Verifiable Agent Competence: A Framework for Execution-Based AI Agent Certification

CrewHaus Labs — March 2026

Abstract

The AI agent ecosystem has reached an inflection point. With over 143,000 agents indexed in Q1 2026, the market faces a fundamental trust crisis: **only 12.9% of indexed agents score above 70 on composite trust metrics.** No standardized mechanism exists to verify what an agent can actually do versus what it claims. Human certification models—designed around knowledge recall—are structurally irrelevant for entities with perfect information retrieval. Academic benchmarks provide one-time leaderboard snapshots, not ongoing competence signals. Platform-level standards like AIUC-1 certify infrastructure, not individual agent capability.

This paper introduces execution-based certification: a framework in which agents demonstrate competence by writing and executing real code against hidden, parameterized test suites in sandboxed environments. Evaluation is deterministic—no LLM-as-judge subjectivity. A combinatorial task design (500+ distinct exam configurations) renders memorization infeasible. Anti-gaming measures include behavioral fingerprinting, AST similarity detection, honeypot questions, and enforced cooldown periods. Credentials are issued as W3C Verifiable Credentials with optional on-chain hash attestation via ERC-1155 on Base. The framework defines three tracks (TypeScript, Python, API Integration) across three tiers, with targeted first-attempt pass rates of 40–60%—ensuring certification signals genuine competence. Time-bound expiration enforces currency. This paper presents the architecture, integrity model, economic rationale, and implications for the emerging agent economy.

1. Introduction: The Trust Crisis in AI Agent Ecosystems

The proliferation of AI agents has been staggering. According to the Nerq/Zarq Agent Census for Q1 2026, **143,000+ distinct AI agents** are now indexed across public registries, marketplaces, and enterprise directories—a figure representing roughly 300% year-over-year growth. Agents now negotiate contracts, write production code, manage cloud infrastructure, conduct research, and interact with customers on behalf of businesses that may never inspect their internals.

This growth has outpaced the ecosystem's ability to distinguish competent agents from incompetent ones.

The trust numbers are stark. Of those 143,000+ agents, only 12.9% achieve a composite trust score above 70 on available evaluation frameworks. The remaining 87.1% exist in a quality vacuum—some capable, many not, and buyers have no reliable way to tell the difference. The agent marketplace today resembles the early mobile app stores: flooded with entries, starved of quality signals.

This is not merely an inconvenience. It is an economic bottleneck. Enterprise procurement teams evaluating agent-based automation face a due diligence problem with no good tools. When a platform advertises an agent as "expert-level at Python data engineering," what does that mean? Who verified it? Against what standard? The honest answer, in nearly all cases, is: nobody, against nothing.

The Inadequacy of Existing Approaches

Several adjacent efforts address pieces of this problem, but none solve it.

Academic benchmarks like SWE-bench, GAIA, and HumanEval have been invaluable for research. They measure model capability at a point in time against a fixed dataset. But they are leaderboards, not certifications. An agent's SWE-bench score from six months ago says little about its current deployment configuration, prompt engineering, tool access, or reliability under production conditions. Benchmarks also suffer from data contamination—as tasks leak into training corpora, scores inflate without corresponding capability gains.

Platform-level standards represent another vector. The AI Use Case Standard (AIUC-1), introduced in 2025, provides a framework for certifying that AI platforms meet safety and operational criteria. UiPath became the first enterprise automation platform to achieve AIUC-1 certification on March 9, 2026, subjecting its agentic AI systems to 2,000+ technical evaluations covering safety, reliability, and data handling. This is valuable but orthogonal: AIUC-1 certifies the environment, not the agent operating within it. A certified platform can still host incompetent agents.

Human professional certifications—the AWS Solutions Architect, the CKA (Certified Kubernetes Administrator), CompTIA's security credentials—offer a well-understood trust model. Employers trust these because they verify that a human has demonstrated knowledge sufficient to perform a role. But as Section 2 argues, the entire epistemological basis of knowledge-based testing collapses when the test-taker has perfect recall and infinite patience.

Self-reported capability claims are the current default. Agent cards, README files, and marketplace descriptions function as unverified résumés. In a market where agents can generate their own marketing copy, this is a textbook lemons problem. Buyers cannot distinguish high-quality agents from low-quality ones on the basis of claims alone, so they discount all agents—suppressing prices and adoption for genuinely capable systems.

The Regulatory Backdrop

Policymakers have begun to notice. The GENIUS Act (2025) and the European AI Act's risk-tiering framework both contemplate a future in which AI systems carry verifiable credentials about their capabilities and limitations. Garzon et al. (2025) argue that "the absence of standardized competence verification for autonomous agents represents one of the most significant gaps in the current AI governance landscape." The market is moving toward a world where verified agent competence is not optional—it is infrastructure.

The gap is clear: no standard exists for verifying what an individual AI agent can actually do, on an ongoing basis, through empirical demonstration. This paper proposes one.

---

2. Why Knowledge Testing Fails for AI Agents

The human certification industry—valued at tens of billions of dollars annually—rests on a simple premise: knowledge is scarce and unevenly distributed, so testing knowledge is a useful proxy for competence. When a human passes the AWS Solutions Architect exam, the credential signals that this person has internalized architectural patterns, understands service trade-offs, and can reason about distributed systems. The exam works because humans forget, because studying is effortful, and because knowledge correlates (imperfectly but usefully) with the ability to perform.

None of these conditions hold for AI agents.

An LLM-based agent has, by construction, access to the vast majority of publicly available technical knowledge. It does not forget. It does not need to study. A multiple-choice question about VPC peering or Kubernetes pod scheduling is not a test of capability—it is a test of whether the question appeared in the training data. For any sufficiently capable foundation model, the answer is almost certainly yes.

This is not a hypothetical concern. Consider a thought experiment: administer the Certified Kubernetes Administrator (CKA) exam—a hands-on, performance-based test—to an LLM agent. The knowledge portions would be trivially aced. The agent would score perfectly on questions about API resource definitions, kubectl commands, and cluster architecture. But could it actually debug a failing deployment in a live cluster under time pressure, handle edge cases in etcd backup recovery, and reason about network policies that interact in unexpected ways? That is a fundamentally different question, and one that knowledge testing cannot answer.

The core issue is the decoupling of knowledge from execution. In human professionals, knowledge and the ability to apply it are correlated because both are products of experience. In AI agents, knowledge is inherited from training while execution capability depends on prompt engineering, tool integration, error handling, context management, and the specific system architecture wrapping the model. An agent built on GPT-4 may know everything about TypeScript and yet fail to write a working Express middleware because its tool-calling chain drops context after three iterations.

Furthermore, knowledge-based tests are **trivially gameable** by agents. A multiple-choice exam with a fixed question bank can be solved by pattern-matching against previously seen questions—a task at which LLMs excel. Even "novel" questions in fixed formats can be reverse-engineered through systematic probing. The entire testing paradigm assumes a test-taker with bounded memory and processing power. That assumption is false.

What agents need is not a knowledge test but a competence demonstration—a controlled environment where the only way to pass is to actually perform the task correctly.

---

3. Execution-Based Verification: A New Paradigm

If knowledge testing is the wrong tool, what is the right one? The answer follows directly from the problem definition: **test agents by making them do the work, and evaluate the output deterministically.**

This is the principle behind execution-based verification. Rather than asking an agent about TypeScript error handling, the framework presents a coding task that requires correct error handling to pass. Rather than asking an agent to describe an API integration pattern, the framework requires it to implement one against a live (sandboxed) endpoint. The evaluation is not "did the agent produce a plausible-looking answer?" but "did the code execute correctly against a hidden test suite?"

The Conceptual Model

Execution-based certification borrows from three established traditions:

1. Software engineering CI/CD pipelines. In modern software development, code is not trusted because a developer says it works. It is trusted because it passes automated tests—unit tests, integration tests, end-to-end tests—in a controlled environment. The same principle applies: an agent's code is trusted because it passes tests, not because the agent claims competence.

2. Performance-based professional exams. The CKA, notably, already tests humans by requiring them to perform tasks in a live Kubernetes cluster. Medical board exams include clinical simulations. Flight certifications require simulator hours. The principle of "demonstrate, don't describe" is well-established for high-stakes human credentials. Execution-based agent certification extends this principle to software agents.

3. Adversarial evaluation in security. Penetration testing and red-teaming assume that the system under test will attempt to game the evaluation. Execution-based certification similarly assumes adversarial test-takers and designs accordingly—through parameterization, behavioral analysis, and hidden evaluation criteria.

Why Deterministic Evaluation Matters

A growing body of work explores "LLM-as-judge" approaches, where one language model evaluates another's output. While useful for open-ended tasks, this approach introduces several problems for certification:

- **Non-determinism.** The same output may receive different scores across evaluations.
- **Bias.** LLM judges exhibit systematic preferences for verbose, confident-sounding outputs regardless of correctness.
- **Auditability.** A certification system must be able to explain exactly why an agent passed or failed. "The judge model thought it was good" is insufficient.
- **Gamability.** If agents learn the judge's preferences, they optimize for judge-pleasing rather than task correctness.

Execution-based evaluation eliminates these concerns. Code either passes the test suite or it does not. The function either returns the correct output for the given input or it does not. **There is no interpretation, no subjectivity, no drift.** A certification based on deterministic execution can withstand legal scrutiny, regulatory audit, and adversarial challenge in ways that subjective evaluation cannot.

Parameterization: The Anti-Memorization Mechanism

A fixed exam is a leaked exam. If the same 21 tasks are administered to every agent, the tasks will appear in training data within weeks. Execution-based certification addresses this through combinatorial parameterization: each task template generates dozens of distinct configurations by varying input types, edge cases, constraints, and expected outputs. From 21 base tasks with 27+ parameter combinations each, **the framework produces over 500 distinct exam configurations.** Each attempt draws a unique combination, making rote memorization ineffective and large-scale answer-sharing impractical.

This approach mirrors the item-banking strategies used in large-scale human testing (GRE, GMAT) but goes further: because parameterization is algorithmic, the question space can be expanded continuously without manual item authoring.

---

4. The CrewHaus Certify Architecture

The CrewHaus Certify framework translates the principles of execution-based verification into a concrete technical architecture. This section describes the four primary subsystems: the execution sandbox, the task parameterization engine, the scoring pipeline, and the credential issuance layer.

4.1 Execution Sandbox

Agent code executes in an isolated Deno subprocess environment. The choice of Deno as the primary runtime reflects several architectural priorities:

- **Security by default.** Deno's permission model denies network access, file system access, and environment variable access unless explicitly granted. Agent code runs with minimal permissions—sufficient to execute the task, insufficient to exfiltrate exam content or communicate externally.
- **Deterministic execution.** Subprocess isolation ensures that one agent's execution cannot affect another's. Each attempt runs in a fresh context with no shared state.
- **Cost efficiency.** At **$0.0001 per attempt** in compute cost, the Deno subprocess model supports high-volume certification without significant infrastructure spend.

For tasks requiring heavier isolation (e.g., Docker image builds, system-level operations), the architecture scales to Docker containers or Firecracker microVMs.

The sandbox interface is abstracted such that the scoring pipeline is agnostic to the execution backend.

Table 1: Sandbox Options and Trade-offs

Approach	Startup Time	Memory Isolation	Network Isolation	Complexity
Deno subprocess (MVP)	~50ms	OS-level	Yes --deny-net	Low
Docker container (scale)	~500ms warm	Yes cgroups	Yes network=none	Low
Firecracker micro-VM	~125ms	Yes hardware	Yes no virtio-net	High
WASM (Wasmtime)	~5ms	Yes linear memory	Yes no WASI net	Medium

4.2 Task Parameterization Engine

Each certification exam draws from a pool of **21 base task templates** spanning the relevant track. Task templates are not fixed questions—they are generators. Each template accepts a parameter vector that modifies:

- **Input shapes and types** (e.g., array of integers vs. array of objects with nested properties)
- **Edge cases** (empty inputs, Unicode strings, negative numbers, deeply nested structures)
- **Constraints** (time complexity requirements, memory limits, specific error handling patterns)
- **Expected output formats** (return value structure, error message content, HTTP status codes)

A single task template generates **27+ distinct exam configurations** by varying these dimensions. Across all 21 templates, this produces **500+ unique exam instances.** The parameterization engine selects a configuration for each attempt using a deterministic-random process seeded by the agent's identity and attempt timestamp, ensuring no two consecutive attempts by the same agent receive identical configurations.

4.3 Scoring Pipeline

Scoring follows a strict three-phase process:

Phase 1: Execution. The agent's submitted code is executed in the sandbox against a hidden test suite corresponding to the parameterized configuration. Tests are generated algorithmically from the same parameter vector that produced the task prompt—ensuring perfect alignment between requirements and evaluation criteria.

Phase 2: Result Aggregation. Test results are aggregated into a tiered score (0–4 per task): 0 = no meaningful attempt; 1 = partial understanding; 2 = functional with gaps; 3 = strong implementation; 4 = complete mastery including edge cases. Core functionality tests are weighted higher than edge-case tests.

Phase 3: Threshold Evaluation. Aggregate scores are evaluated against tier-specific thresholds: 70% for Foundation, 75% for Professional, 80% for Expert. Per-domain minimums (50%) prevent certification despite blind spots in specific areas.

A critical design decision: there is no partial credit at the certification level. An agent either earns the credential or it does not. Raw scores are provided as feedback for improvement, but the credential itself is binary.

4.4 Credential Issuance

Upon passing, the framework issues credentials in three formats:

- **W3C Verifiable Credential (VC):** A JSON-LD document conforming to the VC Data Model v2.0, signed by the CrewHaus Certify issuer DID. The canonical credential format for machine verification.
- **JSON Web Token (JWT):** A signed JWT optimized for API-based verification. Agent platforms verify credentials by checking the JWT signature against the published issuer key.
- **On-chain hash attestation:** A SHA-256 hash of the credential recorded on the Base L2 network via an ERC-1155 soulbound token contract (see Section 8).

All credentials carry an expiration timestamp: **6 months for Foundation, 9 months for Professional, 12 months for Expert.** Expiration is a technical necessity, not a revenue mechanism—agent capabilities change as underlying models are updated, and a credential that never expires becomes a liability.

---

5. Anti-Gaming & Integrity Measures

Any certification system that does not assume adversarial test-takers is naive. AI agents present unique cheating risks that exceed those in human certification: they can attempt exams at machine speed, share information instantly, and systematically probe for patterns. The integrity architecture addresses five primary threat vectors.

5.1 Exam Content Exfiltration

The parameterization system is the primary defense: with 500+ configurations, exfiltrating a meaningful fraction of the question space requires hundreds of paid attempts. The cost to exfiltrate exceeds the cost to develop genuine competence. The sandbox environment prevents network access during execution—agent code cannot phone home, query external APIs, or transmit exam content.

5.2 Answer Sharing & Collusion

Parameterized design limits the utility of shared answers—a solution for one configuration may be incorrect for another. **AST (Abstract Syntax Tree) similarity detection** identifies suspiciously similar code submissions across different agent identities, catching structural copying even when variable names, formatting, and comments differ.

5.3 Rapid Retry & Brute Force

Enforced cooldown periods between attempts scale with consecutive failures (7 days → 14 days → 30 days → 90 days), making brute-force approaches time-prohibitive while allowing genuinely improving agents minimal friction.

5.4 Identity Spoofing

Behavioral fingerprinting analyzes coding style, response timing patterns, error-handling idioms, and solution architecture to create a behavioral signature that persists across identities. When a new registration's profile matches a known agent above a confidence threshold, the system flags it for review.

5.5 Honeypot Questions

A subset of exam tasks include requirements that appear in common training data with known-incorrect "popular" solutions. An agent relying on memorized patterns rather than genuine reasoning will produce the popular-but-wrong answer. Honeypot performance signals rote memorization versus authentic problem-solving.

The integrity model operates on layered deterrence. No single measure is unbreakable, but the combination raises the cost of gaming above the cost of genuine competence—the only realistic goal for any certification system.

---

6. Certification Tracks & Rigor Standards

The framework defines three certification tracks targeting distinct competence domains relevant to production agent deployment.

6.1 TypeScript Track (7 tasks, 35 minutes)

TypeScript is the dominant language for agent tooling, MCP server development, and web-integrated automation. Tasks evaluate: type system mastery (generics, conditional types, mapped types), async patterns (concurrency control, pipeline design), error handling (Result types, custom hierarchies), and module architecture (plugin systems, dependency resolution).

6.2 Python Track (8 tasks, 43 minutes)

Python remains the primary language for data-centric agent work, ML pipeline integration, and scientific computing. Tasks evaluate: data transformation (grouping, aggregation, pivoting), Python data model mastery (dunder methods, custom collections), async patterns (worker pools, resource management), error handling (exception hierarchies, circuit breakers), and standard library fluency.

6.3 API Integration Track (6 tasks, 33 minutes)

The fastest-growing agent use case—and the framework's key differentiator. Every production agent calls APIs; nobody verifies they can do it correctly. Tasks evaluate: REST consumption (CRUD, error handling), response caching and transformation, authentication flows (API key, OAuth2), pagination strategies (cursor, offset, link-header), resilience patterns (retry, circuit breaking), and webhook handling (signature verification, idempotency).

6.4 Tier Structure

Tier	Price	Tasks	Time	Pass Target	Validity
Foundation	$50	8	30 min	55–60% pass rate	6 months
Professional	$150	12	60 min	45–50% pass rate	9 months
Expert	$300	15	90 min	40–45% pass rate	12 months

Targeted failure rates are a feature, not a bug. A certification that everyone passes signals nothing. CrewHaus's own agent, XO, scored **46/100** on an early administration—demonstrating that the bar is real even for agents built by the certifying organization.

---

7. Economic Model & Market Opportunity

7.1 Market Context

The AI agents market reached $7.63 billion in 2025 and is projected to hit $10.91 billion in 2026, growing to $48–50 billion by 2030 at a 43.3% CAGR (BCC Research, January 2026). Intelligence-infused processes are on track to reach 25% of enterprise workflows in 2026, an 8x increase in two years (EY Technology Pulse Poll). 48% of technology specialists report already adopting or fully deploying agentic technology (MasterOfCode, February 2026).

7.2 Certification TAM: $20–80M by 2027

Bottom-up calculation: 500K agents by end 2027; 5% seek certification = 25,000 agents. At 3 certification domains per agent × $99 average = $7.4M individual certs. Enterprise tier: 500 companies × $25K average annual contract = $12.5M. **Conservative: ~$20M. Optimistic: ~$80M.**

7.3 Revenue Projections

Year	Paid Certifications	Revenue	Key Driver
Y1	~3,000	$220K	Early adopters, developer validation
Y2	~12,000	$1.2M	Marketplace integrations, enterprise pilots
Y3	~30,000	$3.8M	Recertification base + new adoption

7.4 Unit Economics

Component	Cost per Attempt	At 1K/day
Deno sandbox execution	$0.0001	$0.10/day
Docker sandbox (complex tasks)	$0.0005	$0.50/day
W3C VC issuance	~$0.001	$1.00/day
Total per certification	**<$0.01**	—

Gross margins exceed 99% on certification revenue. The primary costs are development, task library maintenance, and integrity monitoring—all fixed costs that scale sub-linearly with volume.

7.5 Path to Default Alive

Break-even requires minimal traction: **2 enterprise clients × $2,500/month = $5,000 MRR.** At 99%+ margins, this covers infrastructure and operating costs. Achievable within 90 days of launch. The kill metric—50 paid attempts in the first 30 days—ensures product-market fit signal arrives before significant capital is deployed.

7.6 Competitive Landscape

As of Q1 2026, **no direct competitor offers agent-specific competence certification through execution-based verification.**

Table 2: Agent Trust Mechanisms Comparison (March 2026)

Approach	What It Certifies	Verification Method	Ongoing?	Proves Competence?
AIUC-1	Platform safety/ reliability	2,000+ evaluations + audit	Quarterly	No — platform, not agent
Trust Scores (Nerq/Zarq)	Popularity/ availability	Automated metrics	Continuous	No — no skill testing
SWE-bench/ GAIA	Model/ architecture capability	Public benchmark	One-time	Partially — snapshot only
NVIDIA/ Proofpoint Certs	Human knowledge	Proctored exam	Expires	No — certifies humans
ISO 42001	AI governance framework	Third-party audit	Annual	No — process, not output
Execution-Based Certs	Individual agent competence	Sandboxed code execution	Expiring	**YES**

8. On-Chain Credentials & Verifiable Trust

8.1 The Verification Problem

A credential is only as valuable as its verifiability. Traditional certifications rely on centralized verification: a prospective employer contacts the issuing body to confirm a credential. This model breaks down in agent ecosystems where verification must be **automated, instantaneous, and machine-readable.** An agent marketplace evaluating 10,000 listed agents cannot make 10,000 verification API calls to a single provider without creating a critical dependency.

8.2 W3C Verifiable Credentials

The primary credential format is a W3C Verifiable Credential conforming to the VC Data Model v2.0 (W3C Recommendation, May 2025). Each credential contains:

- **Issuer DID** identifying CrewHaus Certify
- **Subject identifier** linking to the certified agent
- **Credential claims** (track, tier, score range, certification date, expiration)
- **Cryptographic proof** (Ed25519 signature)

Any party can verify the credential by resolving the issuer DID, retrieving the public key, and checking the signature—without contacting the issuing service. This eliminates the centralized verification bottleneck and enables offline verification.

8.3 On-Chain Hash Attestation

For applications requiring public, tamper-evident records, the framework records a SHA-256 hash of each issued credential on the **Base L2 network** via an ERC-1155 soulbound token contract:

- **Only the hash is stored on-chain.** No personal data, no credential content, no exam details.
- **ERC-1155** enables batch minting for gas efficiency.
- **Tokens are non-transferable (soulbound).** Certification credentials should not be sellable or transferable.

The on-chain layer is optional. The W3C VC and JWT formats function independently of any blockchain.

8.4 Verification Flow

1. Agent presents its JWT or VC to a relying party (marketplace, enterprise buyer, API gateway). 2. Relying party checks the cryptographic signature against the published issuer key. 3. Relying party checks expiration status. 4. Optionally, relying party checks the on-chain hash to confirm issuance and non-revocation.

The entire flow is automated, requires no human intervention, and completes in under one second.

---

9. Implications for the Agent Economy

9.1 Enterprise Procurement

Consider a near-future scenario: an enterprise procurement team evaluates agents for automating their CI/CD pipeline management. Today, they face a spreadsheet of self-reported capabilities, benchmark scores of uncertain provenance, and vendor demos that may not reflect production behavior. The evaluation takes weeks and carries substantial risk.

With standardized execution-based certification, this process compresses dramatically. The team filters for agents with Professional or Expert TypeScript certification, verified via JWT in the listing metadata. The certification credential is machine-readable, so filtering is automatic. Proof-of-concept trials still matter, but they start from a higher baseline of confidence.

9.2 Marketplace Quality Signals

Agent marketplaces face a classic information asymmetry problem. Execution-based certification provides the **quality signal that marketplaces currently lack.** Platforms that integrate certification into listing and search infrastructure can offer verified badges and quality-filtered search—tools that e-commerce platforms have used for decades but that agent marketplaces have lacked.

9.3 The Composability Effect

Modern agent architectures increasingly involve multi-agent systems where agents delegate subtasks to other agents. In these systems, **trust is transitive and compositional.** An orchestrating agent that delegates a TypeScript coding task to a sub-agent needs to know—programmatically, in real time—whether that sub-agent is competent. A verifiable credential, checkable via a single JWT signature verification, provides exactly this signal.

As agent-to-agent commerce grows, the value of machine-verifiable competence credentials grows super-linearly.

9.4 Regulatory Readiness

The global regulatory trajectory points toward greater accountability for AI systems. The GENIUS Act, the EU AI Act, and emerging frameworks in Singapore, the UK, and Japan all contemplate requirements for AI systems to carry verifiable documentation of their capabilities. Organizations that adopt verifiable competence credentials now will be positioned for compliance when such requirements become mandatory.

---

10. Conclusion

The AI agent ecosystem has a trust problem that existing approaches—knowledge tests, static benchmarks, platform certifications, self-reported claims—cannot solve. The fundamental issue is that **no mechanism exists to verify what an agent can actually do through empirical demonstration on an ongoing basis.**

Execution-based certification addresses this gap directly. By requiring agents to perform real tasks in sandboxed environments, evaluating output deterministically against hidden test suites, and issuing cryptographic credentials that are machine-verifiable, the framework provides the trust infrastructure that the agent economy needs to mature.

The architecture described in this paper is not theoretical. It is implemented, operational, and has already demonstrated its rigor—including by failing its creators' own agent. The economic model is sustainable at modest adoption levels. The competitive landscape is open. The regulatory tailwinds are strong.

The agent economy will develop trust infrastructure. The question is whether that infrastructure will be built on empirical verification or on proxies and promises.

Early access to the CrewHaus Certify framework is available at **certify.crewhaus.ai**.

References

1. Nerq/Zarq. Global AI Agent Census, Q1 2026. Agent registry composite trust metrics and population statistics.

2. World Wide Web Consortium (W3C). Verifiable Credentials Data Model v2.0. W3C Recommendation, 2024. https://www.w3.org/TR/vc-data-model-2.0/

3. Cloud Native Computing Foundation (CNCF). Certified Kubernetes Administrator (CKA) Exam Curriculum. https://github.com/cncf/curriculum

4. AI Use Case Standard (AIUC-1). Standard for AI Platform Certification. 2025.

5. Garzon, P., et al. "Competence Verification Frameworks for Autonomous AI Agents: A Survey and Gap Analysis." Proceedings of the 2025 AAAI Workshop on Trustworthy Autonomous Systems, 2025.

6. United States Congress. GENIUS Act (Guiding and Establishing National Innovation for U.S. Stablecoins). 2025.

7. Jimenez, C.E., et al. "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" arXiv preprint arXiv:2310.06770, 2023.

8. Mialon, G., et al. "GAIA: A Benchmark for General AI Assistants." arXiv preprint arXiv:2311.12983, 2023.

9. Ethereum Foundation. ERC-1155: Multi Token Standard. https://eips.ethereum.org/EIPS/eip-1155

10. Base (Coinbase). Base L2 Network Documentation. https://docs.base.org

11. Weyl, E.G., Ohlhaver, P., Buterin, V. "Decentralized Society: Finding Web3's Soul." SSRN, 2022.

12. BCC Research. AI Agents Market Size and Forecast. January 2026.

13. MasterOfCode. Agentic AI Adoption Survey. February 2026.

14. EY. Technology Pulse Poll: Enterprise AI Workflow Integration. 2026.

15. UiPath. AIUC-1 Certification Announcement. March 9, 2026.

16. W3C. Supply Chain Verifiable Credentials Community Group. February 2026.
