AVIATRIX

SECURITY ARCHITECTURE BRIEF

Secure Enterprise GitHub Pipelines

Powered by Aviatrix — Self-Hosted Runner Edition

Threat model, enforcement architecture, and compliance evidence for security architecture review

EXECUTIVE SUMMARY

Aviatrix Validated Containment Architectures are lab-tested containment deployment blueprints for the enterprise platforms you are actually running. They are ship-ready, policy-included, and validated before they arrive. This Validated Containment Architecture covers Secure Enterprise GitHub Pipelines for self-hosted runners. When using the native egress path, every self-hosted GitHub Actions runner is ungoverned – there is no egress firewall, no traffic logging, and no destination policy. When a dependency or Action is compromised, it inherits the runner's network permissions and can exfiltrate $GITHUB_TOKEN, cloud credentials, SSH keys, and source code to any destination on the internet. Static IPs and vulnerability scanners cannot stop this. This architecture breaks that attack chain at the network layer. Aviatrix controls what self-hosted GitHub Actions runners can reach. Enforcement is built into the deployment, not bolted on afterward. The result is network-layer egress containment with full audit trail, enforced under a single policy model.

Threat Model

The CI/CD exfiltration trifecta describes the three conditions that together create a critical exfiltration vector in enterprise software pipelines:

CI/CD Trifecta	GitHub Actions Runner Exposure
Leg 1: Secrets and credentials access	Runners hold $GITHUB_TOKEN, cloud credentials (AWS/GCP/Azure), SSH keys, Docker configs, and API secrets. Credentials are present in every runner by design – they are the mechanisms by which pipelines deploy, publish, and communicate with downstream services.
Leg 2: Untrusted code exposure	Runners execute third-party GitHub Actions, npm packages, PyPI packages, Docker images, and RubyGems. Millions of lines of third-party code execute per build. Any of these can be supply-chain-compromised. The tj-actions and TeamPCP attacks exploited exactly this surface.

Leg 3: External communication	When using the native egress path, every self-hosted GitHub Actions runner can reach anything on the internet – there is no egress firewall, no traffic logging, and no destination policy. This is the leg that turns compromise into breach.

WHY SUPPLY CHAIN ATTACKS MAKE LEG 3 THE PRIORITY

The March 2025 tj-actions/changed-files compromise (CVE-2025-30066) affected 23,000+ repositories. The payload dumped Runner Worker process memory – secrets, PATs, cloud credentials – and exfiltrated them to an external server. CISA issued an emergency advisory. In March 2026, the TeamPCP Cascade compromised Trivy, LiteLLM, and 60+ npm packages (CVE-2026-33634, CVSS 9.4). SSH keys, cloud tokens, and Docker configs were harvested across 10,000+ CI/CD workflows and ~1,000 enterprise SaaS environments. In both attacks, a compromised dependency inside a trusted pipeline inherited the runner's egress permissions automatically. Trust-based controls don't catch this. Domain-based enforcement at the network layer does, because it doesn't care whether the code running is legitimate or poisoned. It only cares where the traffic is going.

Aviatrix governs what self-hosted runners can reach.

Attack Scenario and Kill Chain

A self-hosted runner executes npm install, which fetches a supply-chain-compromised package. The package payload reads $GITHUB_TOKEN and cloud credentials from environment variables, then attempts to exfiltrate them to an attacker-controlled endpoint.

Kill Chain Stage	What Happens	Control
Dependency compromise	Poisoned package published to registry. Passes standard vulnerability scanning: no known CVE at time of install.	Outside scope of this control
Package install	npm install fetches from registry.npmjs.org. Package executes install script. Malicious code loads into the runner process.	Outside scope of this control
Credential harvest	Malicious code reads $GITHUB_TOKEN, AWS_ACCESS_KEY_ID, and other environment variables present in the runner. Constructs exfiltration payload.	Outside scope of this control

Exfiltration attempt	Malicious code initiates outbound HTTPS to evil.attacker.io on port 443.	Aviatrix blocks destination not in permit list
Audit trail	Blocked connection logged at the Policy Enforcement Point (Aviatrix Spoke Gateway) with runner identity, destination domain, timestamp, and policy rule.	CoPilot audit trail, attributable to specific runner or repository

POINT OF INTERVENTION

The runner has permission for the credential, not the destination. The Aviatrix DCF WebGroup for this runner permits only registry.npmjs.org, github.com, and infrastructure endpoints. Aviatrix evaluates the destination at the dataplane Spoke Gateway level and drops the connection before it leaves the VPC. Credentials have nowhere to go. The attempt is logged with full attribution. The stolen credential is useless.

Going deeper: TLS inspection and agent activity analysis. Domain-based blocking stops most exfiltration scenarios. For environments requiring deeper assurance, Aviatrix can terminate TLS at the Spoke Gateway to inspect payload content on permitted connections – enabling detection and blocking of data exfiltration attempts even when the destination domain is allowed. This provides visibility into agent activity at the request level, not just the connection level, and allows policy to act on what is being sent, not only where it is going.

Going deeper: URL-level filtering. WebGroup policies can be scoped beyond the domain to target specific URL paths. Rather than permitting all of registry.npmjs.org, policy can restrict to specific packages, namespaces, or paths – reducing the attack surface to exactly what each runner group legitimately needs. This is enforced at the same dataplane enforcement point with no additional infrastructure required.

Enforcement Architecture

Aviatrix enforces the network path. Enforcement runs at the network infrastructure layer, and requires no code changes, agent installation on runners, or application team involvement.

Layer 1 – Spoke Gateway: Network-Level Egress Control

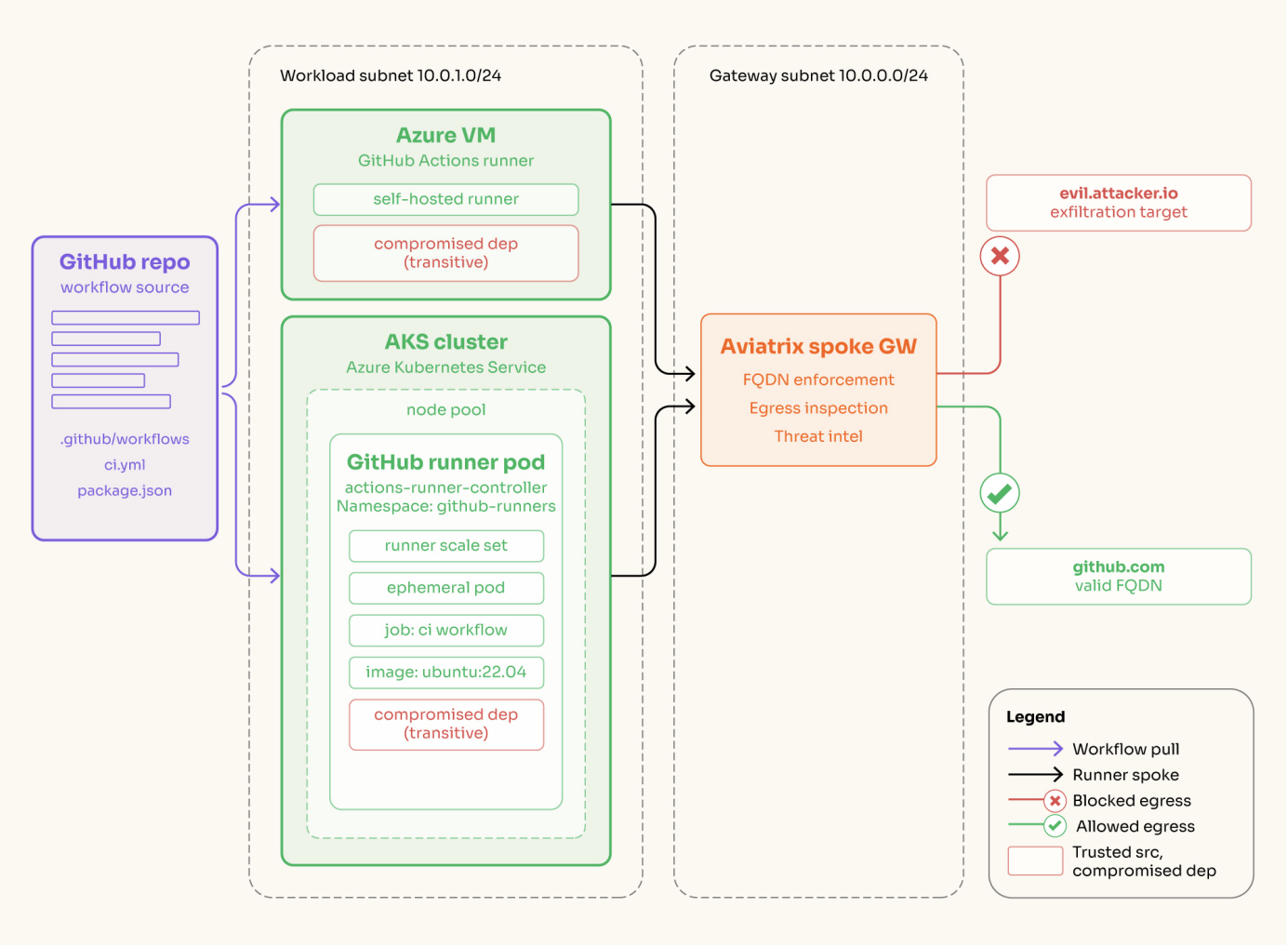
Traffic from every self-hosted runner passes through the Aviatrix Spoke Gateway before reaching the internet. This is enforced at the routing layer – the route table directs 0.0.0.0/0 to the Spoke Gateway network interface, and the runner has no public IP.

- Every outbound connection logged with source identity, destination IP, port, and timestamp
- Known egress IP – internal firewalls can allowlist runner traffic by source address
- Zero bypass risk – no process on the runner can circumvent the route table

Layer 2 – Distributed Cloud Firewall: FQDN-Based Egress Policy

Traffic passing through the Spoke Gateway is evaluated against FQDN-based WebGroup policies using Aviatrix DCF L7 filtering. Three-tier evaluation ensures nothing falls through:

Tier	Role
Tier 1: Named permits	Explicit permits for cloud infrastructure and container registries. Evaluated first.
Tier 2: Per-runner-group rules	SmartGroups match runner VMs by subnet or tag. WebGroups define approved egress by FQDN per runner profile. Python runners allow pypi.org; Node runners allow registry.npmjs.org; Docker runners allow docker.io. Traffic not matching falls to Tier 3.
Tier 3: Default deny	Catches everything not explicitly permitted in Tier 1 or Tier 2. This is not a fallback; it is the posture.



WHY STATIC IPS DON'T SOLVE THIS PROBLEM

Static IPs control where traffic comes from. They do nothing to control where traffic goes. A self-hosted runner with a static IP can exfiltrate to evil.attacker.io:443 just as easily as one without. Static IPs allow the enterprise to set ingress allowlists on internal firewalls; they provide zero egress destination control. Aviatrix Secure CI/CD Pipelines controls the destination – what the runner can reach – enforced and logged at the network layer.

Architectural Boundaries

This control governs where self-hosted GitHub Actions runners can send traffic. It is one layer in a defense-in-depth posture, not a complete CI/CD security solution.

Out of Scope	What Governs It Instead
Pre-execution code analysis	SCA/SAST tools (Snyk, Socket, Dependabot) scan dependencies before code reaches the runner. Aviatrix is complementary – it governs runtime network access regardless of whether code was flagged.
Action hardening	Replacing untrusted third-party Actions with audited versions is the function of Chainguard Actions. This reduces the probability of malicious execution but does not govern what any Action can reach on the network.
Content inspection	Domain-based, not content-based. Aviatrix controls whether a connection to a destination is permitted. What is sent over a permitted connection requires a separate control.
East-west pipeline traffic	This control governs external egress at the Spoke Gateway. Internal VPC traffic between pipeline services requires separate DCF policy scoped to internal SmartGroups.

Compliance Evidence

For SOC 2, HIPAA, PCI-DSS, FedRAMP, and NIST SSDF environments, auditors require architectural proof of enforcement – not a policy document.

Evidence Artifact	What It Proves
DCF WebGroup/ SmartGroup policy in git	Security policy in the same repository as deployment manifests. Same PR. Same pipeline. Every change tracked, attributed, and reversible.

CoPilot-inspectable policy	CoPilot shows exactly which domains each runner group may reach at the moment of inspection, alongside every connection log. No interpretation required.
PEP connection logs	Every allowed and denied connection logged with source identity, destination FQDN, policy rule, and timestamp. Exportable to SIEM. FlowIQ provides per-repo traffic analysis.

The proof of enforcement is the architecture itself – not a statement about the architecture.

Deployment Paths

Self-hosted runners in VPC/VNET

Deploy an Aviatrix Spoke Gateway into the VPC or VNET where self-hosted runners reside. As part of deployment, the Spoke Gateway automatically updates the subnet route table to point default egress traffic (0.0.0.0/0) to its network interface – runners immediately have no public IP and all egress flows through the gateway without any manual routing configuration. Add DCF WebGroups to introduce FQDN-based default-deny policy per runner group. The Terraform reference module ships with the Validated Containment Architecture package.

Next Steps

The Validated Containment Architecture for Secure Enterprise GitHub Pipelines – Self-Hosted Runner Edition is available today. The Terraform reference module and deployment guide ship together.

Request a 30-minute architecture review. We walk through the enforcement model in your environment, inventory your self-hosted runner deployment, and identify the egress paths currently without visibility.

Request a 30-minute architecture review.

Walk through the enforcement model in your environment, map your current AgentCore Runtime inventory, and identify the egress paths you currently cannot see.

About Aviatrix

Aviatrix® is pioneering the Cloud Native Security Fabric – the architecture the Containment Era requires. The Cloud Native Security Fabric governs every workload communication path across every cloud, every VPC, every Kubernetes cluster, and every serverless function, from a single policy plane. One rule. Universal propagation. Enforced at the workload, not at a chokepoint. Trusted by more than 500 of the world's leading enterprises. For more information, visit aviatrix.com.