AVIATRIX

SOLUTION BRIEF

Secure Enterprise GitHub Pipelines

Powered by Aviatrix — Self-Hosted Runner Edition

Aviatrix Validated Containment Architectures are lab-tested containment deployment blueprints for the enterprise platforms you are actually running. Ship-ready, policy-included, validated before they arrive. This Validated Containment Architecture covers Secure Enterprise GitHub Pipelines for self-hosted runners.

The Threat

When using the native egress path, GitHub Actions runners are unrestricted – existing security controls were never designed to govern them. Every self-hosted runner – whether deployed in an AWS VPC or Azure VNET – has unrestricted outbound internet access when using that native path. There is no egress firewall, no traffic logging, and no destination policy.

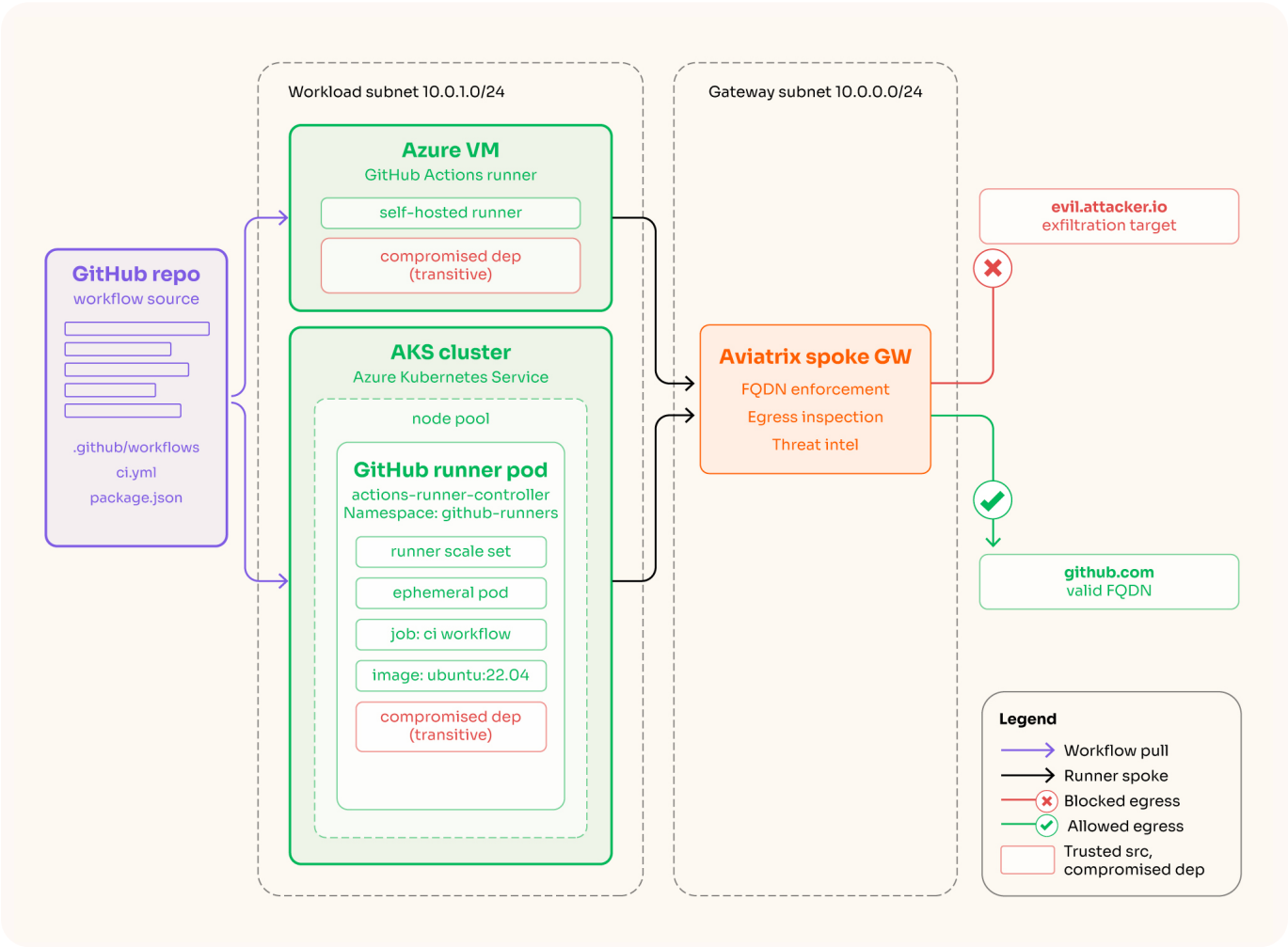
GitHub Actions runners stack three properties that together create a critical exfiltration vector: access to secrets and credentials ($GITHUB_TOKEN, cloud keys, SSH keys), exposure to third-party code from public registries and Actions (transitive dependencies are a good example), and unrestricted outbound network reach. Constraining the third at the network layer materially reduces the blast radius of the other two. The March 2025 tj-actions compromise (23,000+ repositories) and the March 2026 TeamPCP Cascade (10,000+ CI/CD workflows, ~1,000 enterprise SaaS environments) illustrate how a compromised dependency inherits a runner's egress permissions and exfiltrates credentials to attacker-controlled endpoints. Domain-based enforcement is well-suited to this case because it evaluates the destination rather than the caller's identity – blocking exfiltration regardless of which process initiated it.

The Architecture

Aviatrix governs what self-hosted GitHub Actions runners can reach. Private subnet routing through an Aviatrix Spoke Gateway provides network-layer enforcement – with zero bypass risk, as enforcement occurs outside the runner trust boundary.

How it works in practice

Runner Type	Enforcement Method	Bypass Risk	Status
Self-Hosted Runners (VNET, VPC ...)	Private subnet routing through Spoke Gateway	Zero (network-layer; no process to kill)	Architecture Validated



Three Things Your Current Stack Can't Do

01 FQDN-based default-deny egress per runner

Allow pypi.org, registry.npmjs.org, and github.com while blocking everything else – not by ephemeral IP, but by domain name. A compromised package trying to reach evil.example.com:443 is blocked at the network layer before a single byte leaves your environment.

02 Per-repo network identity (Phase 2)

Each repository gets its own Aviatrix VPN identity, visible in logs (ci-repo-payments-api → evil.com:443 DENIED), controllable via policy, revocable in seconds. Blast radius shrinks from “anything any repo needs” to “only what this specific repo needs.

03 Enterprise security fabric integration

CI/CD egress policy lives alongside production workload policy, AI agent policy, and MCP server policy in a single Aviatrix DCF plane. CoPilot provides unified visibility across CI pipelines, AI agents, and production workloads from one console. No standalone CI security tool provides this.

Compliance Evidence

For SOC 2, HIPAA, PCI-DSS, FedRAMP, and NIST SSDF environments, post-incident reviews now expect architectural evidence of network egress enforcement for CI/CD systems – not just written policy.

DCF WebGroup and SmartGroup configurations are Terraform-managed infrastructure-as-code. Policy ships in the same repository as deployment manifests, reviewed in the same PR, deployed in the same pipeline. CoPilot logs every allowed and denied connection with source identity, destination FQDN, and timestamp – continuous audit evidence, not snapshots. FlowIQ provides per-repo traffic analysis.

Why It Applies to You

- How many self-hosted GitHub Actions runners are active in your environment today, and do you know what every one of them can reach on the internet?
- If a compromised npm package or GitHub Action in your pipeline attempted to exfiltrate $GITHUB_TOKEN to an external domain, what in your current stack would block it?
- Can you distinguish in your logs which specific repository a CI/CD outbound connection originated from – or do all repos share one identity?
- How are you currently proving to auditors that CI/CD workload egress is controlled, not just described in policy?

What's Included

Component	Description
AWS Self-Hosted Runner	Private subnet routing through Aviatrix Spoke Gateway on AWS.
Azure Self-Hosted Runner	Private subnet routing through Aviatrix Spoke Gateway on Azure.
DCF Egress Policy Templates	Pre-built WebGroups for Python (PyPI + GitHub), Node (npm + GitHub), Docker (registries + GitHub), and minimal (GitHub only) egress profiles. Default-deny base policy included.
CoPilot Dashboard	CI pipeline traffic visibility with per-repo FlowIQ, anomaly detection, and every allowed/denied connection logged with source identity and destination FQDN.

Get Started

Request a 30-minute architecture review

We walk through the egress control model in your environment, inventory the self-hosted runner types currently deployed, and surface the network paths worth bringing under explicit policy.

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.ai.