** Registers HTTPS/SSE endpoints (`https://.../mcp`) across Web, Mobile, Cowork, and Managed Agents without local binary installs.

2 **PostgreSQL Session Store:** Persists `agents`, `sessions`, and event logs inside PostgreSQL 16+ so long-running jobs survive restarts and scale horizontally.

3 **OAuth Vault Token Broker:** Holds user OAuth refresh tokens and service keys outside the model context and injects headers server-side at egress.

4 **Model Provider Router:** Routes inference from `cma-server` to Anthropic API, Amazon Bedrock, Google Cloud Vertex, or Microsoft Foundry per policy.

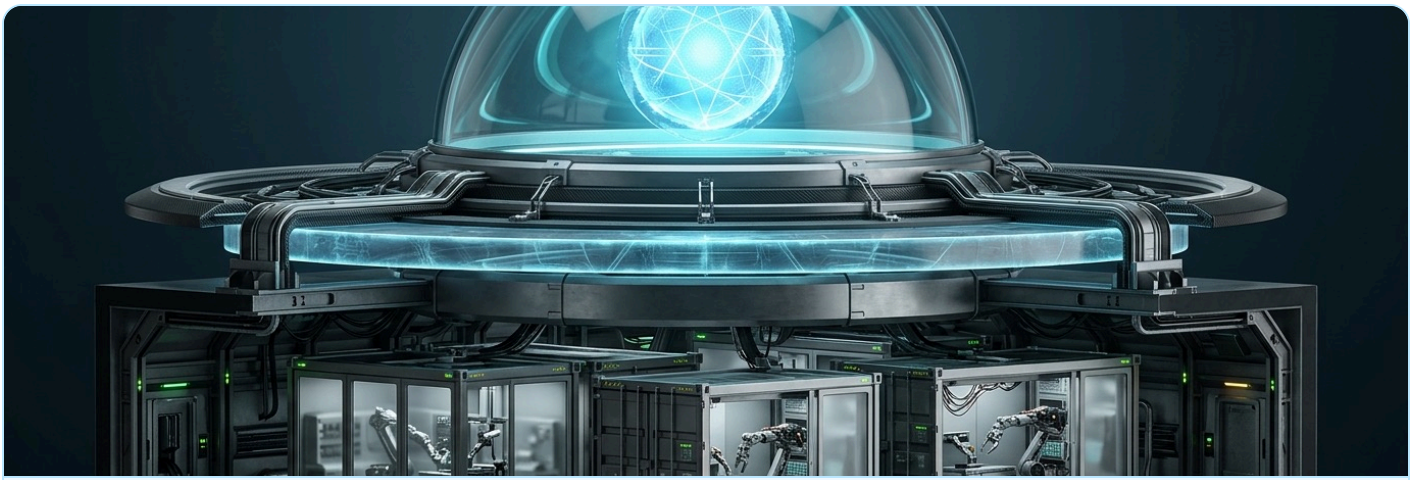
5 **Sandbox Egress Policy:** Confines `bash` and file tools to disposable containers with explicit `allowed_hosts` — blocking arbitrary data exfiltration.

6 **Per-Tool Read/Write Gates:** Auto-runs safe read queries while pausing the session loop whenever an agent attempts a mutating write, merge, or spend action.

7 **Schema Budget & Tool Curation Gauge:** Tracks active MCP tool definitions loaded into the session context. Capping each agent to 8–15 focused tools prevents context bloat and sharpens tool selection accuracy.

Decoupling the Brain From the Hands

Building your own production agent loop looks simple on day one—until timeouts, partial tool errors, context compaction, and shell escapes break in production. Managed Agents separate the durable cognitive loop from isolated execution workspaces.



Architectural Separation: The cognitive loop reasons above a strict trust boundary, dispatching deterministic tool commands down to disposable sandbox containers while keeping secrets and session state safely isolated.

1 What the “Brain” Owns (Control Plane)

- **Multi-Turn Loop Management:** Streams model responses, parses tool calls, handles exponential backoff on transient 429/5xx errors, and enforces max-turn budgets.
- **Session Checkpointing:** Writes every message, tool request, and artifact pointer into durable storage so clients can disconnect and resume hours later.
- **Context Compaction:** Summarizes stale tool outputs automatically when long investigations approach context limits without losing active goals.

2 What the “Hands” Own (Sandboxes)

- **Disposable Workspaces:** Each session boots inside an isolated container with scratch disk for cloning repos, running tests, and compiling reports.
- **Blast-Radius Containment:** A runaway `rm -rf` or malformed Python package affects only that ephemeral container—never the control plane or sibling tenants.
- **Network Egress Rules:** Lock outbound container traffic to explicit package registries or `DENY_ALL` so untrusted scripts cannot phone home.

Why You Should Stop Writing Custom `while True` Tool Loops

A DIY harness must implement prompt caching, streaming SSE recovery, tool-timeout interrupts, human-in-the-loop pause/resume, and sandbox cleanup. Managed Agents bundle all five behind four clean primitives: **Agents**, **Environments**, **Vaults**, and **Sessions**.

Color-Coded Remote MCP & `cma-server` Anatomy

Managed Agents keep configuration cleanly partitioned: agent definitions declare endpoints and prompts, sessions bind runtime Vaults, and `cma-server` wires PostgreSQL persistence, model routing, and sandbox egress.

Color-Coded 5-Layer Managed Agent & Server Stack

[Remote /mcp Endpoint] [OAuth / Vault Broker] [Model Provider Router] [PostgreSQL Persistence] [Sandbox Egress Policy]

```
# 1. Agent Definition: declare url-type MCP servers cleanly with zero inline secrets
mcp_servers: [{ type: "url", name: "linear-ops", url: "https://mcp.acme.internal/linear/mcp" }]
# 2. Session Creation: attach user-scoped Vault ID at runtime for server-side token injection
session.create(agent_id="agt_triage", environment_id="env_locked", vault_ids=["vault_usr_91f"])
# 3. Model Provider Router: direct inference to Anthropic, Bedrock, Vertex, or Foundry
CMA_PROVIDER__DEFAULT="vertex" | VERTEX_REGION="us-east5" | MODEL="claude-sonnet-4-5"
# 4. PostgreSQL Persistence: durable agents, session logs, and event listener
CMA_STORE__DATABASE_URL="postgres://cma:***@pg-primary.internal:5432/managed_agents"
# 5. Sandbox Egress Policy: restrict container network to vetted registries only
environment.networking: { type: "limited", allowed_hosts: ["pypi.org", "registry.npmjs.org"] }
```

1 Agent vs. Session Separation

- **Versioned Blueprints:** Commit Agent JSON/YAML definitions to Git. Because `mcp_servers` holds only `name`, `type`, and `url`, PR reviews stay 100% secret-free.
- **Per-User Vault Binding:** Ten engineers can share one `agt_triage` blueprint while each session injects that engineer's own OAuth identity via `vault_ids`.
- **Tool Filtering:** Allowlist only required MCP tool names on the agent so noisy unused endpoints never pollute the prompt.

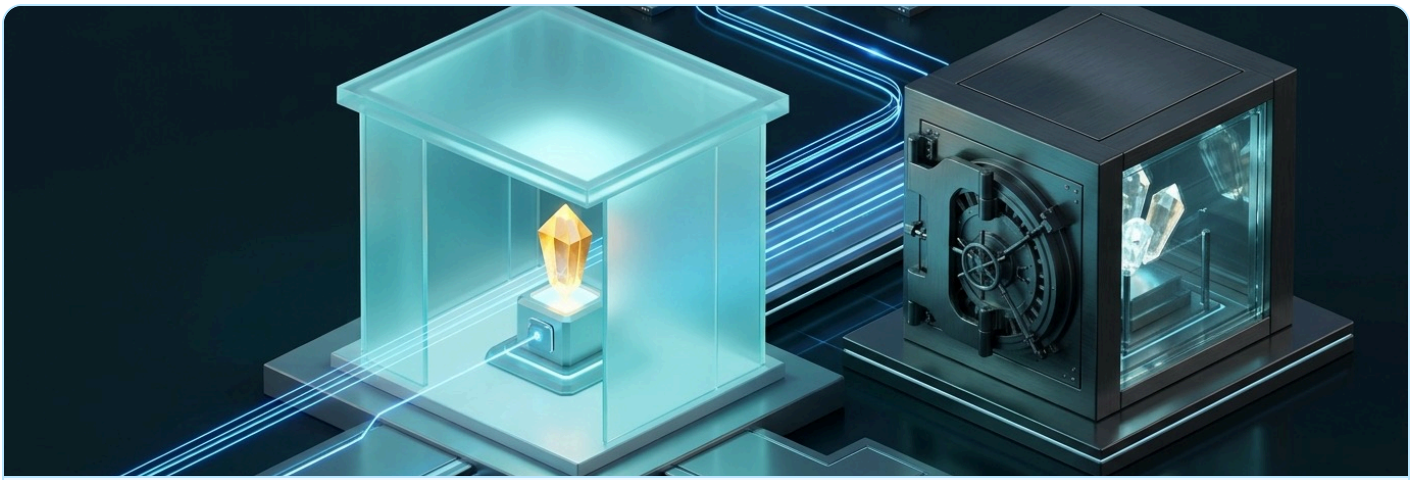
2 Environment & Networking Controls

- **Pinned Toolchains:** Pre-bake Python, Node.js, and internal CLI linters into an Environment image so sessions skip 40-second `pip install` cold starts.
- **Strict Egress Allowlists:** Set `allowed_hosts` to exact domains (or disable internet entirely) while letting server-side MCP calls still reach SaaS APIs.
- **Resource Caps:** Enforce memory, disk, and execution-time ceilings per sandbox container to stop infinite loops cleanly.

```
# Declarative Agent + Environment payload pattern (JSON): zero credentials embedded
{
  "name": "incident-investigator",
  "model": "claude-sonnet-4-5",
  "mcp_servers": [{ "type": "url", "name": "datadog-ro", "url": "https://mcp.acme.dev/datadog/mcp" }],
  "tools": [{ "type": "agent_toolset_20260401" }, { "type": "mcp_toolset", "mcp_server_name": "datadog-ro" }]
}
```

Remote HTTPS/SSE Connectors & OAuth Vaults

Remote MCP moves integrations off local laptops onto standard `https://.../mcp` endpoints. Combined with OAuth 2.1 and Vaults, one connector works seamlessly across Claude Web, Mobile, Cowork, and headless Managed Agents.



Zero-Trust Credential Brokering: The frosted cryptographic gatehouse exchanges scoped OAuth tokens for outbound `https://.../mcp` calls while long-lived secrets remain locked inside the Vault—never exposed to the sandbox.

1 Four Remote Auth Patterns

- **OAuth 2.1 + PKCE & PRM:** Server publishes Protected Resource Metadata (`.well-known/oauth-protected-resource`) so Claude discovers scopes automatically.
- **Dynamic Client Registration (DCR):** RFC 7591 lets Claude register as an OAuth client dynamically without manual admin client-ID copying.
- **Enterprise IdP Token Exchange:** Org admins federate Okta/Entra ID so employees inherit scoped access silently without 10 separate login popups.
- **Static Bearer / Vault Headers:** For machine-to-machine services, store the API key inside a Vault credential rather than pasting headers in plaintext.

2 Transport & Reachability Checklist

- **Streamable HTTP First:** Prefer MCP's modern Streamable HTTP (`POST /mcp` with optional SSE stream) over legacy dual-endpoint SSE for cleaner load balancing.
- **Disable Proxy Buffering:** On Nginx, Cloudflare, or ALB, disable response buffering on `/mcp` so streaming tool progress never stalls until timeout.
- **Firewall & Private Mesh:** Cloud connectors require public HTTPS or Anthropic IP allowlisting; self-hosted `cma-server` reaches private VPC `/mcp` servers directly.

Cross-Surface Payoff: Connect Once, Use Everywhere

Adding a custom connector under `Settings → Connectors → Add Custom Connector` makes that authenticated `https://.../mcp` server instantly usable from desktop, web browsers, and mobile Claude apps—no Node.js or Python setup on user devices.

Self-Hosting `cma-server` in Your VPC

Regulated industries and teams with internal-only microservices can run the entire Managed Agents control plane inside their own Kubernetes cluster or VPC using `cma-server`—retaining 100% session sovereignty.

1 PostgreSQL 16+ Persistence

Without a database, `cma-server` falls back to volatile memory and wipes sessions on restart. Production clusters wire `CMA_STORE__DATABASE_URL` to **PostgreSQL 16+** so agent specs, turn checkpoints, and audit trails persist durably across pod rollouts.

2 Control & Listen Connections

High-throughput deployments configure dedicated `CMA_STORE__CONTROL_DATABASE_URL` (heartbeats and maintenance) and `CMA_STORE__LISTEN_DATABASE_URL` (PostgreSQL `LISTEN/NOTIFY` events). Avoid transaction-mode poolers like PgBouncer on the listener URL.

3 Inference Gateway Routing

Route model calls out of `cma-server` directly to **Anthropic API**, **Amazon Bedrock**, **Google Cloud Vertex AI**, **Microsoft Foundry**, or an internal LiteLLM/Envoy proxy for unified spend caps, PII scrubbing, and regional failover.

4 Horizontal Scaling & Failover

Because session state lives in PostgreSQL rather than pod RAM, run 3+ stateless `cma-server` replicas behind an internal load balancer. If one node crashes mid-turn, another replica resumes the exact event stream seamlessly.

Production `cma-server` Environment & Kubernetes Pattern

```
# Self-hosted cma-server core environment wiring (inject passwords via K8s Secrets / External Secrets)
CMA_STORE__DATABASE_URL="postgres://cma_app:${DB_PASS}@pg-ha.internal:5432/cma?sslmode=require"
CMA_STORE__CONTROL_DATABASE_URL="postgres://cma_app:${DB_PASS}@pg-ha.internal:5432/cma?sslmode=require"
CMA_STORE__LISTEN_DATABASE_URL="postgres://cma_app:${DB_PASS}@pg-direct.internal:5432/cma?sslmode=require" # No PgBouncer txn pooling
ANTHROPIC_BEDROCK_BASE_URL="https://bedrock-runtime.us-east-1.amazonaws.com"   # Or Vertex / Foundry / Anthropic API
```

Sovereignty Requirement	How Self-Hosted `cma-server` Solves It	Operational Tip
Zero Public Exposure for Internal APIs	`cma-server` calls cluster-local `https://*.svc.cluster.local/mcp` services inside your VPC.	Enforce mTLS or network policies so only the `cma-server` namespace can reach internal MCP pods.
Data Residency & Audit Logs	All conversation transcripts, tool outputs, and agent artifacts remain inside your PostgreSQL & object store.	Schedule encrypted PostgreSQL backups and set retention TTLs on archived session rows.
Multi-Region Model Failover	Swap or failover between Bedrock, Vertex, and Foundry endpoints without touching client SDK code.	Pair with health checks on your inference gateway and configure per-agent token rate limits.

5 Real-World Managed Agent & MCP Blueprints

Use these five production blueprints as starting templates. Notice how every design isolates secrets in Vaults, trims exposed MCP tools to essentials, and pairs sandboxed execution with clear human approval boundaries.

Blueprint & Hosting	Remote `/mcp` Stack & Auth	Sandbox Work & Approval Boundary
1. On-Call Incident Investigator `cma-server` (VPC) Bedrock / Vertex	**MCP:** `datadog-ro` + `github-ro` + `pagerduty` via internal VPC HTTPS. **Auth:** Service Vault token (read-only).	**Sandbox:** Checks out commit SHA, runs reproduction script inside `DENY_ALL` container. **Gate:** Posts root-cause brief to Slack; requires human click before reverting PRs.
2. Autonomous PR Reviewer Cloud Managed Agents Anthropic API	**MCP:** `https://mcp.acme.dev/github/mcp` (filtered to 4 PR tools). **Auth:** GitHub App OAuth inside Org Vault.	**Sandbox:** Runs unit tests, linters, and migration checks cleanly. **Gate:** Auto-posts line comments; `merge_pull_request` blocked behind maintainer approval.
3. RevOps Weekly Cohort Analyst Cloud Managed Agents Web + Scheduled API	**MCP:** `snowflake-ro` + `hubspot-ro` + `slack-post`. **Auth:** Per-analyst OAuth 2.1 PKCE Vault.	**Sandbox:** Runs Python pandas cohort reconciliation and builds `.xlsx` + `.pptx` outputs. **Gate:** Delivers draft deck to private channel before executive broadcast.
4. Support Escalation Triage `cma-server` + Cowork Microsoft Foundry	**MCP:** `zendesk` + `linear` + internal `billing-ro`. **Auth:** Entra ID IdP token exchange.	**Sandbox:** Parses uploaded HAR logs and stack traces without risking host OS. **Gate:** Creates Linear bug automatically; refunds/credits pause for support lead sign-off.
5. Overnight Security Patcher `cma-server` (K8s) PostgreSQL 16+	**MCP:** `snyk-vulns` + `gitlab-internal`. **Auth:** Short-lived Vault tokens scoped to `dev-patches/*` branches.	**Sandbox:** Upgrades vulnerable lockfiles, runs full test suite, and verifies build. **Gate:** Opens draft MR with diff proof; zero direct push permissions to `main`.

Starter Session Launch Pattern (Agent + Environment + Vault Binding)

```
# Bind a versioned, secret-free agent blueprint to an isolated sandbox environment and user Vault at session start
session = client.beta.sessions.create(
    agent=agent_id,                          # Declares instructions + https://.../mcp URLs only
    environment_id="env_py312_restricted",   # Disposable container with allowed_hosts egress policy
    vault_ids=[user_oauth_vault_id],         # Server-side token broker; sandbox never sees raw credentials
    title="PR #482 Regression Triage"
)
```

Security, Blast-Radius Containment & Schema Hygiene

Connecting autonomous agents to live company systems introduces two distinct risks: oversized blast radii if a tool mutates data unchecked, and indirect prompt injection hidden inside external tickets, PR comments, or database rows.

Read vs. Write Gates

Separate read tools (`get_*`, `search_*`, `query_*`) from write tools (`delete_*`, `merge_*`, `send_*`). Auto-approve reads so agents move fast, and enforce human confirmation on every external state change.

Anti-Tool-Poisoning

Treat every third-party MCP tool description and returned payload as **untrusted data**. Pin vetted server versions, strip hidden HTML/markdown instructions, and never mix untrusted web-scrape tools with privileged write keys in one session.

MCP Schema Hygiene

Every MCP tool injects its JSON schema into the system prompt (~150–300 tokens each). Loading 60 tools wastes 12,000+ tokens before Turn 1 and lowers accuracy. Filter each agent down to **5–15 sharp tools**.

Four-Layer Defense Matrix for Production MCP Deployments

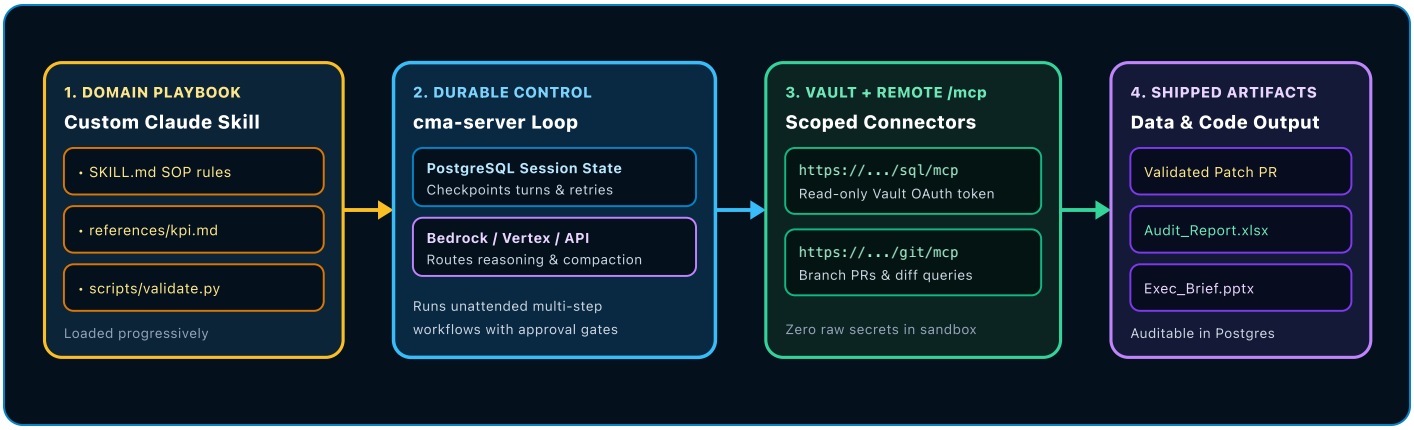
Threat Vector	Failure Mode Without Controls	Managed Agents + MCP Defense
1. Secret Exfiltration via Bash	Prompt-injected code inside sandbox prints `env` or `curl`s API tokens to an attacker host.	Store tokens in server-side **Vaults** (never container env vars) and lock sandbox networking to `allowed_hosts`.
2. Indirect Prompt Injection	Malicious text inside a Jira ticket tells the agent to export customer emails via Slack MCP.	Require confirmation gates on outbound/write tools and scope OAuth tokens strictly to least-privilege read scopes.
3. Tool Poisoning / Rug Pulls	An external MCP server silently mutates its tool description after approval to hijack model behavior.	Audit remote servers before allowlisting, pin verified internal gateways, and alert on unexpected tool schema diffs.
4. Context Bloat & Wrong Tool Pick	Connecting 5 full SaaS servers dumps 80 overlapping tools, burning tokens and causing hallucinated parameters.	Split into specialized role agents (Triage vs. Finance) and allowlist only exact needed tool names per agent.

Never Bridge Untrusted Public Inputs With Unattended Admin Write Scopes

If an agent reads public GitHub issues, incoming support emails, or arbitrary web pages, assume adversarial text will arrive. Keep that session's Vault tokens read-only or gate every write action behind human approval.

Cross-Tool Synergy: Skills + Managed Agents + MCP

Claude Skills teach your team's rules and formatting standards, `cma-server` orchestrates durable multi-turn loops, and **Remote** `/mcp` **Connectors** supply live authenticated data—producing verified Data and Code artifacts automatically.



End-to-End Walkthrough: Automated Weekly Engineering & Revenue Audit

1. Load Skill Playbook: Your `revops-audit` Skill instructs Claude on exact ARR exclusion rules, chart styles, and validation scripts without cluttering idle prompts.

2. Orchestrate via `cma-server`: A scheduled job opens a Managed Agent session backed by PostgreSQL and routes reasoning through Vertex AI or Bedrock.

3. Query via Vault-Protected `/mcp`: The Vault injects scoped OAuth tokens to pull live warehouse metrics and GitHub release notes server-side.

4. Verify in Sandbox & Ship: Inside the isolated container, Claude runs `scripts/validate.py` and emits checked `.xlsx`, `.pptx`, or PR artifacts.

Architecture Payoff: Clean Separation of Concerns

Swap models without rewriting skills, rotate OAuth credentials in Vaults without redeploying agents, and upgrade `/mcp` tools without touching sandbox images.

Troubleshooting Clinic & Pre-Flight Checklist

Most Managed Agent and Remote MCP snags come from proxy buffering, OAuth redirect URIs, PgBouncer listener pooling, or bloated tool lists. Use this quick clinic to diagnose and resolve issues in seconds.

Symptom	Root Cause	10-Second Fix
1. Remote `/mcp` hangs or times out	Reverse proxy (Nginx/ALB/Cloudflare) buffers SSE chunks instead of streaming immediately.	Disable proxy buffering on `/mcp` (e.g., `proxy_buffering off;`) and set idle timeouts above 300s.
2. OAuth handshake fails on connect	Missing `.well-known` PRM metadata, strict redirect mismatch, or blocked DCR endpoint.	Verify RFC 9728 metadata, allowlist Claude's OAuth callback URL, or supply custom OAuth client credentials.
3. `cma-server` loses sessions on restart	`CMA_STORE__DATABASE_URL` unset, causing fallback to ephemeral in-memory store.	Point `CMA_STORE__DATABASE_URL` to PostgreSQL 16+ and confirm schema migrations succeed at boot.
4. DB event listener disconnects randomly	`CMA_STORE__LISTEN_DATABASE_URL` routed through a transaction-mode PgBouncer pooler.	Connect `LISTEN_DATABASE_URL` directly to PostgreSQL (or session-mode pooler) so `LISTEN/NOTIFY` stays open.
5. Sandbox bash fails to hit internal API	Sandbox egress policy blocks outbound calls, or developer tried calling Vault secrets inside bash.	Route private API access through server-side `/mcp` + Vault instead of container scripts; allowlist only package hosts.
6. Agent picks wrong MCP tool or stalls	Too many overlapping tool schemas (50+) bloat prompt context and confuse parameter selection.	Trim `mcp_toolset` down to 5–15 sharp tools per agent and write crisp single-purpose parameter descriptions.
7. 401/403 from Bedrock, Vertex, or Foundry	Expired cloud IAM role, missing region quota, or mismatched model ID mapping in router.	Verify cloud IAM/OIDC permissions, confirm model availability in target region, and test gateway health endpoint.

6-Point Pre-Flight Checklist Before Shipping Any Managed Agent

☑ **1. Zero Secrets in Blueprints:** Agent JSON/YAML lists only `https://.../mcp` URLs; all tokens live inside Vaults.

☑ **2. PostgreSQL 16+ Wired:** Self-hosted `cma-server` points to durable PostgreSQL with direct listener connectivity.

☑ **3. Tight Sandbox Egress:** Container environment locks outbound traffic to exact `allowed_hosts` or `DENY_ALL`.

☑ **4. Read/Write Gates Active:** Mutating, merging, billing, or external messaging MCP tools pause for human sign-off.

☑ **5. Lean Tool Budget (≤15):** Redundant MCP tools are filtered out so schema overhead stays crisp and accurate.

☑ **6. Streaming & Auth Tested:** Verified unbuffered `/mcp` SSE/HTTP responses, OAuth token refresh, and provider failover.