AVIATRIX

TECHNICAL BRIEF

Secure Enterprise AI Chat

Practitioner reference for deploying network containment for LibreChat on Kubernetes

Aviatrix Distributed Cloud Firewall · Reference Architecture for LibreChat on Kubernetes

Threat Context

LibreChat keeps enterprise data in the customer environment at the application layer but ships with no enforcement layer beneath it, and Kubernetes networking provides none by default. Any configured large language model (LLM) provider can be called immediately, any Model Context Protocol (MCP) server can reach any internet destination, and a compromised open-source dependency can beacon outbound. This architecture installs default-deny network containment below the application, with the policy derived from LibreChat's own configuration so it cannot drift.

LAB-VALIDATED THREAT SCENARIO

A compromised transitive dependency in a LibreChat pod attempts to beacon to an attacker-controlled host. With LibreChat pods scoped by a SmartGroup, approved destinations expressed as WebGroups, and a default-deny base, the Spoke Gateway evaluates the connection, matches no permit rule, blocks it before the socket completes, and logs it in CoPilot, regardless of where the compromise entered.

Insertion Pattern

Enforcement is inserted as a policy overlay at the cluster's network boundary – not inside the application. No new network infrastructure is provisioned.

1. Attach the LibreChat cluster's network to an existing Aviatrix Spoke Gateway using the corresponding Aviatrix Blueprint, so pod egress routes through the gateway at the node subnet level.

2. Run the configuration shim against librechat.yaml in the continuous integration pipeline to generate Aviatrix Kubernetes custom resource definitions (CRDs) – WebGroups, SmartGroups, and DCF rules.

3. Apply the generated overlay via the Terraform Aviatrix provider; review the diff in the same pull request as the configuration change.

4. The Controller programs the Spoke Gateway; DCF evaluates every LibreChat pod's outbound connection against the default-deny posture.

Enforcement is below the application trust boundary. Because policy is a property of the network path, a compromised dependency inside a pod has no process to disable and no alternate path to the internet.

Prerequisites

Before applying the overlay, verify the following are in place:

- Aviatrix Controller and CoPilot deployed and reachable. Controller 8.2 or later for the full policy layer (SmartGroups, WebGroups, Kubernetes identity, DCF rules); Controller 9.0 or later for transparent Transport Layer Security (TLS) decryption and the AgentGuard insertion point.

  An Aviatrix-secured EKS, AKS, or GKE cluster running LibreChat, with an Aviatrix Spoke Gateway and the corresponding Aviatrix Blueprint applied.

- Access to the LibreChat configuration file (librechat.yaml) in the repository the shim reads.

- The continuous integration pipeline able to run the shim and apply Terraform, so the generated overlay is reviewed and merged as code.

- Consistent LibreChat pod labels and namespace conventions so SmartGroups can resolve workload identity.

- For transparent decryption: ability to mount the Aviatrix certificate into the LibreChat container via a ConfigMap (no image rebuild required).

> **TLS DECRYPTION NOTE**
>
> The Aviatrix certificate is loaded into the LibreChat container via a ConfigMap mounted as the Node.js extra-certificates path – no image rebuild. The default-allow-listed LLM providers defer to the system trust store rather than pinning certificates, so they re-negotiate through the gateway chain. Decryption is scoped only to permit rules; deny rules explicitly disable decryption. Certificate rotation is a ConfigMap update and a rolling restart.

Namespace and Group Layout

Policy follows pod identity, so the grouping model is the foundation of the deployment. The shim derives the groups from the configuration; the layout below is what it produces.

Object	Type / scope / purpose
LibreChat pod SmartGroup	Matches LibreChat application pods by label and namespace. Survives pod restarts, scaling, and MCP sidecar churn.
LLM provider WebGroup	Approved LLM provider destinations, one entry per provider declared in the configuration (for example, the providers the deployment is configured to use).

Per-MCP allow-list WebGroup	One allow-list per MCP server, scoped to exactly the downstream interfaces that server is authorized to reach. When Obot is the MCP platform, the shim emits a single internal entry for the Obot endpoint instead of per-server external entries.
Registry WebGroup	The exact Open Container Initiative (OCI) registries the LibreChat chart pulls from – gated by a cloud_provider parameter – and nothing else.
Data-store isolation rule	Denies internet egress from the conversation data store; permits only required in-cluster paths.
Default-deny base	Any destination not matched by a permit WebGroup is denied and logged.

Policy Evaluation Tiers

DCF evaluates LibreChat egress in tiers, from coarse default-deny to optional URL-path visibility. Apply progressively – baseline in monitor mode, then enforce.

Tier	What it evaluates / status
Tier 1 – Default-deny	Any destination not matched by a permit is denied and logged. Removes every unguarded outbound path, including supply-chain beacons. (Today.)
Tier 2 – Domain (SNI) filtering	WebGroups permit destinations by fully qualified domain name using Server Name Indication (SNI) classification. Enforces all containment goals without decryption. (Controller 8.2+.)
Tier 3 – Transparent TLS decryption	The gateway decrypts permitted LLM traffic inline for Uniform Resource Locator-path visibility in CoPilot, scoped only to permit rules. (Controller 9.0+.)
Tier 4 – AgentGuard inspection	Prompt-injection detection, output classification, and tool-argument validation at the decryption insertion point, enabled by configuration as the capabilities ship. (Roadmap; no architectural change.)

SNI versus decryption. On Controller 8.2, set decryption off and Tier 2 still fully enforces every containment goal; you defer URL-path visibility and the AgentGuard insertion point until you upgrade to 9.0.

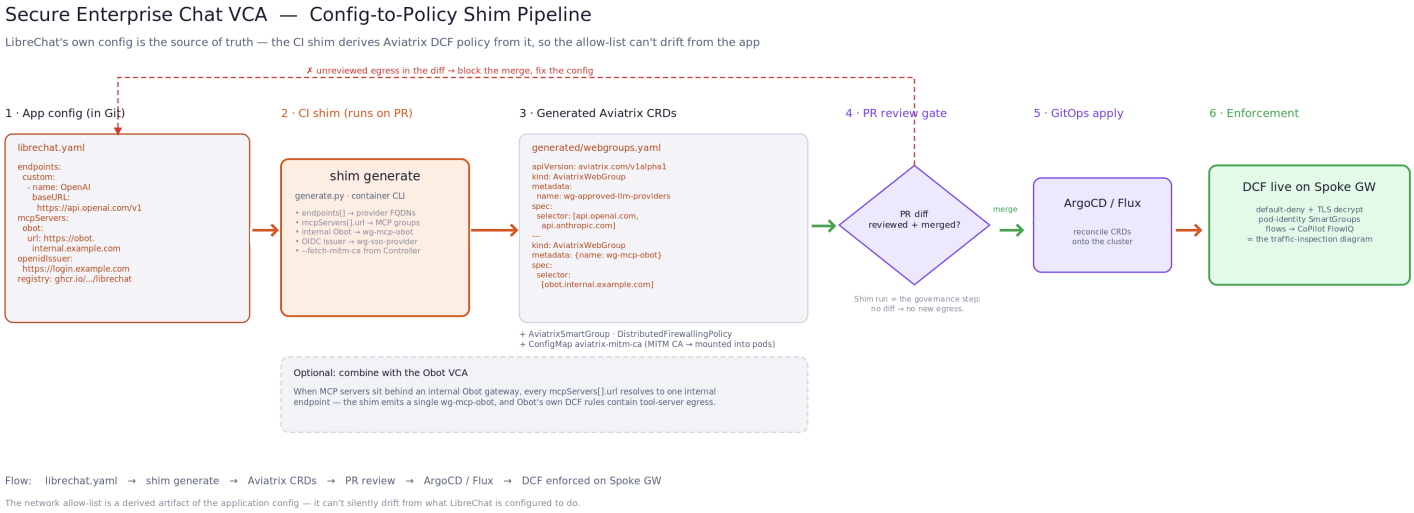
How the Controller Drives Enforcement

The Aviatrix Controller is the policy plane; the Spoke Gateway is the enforcement point. The shim derives DCF objects from the configuration, which the Controller programs onto the gateway without touching application code.

Configuration-to-CRD flow

1. The shim reads librechat.yaml and resolves declared LLM providers and MCP servers.
2. It emits Aviatrix Kubernetes CRDs: the LibreChat SmartGroup, provider and per-MCP WebGroups, the registry WebGroup, the data-store isolation rule, and the default-deny base.
3. The overlay is applied via the Terraform Aviatrix provider and reviewed in the same pull request as the config change.
4. The Controller reconciles the CRDs and programs the Spoke Gateway; enforcement takes effect on pod egress at the node subnet boundary.

Drift resistance. Adding a provider to the configuration produces a new WebGroup entry in the diff; removing one removes the rule. An administrator cannot merge an unauthorized provider without surfacing an unreviewed egress path, and the network blocks the call regardless.



Known Constraints

Constraint	Workaround / Notes
Transparent decryption requires providers that defer to the system trust store; a client that pins certificates would not re-negotiate.	The default-allow-listed providers do not pin in their client toolkits. For a provider that pins, run that destination in SNI mode (Tier 2) without decryption.

URL-path enforcement (permit one path, deny another on the same domain) is not yet available.	Transparent decryption provides URL-path visibility today; URL-path enforcement is pending Uniform Resource Locator WebGroup general availability. Enforce at the domain level until then.
AgentGuard content inspection is not yet generally available.	Deploy the architecture now to install the decryption insertion point; enable AgentGuard by configuration when the capabilities ship – no architectural change.
Certificate-delivery via init container is not in this version.	The shim delivers the certificate at continuous integration time via ConfigMap (GitOps-idiomatic, avoids privilege escalation at pod startup). Init-container delivery is a planned follow-on.
Bare wildcard destinations are not accepted as a permit target.	Use exact hostnames or leading wildcards (for example, *.example.com). A bare * is rejected by the Controller.

Conclusion

The overlay installs default-deny network containment for LibreChat on Kubernetes in under twenty minutes, with the policy derived from the application's own configuration and kept honest by the continuous integration shim. SNI-based filtering enforces every containment goal on the current general-availability Controller; transparent TLS decryption adds URL-path visibility and the AgentGuard insertion point on the latest release. The same DCF fabric governs production workloads, AI agents, and MCP servers – one policy model, one audit log, one control plane.

Appendix — Cloud-Specific Domain

The policy is identical across clouds; the only cloud-specific parameter is the container registry allow-list, gated by the cloud_provider variable. Permit only the destinations the deployment requires. Express destinations by fully qualified domain name; use leading wildcards where a provider rotates subdomains. The lists below are representative starting points – baseline against monitor-mode data before enforcing.

Representative LLM provider destinations (from configuration)

Provider / platform	Representative permitted domains
OpenAI	api.openai.com
Anthropic	api.anthropic.com
Azure OpenAI	*.openai.azure.com scoped to the resources in use
Google	*.googleapis.com scoped to the services in use
Amazon Bedrock	bedrock*.<region>.amazonaws.com scoped to the regions in use

Container registry allow-list (gated by cloud_provider)

Cloud	Representative registries
Amazon Web Services	Amazon Elastic Container Registry endpoints for the account and region, plus the public registries the LibreChat chart pulls from
Microsoft Azure	Azure Container Registry endpoints for the registry in use, plus the chart's public registries
Google Cloud Platform	Google Artifact Registry endpoints for the project and region, plus the chart's public registries

Domain syntax: exact hostnames or leading wildcards (*.example.com). A bare * is rejected by the controller. The shim derives the provider and MCP destinations from the configuration; the registry list is supplied per cloud via the cloud_provider parameter.

Aviatrix Validated Containment Architecture for Librechat for Kubernetes removes the unknown from agentic deployment by enforcing policy at the network layer. **Ask your Aviatrix account team** for a guided deployment.

Explore Validated Containment Architectures for other AI platforms.