AVIATRIX

TECHNICAL BRIEF

Contain Enterprise MCP Infrastructure

Powered by Aviatrix + Obot

Aviatrix Distributed Cloud Firewall · Reference Architecture for AKS / EKS

Aviatrix Validated Containment Architectures are lab-tested containment deployment blueprints for the AI platforms enterprises are actually running. They are ship-ready, policy-included, and validated before they arrive. This Validated Containment Architecture covers Enterprise MCP Infrastructure with Obot.

Threat Context

By default, MCP servers operate as ungoverned egress paths. Standard Kubernetes NetworkPolicy, service mesh, and audit logging do not observe MCP server egress by default.

The Lethal Trifecta describes the three conditions that create an exfiltration vector: agents with access to private data, exposure to untrusted content, and the ability to communicate externally. MCP servers stack the first two legs by design. This architecture breaks the third.

> **LAB-VALIDATED THREAT SCENARIO**
>
> An MCP server is permitted to send patient data to an Electronic Health Record (EHR) endpoint. A compromised npm dependency attempts to exfiltrate the same data to an attacker-controlled domain. The server has permission for the action, not the destination. Aviatrix sees the destination, applies the FirewallPolicy Custom Resource Definition (CRD), and blocks the egress. The attempt is logged. The patient data does not leave the VPC/VNet.

Prerequisites

Before configuring DCF enforcement, verify the following are in place:

- Obot v0.21.0+ deployed via Helm into namespace obot-system with OBOT_SERVER_MCPNETWORK_POLICY_PROVIDER_CHART_* env vars configured. Network Parameter Control (NPC) is self-deployed by Obot automatically; do not install separately.
- Aviatrix Spoke Gateway (Policy Enforcement Point) attached to the cluster VPC/VNet and registered with the Aviatrix controller.
- **Amazon Elastic Kubernetes Service (EKS):** validated on Kubernetes 1.32 using AWS_VPC_K8S_CNI_EXTERNALSNAT=true on the vpc-cni addon.
    - The controller auto-discovers EKS clusters via CSP account scan.

- **Azure Kubernetes Service (AKS):** validated on Kubernetes 1.34.4 with cert-based credentials (use_csp_credentials = true). Exec-credential-based clusters have not been validated for K8S_POLICY_LIST enforcement.
  - ip-masq-agent configured with nonMasqueradeCIDRs: 0.0.0.0/0; pod source IPs must reach the gateway unmasqueraded. Without this, Kubernetes label-based CRD enforcement silently fails.
- Aviatrix Controller 8.2.x minimum: required for aviatrix_distributed_firewalling_default_action_rule (POST_RULES deny-all). Provider aviatrixsystems/aviatrix ≥ 8.2.0.

What Ships on Day One

Two Terraform blueprints ship in the public GitHub repo (https://github.com/AviatrixSystems/aviatrix-blueprints):

Blueprint 1: blueprints/obot-mcp-egress-aws

- Provisions: AWS VPC, 3 private subnets (one per AZ), 1 public subnet, Internet Gateway, EKS cluster (Kubernetes 1.32, vpc-cni with EXTERNALSNAT=true), EKS managed node group (default: 2x m5.large), IAM roles, Aviatrix Spoke Gateway (Policy Enforcement Point), Aviatrix SmartGroups (obot-mcp namespace, obot-system namespace, EKS VPC CIDR, API server /32s), Aviatrix WebGroups (EKS infra domains, Obot app domains), DCF V1 permit policy list, POST_RULES default-deny action, CoPilot association, remote syslog (index 9), Kubernetes namespaces (obot-system, obot-mcp), k8s-firewall Helm release (installs FirewallPolicy + WebgroupPolicy CRDs), Obot Helm release (v0.21.0+, includes aviatrix-network-policy-controller)
- Also includes: terraform.tfvars.example, AGENTS.md (machine-readable deploy guide for Claude Code/Codex/Cursor/Gemini CLI), architecture.svg, k8s-apps/example-mcpnetworkpolicy.yaml
- Estimated cost: ~$0.25–0.45/hour (Spoke Gateway + EKS nodes + control plane)

Blueprint 2: blueprints/obot-mcp-egress-azure

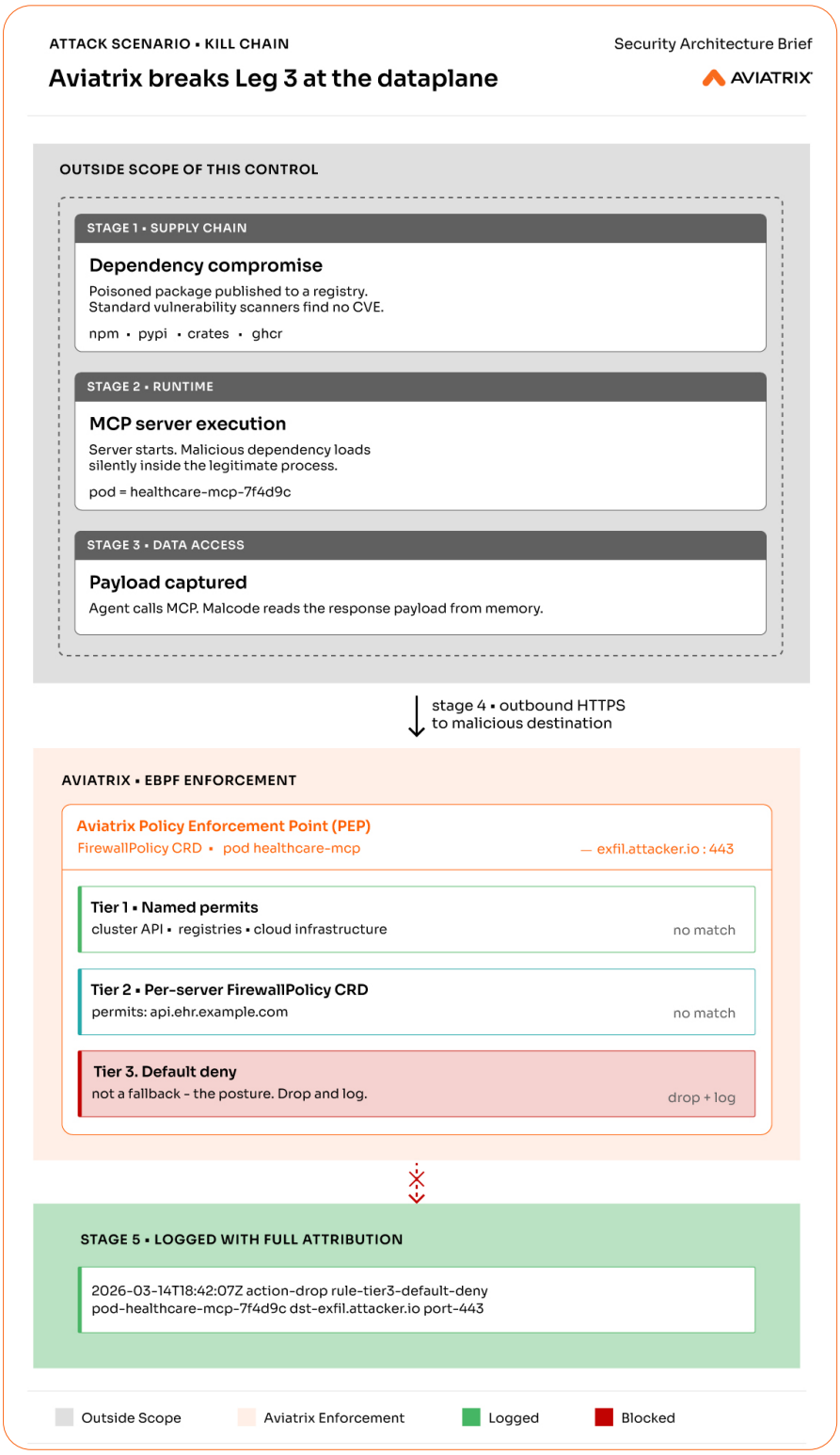
- Provisions: Azure Resource Group, VNet, AKS subnet, Spoke Gateway subnet, two Azure Route Tables, AKS cluster (Azure CNI, cert-based credentials), Azure Role Assignment, Aviatrix Spoke Gateway, Aviatrix SmartGroups (obot-mcp namespace, AKS subnet CIDR, K8s API server public IP, obot-system namespace), Aviatrix WebGroups (AKS infra domains, Obot app domains), DCF V1 permit policy list, POST_RULES default-deny, ip-masq-agent ConfigMap (disables pod SNAT), Kubernetes namespaces, k8s-firewall Helm release, Obot Helm release
- Also includes: terraform.tfvars.example, AGENTS.md, architecture.svg, k8s-apps/example-mcpnetworkpolicy.yaml
- Estimated cost: ~$0.25–0.45/hour (Spoke Gateway VM + AKS nodes)

Both blueprints also include: README with step-by-step deployment guide (init/plan/apply), 3 validated test scenarios (permit passes, block fires, live policy update), cleanup instructions, and troubleshooting reference.

Namespace Layout

Namespace	Purpose
obot-system	Obot orchestration pods (obot-obot, aviatrix-network-policy-controller)
obot-mcp	MCP server pods : one pod per server, spawned by Obot on demand

DCF enforcement targets obot-mcp. Obot's own egress (Anthropic API, GitHub, Helm charts) is permitted via separate rules scoped to obot-system pod IPs.



Policy Evaluation Order

Every packet from every pod traverses three tiers in order:

Tier 1: Named Permit Rules

- CIDR permit for the Kubernetes API server
- Cloud infrastructure egress permit (ECR/ACR, blob storage, container registries) scoped to cluster subnet CIDR
- Obot pod egress permit (Anthropic API, GitHub, charts.obot.ai) scoped to obot-system pod IPs

Tier 2: Per-Pod FirewallPolicy CRDs

Each MCP server carries its own FirewallPolicy in obot-mcp specifying exactly which domains it may reach. If no CRD exists for a pod, traffic falls through to Tier 3.

Tier 3: Default Deny

POST_RULES default action: DENY + LOG. Catches everything not explicitly permitted in Tier 1 or Tier 2. Must be configured as a default action rule, not as a named policy entry.

> **WHY THE KUBERNETES API SERVER NEEDS A CIDR PERMIT**
>
> Pods access the API server via ClusterIP → kube-proxy DNAT → real API server IP. TLS over an IP has no SNI, so domain-based matching cannot fire. The CIDR SmartGroup permit must sit in Tier 1 at priority 1.

REFERENCE ARCHITECTURE CONCEPTUAL

Two-layer enforcement

Solution Brief

AVIATRIX + obot

LAYER 1 ▪ OBOT ▪ GOVERNANCE

Which MCP servers run

Each MCP server's manifest declares the domains it may reach. Empty list = zero egress.

namespace: obot-mcp

mcp-server
healthcare-mcp
permits: api.ehr.example.com

mcp-server
code-mcp
permits: api.github.com

mcp-server
internal-mcp
permits: (none) · zero egress

every packet from every pod

LAYER 2 ▪ AVIATRIX ▪ ENFORCEMENT

Where they can reach

The Aviatrix gateway evaluates each server's permit list inline. Three tiers nothing falls through.

Aviatrix Policy Enforcement Point (PEP)
eBPF dataplane inline policy evaluation

Tier 1 ▪ Named permits
Cluster API ▪ container registries ▪ cloud infrastructure — allow

Tier 2 ▪ Per-server permits
Match by pod label ▪ allow declared domains only — deny/allow

Tier 3. Default deny
The posture, not a fallback. — deny + log

EXTERNAL DESTINATIONS

Cluster infrastructure
K8s API ▪ registries

Permitted domains
api.ehr.example.com
api.github.com

Cluster infrastructure
K8s API ▪ registries

Obot governance · Aviatrix enforcement · Permitted · Blocked + logged

SmartGroup and WebGroup Design

Object	Type / Scope / Purpose
obot-mcp	SmartGroup: K8s namespace selector, target for CRD-based rules
obot-system-pod-ips	SmartGroup: CIDR /32 list of obot-system pod IPs, source for Obot orchestration egress permits
<cloud>-subnet	SmartGroup: CIDR, cluster node+pod CIDR, source for infra egress permits
<cloud>-api-server	SmartGroup: CIDR /32: K8s API server resolved IPs, destination for no-SNI CIDR permit
<cloud>-infra-egress	WebGroup: SNI filter, cloud infra domains, permits node bootstrap, image pull
obot-pod-egress	WebGroup: SNI filter: Obot app domains, permits Anthropic, GitHub, charts.obot.ai

CRD-defined WebGroups (per MCP server) are separate from the above and live in the cluster, not on the controller.

How Obot Drives FirewallPolicy CRDs

The aviatrix-network-policy-controller (running in obot-system) watches Obot's MCP server objects and reconciles a FirewallPolicy CRD for each in obot-mcp. Platform engineers do not write CRDs directly.

Set egressDomains in the MCP server manifest via the Obot API:

```
{ "name": "my-mcp",
  "npxConfig": {
    "package": "@my-org/my-mcp",
    "egressDomains": ["api.example.com", "*.storage.example.com"]
  }
}
```

egressDomains / defaultEgressPolicy	Result
obot-system	FirewallPolicy permits listed domains only
Empty list + deny	FirewallPolicy denies all egress
Empty list + allow	No enforcement : all egress permitted

Domain syntax: exact hostnames or leading wildcards (*.example.com). Bare * is rejected by the controller.

Known Constraints

Constraint	Workaround
CoPilot association resource accepts private IP only	Call controller API directly with both private and public IPs. Public IP required for DCF Monitor when VNets are not peered.
obot-system pod IPs change on restart	CRD enforcement continues unaffected. Only Obot-egress permits in Tier 1 break until IPs are refreshed. MCP server isolation is maintained.

Appendix: Cloud-Specific Domain Requirements

AKS (Azure): aks-infra-egress WebGroup

- *.hcp.<region>.azmk8s.io : AKS API server
- *.azurecr.io : Azure Container Registry
- *.blob.core.windows.net / *.servicebus.windows.net
- mcr.microsoft.com / *.data.mcr.microsoft.com
- management.azure.com / login.microsoftonline.com
- packages.microsoft.com / acs-mirror.azureedge.net
- *.aks.azure.com : AKS node management plane
- ghcr.io / *.ghcr.io / pkg-containers.githubusercontent.com

*K8s API server SmartGroup: resolve *.hcp.<region>.azmk8s.io to /32 IPs at deploy time. Regenerate after cluster upgrades.*

EKS (AWS): eks-infra-egress WebGroup

- *.eks.amazonaws.com
- *.dkr.ecr.<region>.amazonaws.com / *.ecr.aws
- *.s3.amazonaws.com / s3.amazonaws.com
- ec2.<region>.amazonaws.com / ssm.<region>.amazonaws.com / sts.amazonaws.com
- ghcr.io / *.ghcr.io / pkg-containers.githubusercontent.com

K8s API server SmartGroup: DNS lookup against private cluster endpoint FQDN. Terraform runner must be inside the VPC or on a peered network.

Both Clouds: obot-pod-egress WebGroup

- charts.obot.ai
- github.com / *.github.com
- raw.githubusercontent.com / *.githubusercontent.com

Aviatrix Validated Containment Architecture for Enterprise MCP Infrastructure with Obot 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.