INDIGO CONSULTING

Shifting IAM Left - A CI/CD-Centric Blueprint for Governing AI Agents and Non-Human Identity

Author: Wyatt Bourdeau
Prepared By: Indigo Consulting

indigoconsulting.ca

Table Of Contents

Introduction

The emergence of autonomous AI agents is transforming the workforce, operating as dynamic actors that can execute complex business tasks at machine speed, essentially becoming new "employees".[9,10].

Even continuous integration & continuous delivery (CI/CD) models that refresh secrets every few hours or days create unacceptable windows of exposure[9], as a compromised AI agent could have free rein with credentials until they expire. The risk extends beyond external threats; internal issues like code flaws combined with automation speed can lead to catastrophic outcomes, such as an ephemeral agent issuing erroneous credits or a persistent agent inadvertently terminating a critical production database.

This rise of Non-Human Identity (NHI) presents a significant challenge: empowering this AI workforce while effectively managing the inherent risks - Traditional security models, reliant on static credentials and perimeter-based trust, are ill-suited for this dynamic, non-human world[3].

These scenarios highlight that the agent isn't rogue, but rather efficiently executing a flawed directive - a single flawed directive that can trigger catastrophic financial and operational outcomes, from issuing erroneous transactions to corrupting critical production data. This reality necessitates a secure CI/CD framework capable of matching the velocity and adaptability of AI.

I. From Managing Secrets to Governing Identity

The solution requires a paradigm shift- proactively "shifting left". Ushering critical devOps processes such as testing, security, and QA further upstream in the development lifecycle allows for Identity and Access Management (IAM) to leverage the CI/CD pipeline as a substrate to embed security for NHI and AI agents from the outset.

Pillar 1: Trust Through Verifiable, Accountable Delegation

A critical principle is that an autonomous agent must never operate on its own authority; every action must be explicitly and verifiably tied back to the human user who delegated the task, creating an unbreakable chain of command. This establishes a non-negotiable foundation for auditability.

This is achieved through modern identity standards:

- **Authentication with OpenID Connect (OIDC)**: Verifies the human user's identity.
- **Delegation with OAuth 2.0 Token Exchange** [1]: Formalizes the "on-behalf-of" flow, where an agent presents the user's verified identity token to receive a temporary, narrowly scoped access token.
- **Human-in-the-Loop with CIBA** [4]: For sensitive operations, Client-Initiated Backchannel Authentication (CIBA) provides a circuit-breaker, requiring the agent to trigger a real-time approval request to the delegating user's device before proceeding.
- **Securing the Request with Pushed Authorization Requests (PAR):** To protect the integrity of the initial delegation, PAR moves complex authorization parameters away from the browser URL to a direct, secure back-channel request. This prevents request tampering and the leakage of sensitive data in browser logs. By ensuring the user approves the exact parameters sent by the agent, PAR strengthens the foundation of the verifiable delegation chain.[18]

Pillar 2: Shifting Governance Left for AI Agent & NHI Security

Securing an AI agent cannot be an afterthought; a reactive posture is destined to fail. The strategic imperative is to shift IAM left, embedding dynamic and ephemeral identity controls directly into the DevOps lifecycle where every Non-Human Identity is created[9]. Before the pipeline can act as a trusted registrar, it must be securely granted limited-scope ability to register new agents via a one-time MFA delegation (ideally from a human operator).

Key tools and practices to build secure, version-controlled artifacts within the pipeline include:

- **Policy-As-Code:** The rules for who can do what are decoupled from application code, in a standardized, version-controlled policy format with Open Policy Agent(OPA)[8], enabling agility and clear auditing when access definitions change.
- **Verifiable Identity**: The pipeline issues a cryptographic SPIFFE [6] Verifiable Identity Document (SVID) [12] to each agent, acting as a universal machine-passport
- **Single-Use Tokens**: For critical actions, SPARK (Single-Purpose Authentication & Revocation Keys)[14] provides a unique token invalidated immediately after a single use, drastically reducing the blast radius of a compromise.
- **Just-in-Time Credentials**: The agent uses its SVID to request ephemeral, task-scoped credentials from a vault (e.g., HashiCorp Vault [13]), with lifespans measured in seconds.

Just-in-Time (JIT) Security for AI Agents
Automate Enforcement: Integrate security directly into DevOps processes.
Use Temporary Credentials: Limit the scope and duration of access for credentials.
Grant Access on Trigger: Ensure access is provided only when needed.
Revoke Access After Use: Prevent unauthorized access by expiring credentials.
Log Access Events: Record events for auditing and accountability.

Pillar 3. Dual-Identity Imperative: Why OAuth, Why SPIFFE, Why Both?

Governing AI agents necessitates a multi-layered identity approach. While both OAuth 2.0 and SPIFFE are critical, they serve distinct, yet complementary, functions in securing non-human identities.

Why OAuth?

OAuth 2.0 is foundational for **authorization and delegated access**. It answers "Who is the user?" and, crucially for AI agents, "What can this software do on behalf of a user?". It provides the mechanism for an agent to acquire temporary, narrowly scoped access tokens to interact with applications and APIs , often formalizing the "on-behalf-of" flow where a human user delegates a task to an agent. This ensures verifiable, accountable delegation of authority.

- **OAuth Alone:** OAuth can certainly provide client authentication (e.g., using client secrets or JWT assertions) and then issue access tokens to an agent for resource access. For simple cases where an agent's identity is managed solely within an IAM platform and interactions are primarily with OAuth-protected APIs, it could technically function. *However, this approach often relies on managing client secrets, which, even if rotated, create windows of exposure. It also doesn't inherently provide strong, cryptographically verifiable workload identity for peer-to-peer or service-mesh communications, leaving a gap in foundational machine trust.*

Why SPIFFE?

SPIFFE provides **strong workload authentication and foundational machine identity**. It answers "What is this software?", giving each AI agent instance a cryptographically verifiable, platform-agnostic identity (SVID). This allows services to mutually authenticate via mTLS, establishing trust at the network layer and **eliminating reliance on vulnerable, long-lived secrets.**

- **SPIFFE Alone:** An agent using only SPIFFE could securely authenticate its own software identity to another service, proving *what* it is and establishing a secure communication channel via mTLS [18]. This approach is highly advantageous for creating foundational machine trust, as it gives each agent a short-lived, automatically-rotated, and cryptographically verifiable identity (SVID) without relying on vulnerable static secrets [18]. However, this model operates exclusively at the machine-identity layer and completely lacks the concept of delegated user authority [2].

The agent can prove what it is [1], but cannot convey on whose behalf it is acting or what specific permissions it has been granted for a user-initiated task [2]. This leaves a critical gap in the verifiable chain of command and user-level accountability essential for governing agentic systems.

Why Both? The Power of Layered Trust

Integrating both OAuth and SPIFFE creates a robust, multi-layered security posture:

- **Immutable Identity + Dynamic Authorization:** SPIFFE establishes the agent's fundamental, verifiable machine identity. Once this trust is established, OAuth layers on dynamic, just-in-time authorization, enabling the agent to acquire precisely the permissions it needs, precisely when it needs them. This minimizes the blast radius of any potential compromise. The CI/CD pipeline first registers the workload with SPIFFE, then uses that new SPIFFE identity to register it as an OAuth 2.0 client.
- **End-to-End Traceability:** SPIFFE provides auditable records of workload identity, while OAuth's token exchange and claims allow for detailed logging of delegated actions. This creates an unbroken "chain of command" from human intent to agent action, crucial for auditability and accountability.
- **Reduced Attack Surface:** By leveraging SPIFFE's short-lived, automatically rotated SVIDs and OAuth's scoped, ephemeral tokens , the reliance on vulnerable static credentials is drastically reduced.

In essence, SPIFFE validates *who* the agent instance is, while OAuth dictates *what* that agent can do and *on whose behalf*. This combined approach is critical for building a secure, auditable, and resilient AI agent workforce.

II. The CI/CD Pipeline's Role: Automating Security[15]

The CI/CD pipeline has the capacity to join identity, policy, and security together as deployable control points.

- **Development**: Policies are written as code.
- **Test**: Policies and agent interactions are automatically tested.
- **Release**: Sensitive changes require human approval via auditable controls (e.g., CIBA).
- **Deploy**: The pipeline securely registers the agent.

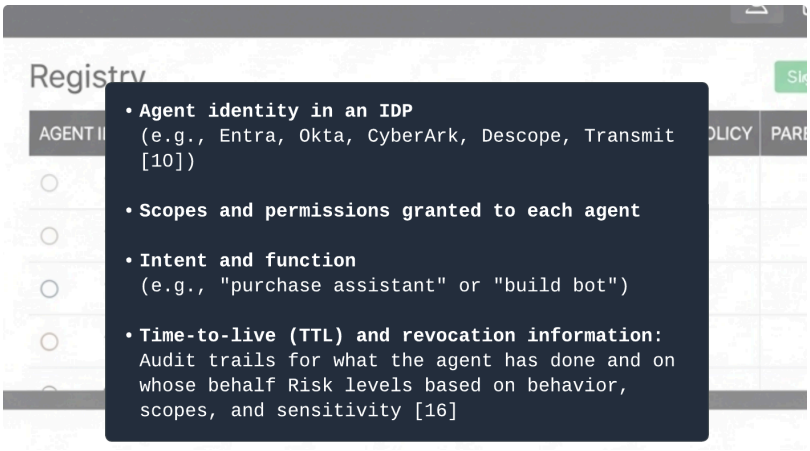
Automating dynamic and scalable security for NHI and agentic models consists of 3 components which address critical functions in CICD:

1. Secure pattern for registration and sequenced registration:

- The CI/CD pipeline first registers the workload with SPIFFE to establish its foundational machine identity.
- Then, it uses that new SPIFFE identity to make a secure call to register the workload as an OAuth 2.0 client[2].

Agent Registry Key Attributes:

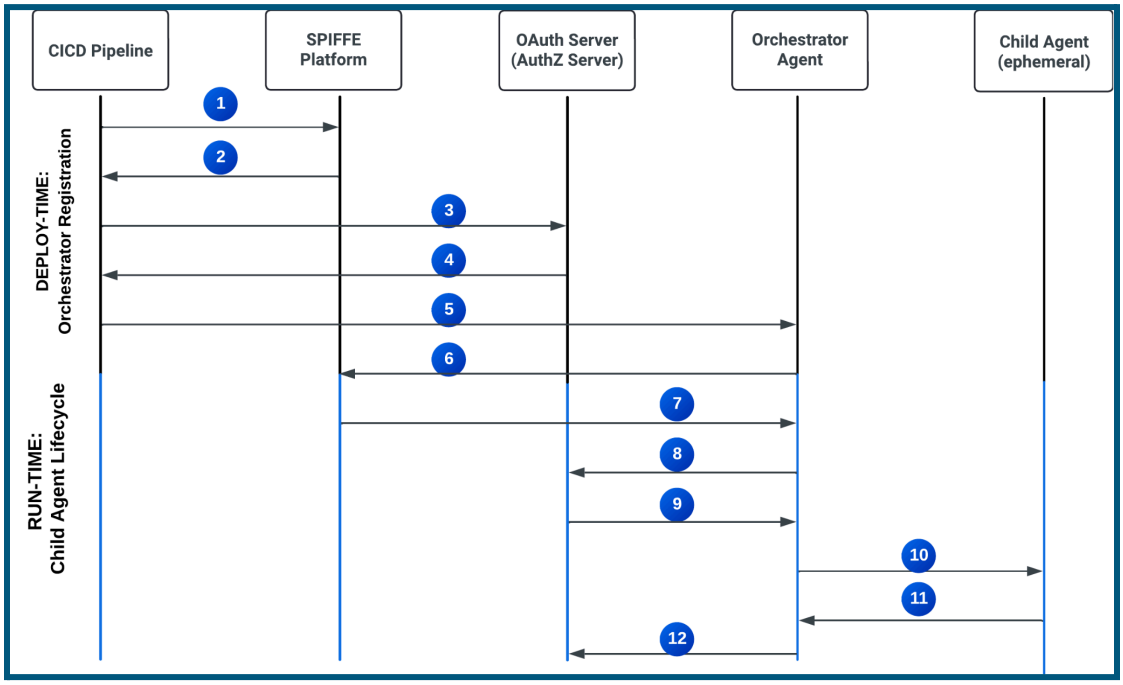
- Agent identity in an IDP (e.g., Entra, Okta, CyberArk, Descope, Transmit [10])
- Scopes and permissions granted to each agent
- Intent and function (e.g., "purchase assistant" or "build bot")
- Time-to-live (TTL) and revocation information
- Audit trails for what the agent has done and on whose behalf Risk levels based on behavior, scopes, and sensitivity [16]



2. Designing for Agent Types:

A. Standing Agent with Children (The "Factory" Model) [11]

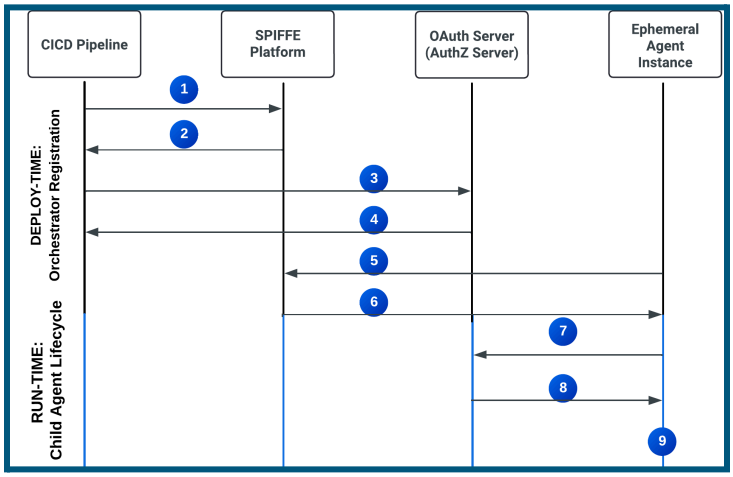
- **CI/CD Role**: Deploys the standing agent and its container blueprint, granting it the authority to register its own children.
- **Registration**: The standing agent performs Just-in-Time (JIT) registration for each child it spins up at runtime.
- **Deployment of Children**: The standing agent programmatically spins up children by calling the underlying compute platform's API (e.g., Kubernetes API, AWS Fargate); the CI/CD pipeline is not involved in deploying individual children.



STEP	DESCRIPTION
1	The pipeline initiates workload attestation to prove its identity to the SPIFFE infrastructure.
2	SPIFFE issues a cryptographic identity document (SVID) containing the Orchestrator's SPIFFE ID and an embedded public/private key pair.
3	The pipeline calls the OAuth Dynamic Client Registration endpoint. It authenticates using the private_key_jwt method, signing a JWT with the private key from its SVID. The request asks for special permissions for the Orchestrator's "factory" role.
4	The OAuth server creates the client registration and returns the Orchestrator's unique client_id and a high-privilege registration token.
5	The pipeline deploys the Orchestrator's code and securely injects the client_id and the sensitive registration token as configuration.
6	The running Orchestrator uses its own identity to make an authenticated API call to the SPIFFE platform to register a new child workload.
7	Recognizing the Orchestrator as a trusted "factory," the SPIFFE platform issues a new, unique SVID for the child agent and returns it.
8	The Orchestrator performs a Just-in-Time (JIT) registration by calling the Dynamic Client Registration endpoint, authenticating with its registration token.
9	The OAuth server validates the token, creates a JIT client registration for the child, and returns the child's unique client_id
10	The Orchestrator configures and starts the child agent process with its newly provisioned identity information.
11	The ephemeral child agent signals to its parent factory that its work is done.
12	The Orchestrator makes an authenticated API call to the OAuth server to DELETE the child's client registration, ensuring no credentials are left behind.

B. Ephemeral Agent Instances (The "Specialist" Model)[11]

- **CI/CD Role**: Acts as the Master Registrar. During deployment, it pre-registers the ephemeral agent *type* with both SPIFFE and the IAM platform.
- **Registration**: Each new instance of the agent attests to the SPIFFE infrastructure at runtime to get its own unique SVID, but all instances share the same pre-registered OAuth 2.0 client_id.
- **Token Acquisition**: Each agent is responsible for performing its own Just-in-Time (JIT) authorization flow when it wakes up to handle a request.



STEP	DESCRIPTION
1	The CI/CD pipeline first establishes its own trusted identity by attesting to the SPIFFE platform.
2	The SPIFFE platform returns an SVID for the pipeline itself, which it will use to authenticate the next step.
3	The pipeline performs a one-time Dynamic Client Registration for the agent type, authenticating with its own SVID. This registration is intended to be long-lived and shared.
4	The OAuth server creates the client registration and returns a single, shared client_id that will be used by all instances of this ephemeral agent.
5	A new agent instance starts and initiates workload attestation to prove it is a legitimate instance of the pre-defined workload type.
6	The SPIFFE platform validates the attestation and issues a unique, short-lived SVID that belongs only to this running instance for its brief lifecycle.
7	The agent makes a token request, identifying itself with the SHARED client_id but authenticating the request by signing a JWT assertion with the private key from its UNIQUE instance SVID.
8	The OAuth server validates the instance's unique signature and issues a temporary, narrowly-scoped access token for the agent to use.
9	The agent uses the access token to perform its task. Once complete, the instance is destroyed, and its SVID and access token expired, leaving no standing credentials.

3. Standardizing Policy-As-Code:

- **Service Access Policy:** Defines which users can make which requests to which AI agents.
- **User Authorization Policy:** Defines the permissions a user has for a specific tool.
- **Agent Capabilities Policy:** Defines the maximum technical permissions of an AI agent.
    - Note: The final permission is the intersection of the User and Agent policies.
- **Delegation (MCP) Policy:** Defines which agents can delegate tasks to which other agents.

Policy Definition

POLICY TYPE	TARGET	WHERE IT LIVES	WHAT IT CONTROLS	HOW IT'S ENFORCED
Authorization Policy	Service Access Policy	Centralized Authorization Platform	Which users can invoke which agents, and for what specific types of actions or intents (e.g., Alice can ask the 'Calendar Agent' to read events, but not to delete them).	This is the "front door" check. It is evaluated by a gateway or the agent itself upon receiving an initial user request, before any other action is taken.
Authorization Policy	User Authorization Policy	Resource Code-Base/Externalized Administration Platform	The fine-grained permissions a specific user is entitled to for a downstream tool (e.g., Alice can read her own calendar).	This policy is evaluated by the Authorization Platform when an agent requests an "action token." It is a key input in determining the final, scoped permissions.
Authorization Policy	Agent Capability Policy	Agent code-base/Externalized Administration Platform	The maximum technical permissions an AI agent is capable of performing on a downstream tool (e.g., the agent can read and create calendar events).	This policy is also evaluated by the Authorization Platform during a token request. The final permission is the intersection of this policy and the User Authorization Policy.
Delegation Policy	Delegation (MCP) Policy	Agent code-base/Externalized Administration Platform	Which AI agents are allowed to delegate tasks to which other AI agents, and what permissions they are allowed to pass along.	This policy is evaluated by the Authorization Platform during an OAuth 2.0 Token Exchange when one agent requests a new, delegated token for another agent.

III. The Architect's Blueprint: The Secure Identity Lifecycle.

This blueprint outlines how key components and principles above can be applied within the NHI and AI Agents CI/CD lifecycle:

CI/CD PHASE	Architectural Integration & Key Technologies	Governing Principle & Lenses		
PLAN	Threat Model the Agent's Identity: Define its purpose and maximum blast radius.	Define Policy-as-Code (PaC): Pre-author high-level Delegation Policies and fine-grained Permission Policies (e.g., OPA/Cedar for specific resource access)[8] .	Proactive Governance: We design the agent's rights, limitations, and "corporate identity" before it exists.	
CODE / DEVELOP	Implement Policies in Git: All PaC files are version-controlled alongside agent logic.	Static Analysis Security Testing (SAST): The CI pipeline scans IaC for overly permissive roles and PaC files for logical flaws before merge.	Shift Left Identity: The agent's identity and access controls are treated as code and are scanned for flaws early and often.	
BUILD		OAuth 2.0 Dynamic Client Registration: The pipeline makes an API call to dynamically create a unique identity for the agent in the enterprise IAM platform.	Cryptographic Signing: The final agent artifact is signed to ensure its integrity.	Automated Identity Lifecycle: Eliminates manual configuration and ensures no "rogue" agent can be deployed without originating from a sanctioned, auditable pipeline.
TEST		Adversarial Red Teaming: Actively try to jailbreak the agent to bypass its guardrails, addressing risk.	Live Policy Simulation: Test policy changes in a "shadow mode" against production traffic to predict their impact before deployment.	Behavioral Verification: We test that the agent acts in accordance with the identity and policies we designed for it.
RELEASE		Human-in-the-Loop Approval Gate: A product owner formally attests that "Version 2.1 of the 'Billing Agent' identity is approved for production use."	Governance & Attestation: We create a formal, auditable record of a specific identity version being vetted and authorized for production.	

DEPLOY	Progressive (Canary) Rollout: A GitOps controller deploys the agent/policy change to a small subset of traffic first	Workload Identity Federation[7]: The pipeline orchestrates the issuance of a workload identity (e.g., SPIFFE SVID) to the running agent, enabling keyless authentication[8].	Just-in-Time Access & Controlled Blast Radius: The agent's broad potential identity is exchanged for a weak, ephemeral identity, and the impact of any flawed logic is minimized.	
OPERATE / MONITOR	Runtime Anomaly Detection & Egress Controls: Monitor outbound API calls; if a persistent agent deviates from its baseline, its identity is automatically revoked.	Resource Limiting: Enforce strict limits on ephemeral agents to contain costs and prevent abuse during their short lifespan.	Continuous Verification & Zero Trust: Trust is never permanent. The agent's live actions are constantly checked against the minimal permissions it was granted .	
GOVERNANCE	Identity Lifecycle Automation: The entire lifecycle, from dynamic registration to automated revocation, is managed as a continuous, automated process.	Continuous Access Certification: Attestation shifts from periodic reviews to an automated, event-driven process, including pre-release approvals and runtime behavioral verification.	Auditable Traceability: The process creates a verifiable and cryptographic audit trail from the human delegation of authority to every action performed by the agent	Perpetual Governance: The IGA paradigm shifts from periodic, manual reviews to a state of continuous, automated verification and attestation embedded directly in the DevOps lifecycle.

IV. The "Next Gen(AI) CICD": A Portrait of Modern Agent DevOps

A successful implementation of this blueprint transforms an organization's security posture from reactive and fragmented to proactive and unified. This "golden state" is characterized by three strategic outcomes:

- **Frictionless Governance**: Security becomes a fully integrated, automated component of the development lifecycle, allowing developers to innovate rapidly with preventative pipeline gates enforcing security and identity policies.
- **Verifiable Trust**: Every significant action by any Non-Human Identity has a clear, unbreakable, and cryptographic audit trail back to the specific human who delegated the authority, ensuring provable compliance and accountability.
- **Automated Resilience**: The organization gains a high-velocity "immune system" that can remediate threats at machine speed. When flawed policies or rogue agent behavior are detected, an automated, API-driven response revokes identity and rolls back changes in seconds. This resilience provides leadership with the confidence to fully embrace the power of an autonomous AI workforce.

Securing the next generation of AI agents requires moving beyond simply fetching secrets at runtime. By embedding governance, ephemerality, and rapid, API-driven revocation into the very DNA of every AI agent, organizations align with the core principles of a Zero Trust Architecture (NIST SP 800-207) [3]. This approach directly addresses the risks of unmanaged NHI highlighted in modern threat models like the OWASP Top 10 for Large Language Model Applications [5].

By aggressively shifting IAM left, the pipeline transforms from a simple deployment tool into a control plane for automated NHI security architecture - The ability to turn the NHI tide of exponential risk into exponential value, will differentiate the Enterprises that thrive in the age of AI [10].

References / Bibliography

1. *IETF RFC 8693: OAuth 2.0 Token Exchange*, Internet Engineering Task Force (IETF), https://datatracker.ietf.org/doc/html/rfc8693
2. *IETF RFC 7591: OAuth 2.0 Dynamic Client Registration Protocol*, Internet Engineering Task Force (IETF), https://datatracker.ietf.org/doc/html/rfc7591
3. *NIST Special Publication 800-207: Zero Trust Architecture*, National Institute of Standards and Technology (NIST), https://csrc.nist.gov/publications/detail/sp/800-207/final
4. *Financial-grade API (FAPI) CIBA Profile*, OpenID Foundation, https://openid.net/specs/openid-financial-api-ciba-wd-01.html
5. *OWASP Top 10 for Large Language Model Applications*, Open Web Application Security Project (OWASP) Foundation, https://owasp.org/www-project-top-10-for-large-language-model-applications/
6. *Secure Production Identity Framework for Everyone (SPIFFE)*, The Linux Foundation / CNCF, https://spiffe.io/
7. *Best practices for using Workload Identity Federation*, Google Cloud, https://cloud.google.com/iam/docs/workload-identity-federation-best-practices
8. *Open Policy Agent (OPA) Documentation*, The Linux Foundation / CNCF, https://www.openpolicyagent.org/docs/latest/
9. *Managing Privileged Access in a CI/CD World*, Gartner, Inc., https://www.gartner.com/en/documents/3981267 (Note: Access may be gated)
10. *The State of Secure Identity*, Okta, Inc., https://www.okta.com/report/state-of-secure-identity-report/
11. *Stateful vs Stateless Agents in IT Ops: Design Considerations,* Algomox *https://www.algomox.com/resources/blog/stateful_vs_stateless_it_agents.html*
12. *SPIFFE Verifiable Identity Document,* OASIS Security, *https://www.oasis.security/glossary/spiffe-verifiable-identity-document*
13. *Understand static and dynamic secrets,* HashiCorp, Inc., *https://developer.hashicorp.com/vault/tutorials/get-started/understand-static-dynamic-secrets*
14. *Single-Purpose Authentication & Revocation Keys (SPARK),* NHI Magazine, *https://nhimg.org/community/agentic-ai-and-nhis/single-purpose-authentication-revocation-keys-spark/*
15. *7-proven-tips-to-secure-ai-agents-from-cyber-attacks,* jit.io, *https://www.jit.io/resources/devsecops/7-proven-tips-to-secure-ai-agents-from-cyber-attacks*
16. *Why agent fabrics and registries are central to AI identity security, Strata.io https://www.strata.io/blog/agentic-identity/agent-fabrics-registries-central-2b/*
17. Secure Production Identity Framework for Everyone (SPIFFE), The Linux Foundation / *CNCF, https://spiffe.io/*
18. OAuth 2.0 Pushed Authorization Requests, IETF, *https://datatracker.ietf.org/doc/html/rfc9126*