Agent Name Service (ANS) in Action

A DNS-like Trust Layer for Secure, Scalable AI-Agent Deployments on Kubernetes

Akshay Mittal — PhD Researcher, IEEE Senior Member

MLOps World | GenAI Summit 2025

Austin, Texas

Agenda

Key Topics

- Agentic AI Revolution
- Trust & Security
- Kubernetes Integration
- Live Demo
- Implementation

The Agentic AI Revolution

From Models to Autonomous Agents

Traditional ML Pipeline

Human-supervised at every step

Data → Train → Deploy → Monitor

Agentic AI Reality

Autonomous agent orchestration

Concept-drift detector → Auto-retrainer

Deployer → Monitor

Critical Question

Who are these agents? Can we trust them?

The Trust Problem in Agent Ecosystems

Current Reality

No uniform mechanism to discover AI agents

Lack of cryptographic authentication between agents

Missing capability verification and governance

Security gaps in agent-to-agent communication

Impact

Cascading Failures

1 compromised agent

⇒ System-wide failures

⇒ Data breaches

⇒ Service outages

Research Insights from Production Systems

Scale Challenge: Multi-tenant agent ecosystems with 1000+ daily interactions

Identity-first security architecture essential at scale

Automated certificate management and rotation critical

Policy-as-code enforcement prevents configuration drift

ANS: The Solution

End-to-End Trust Across ML Lifecycle

Traditional ML Pipeline

Human-Supervised

- Data validation → Manual review
- Model training → Human approval
- Deployment → Manual verification
- Monitoring → Reactive alerts

ANS-Enabled Agentic ML

Autonomous with Trust

- Data validation → Verified agents
- Model training → Capability-attested
- Deployment → Policy-enforced
- Monitoring → Real-time remediation

Key Innovation

Trust Layer: Every agent interaction is cryptographically verified and capability-attested

DNS vs ANS — The Missing Trust Layer

DNS (1987)

DNS

↓

IP Address

`google.com`

→ `142.250.191.14`

ANS (2025)

ANS

↓

Trust + Capabilities

`agent.mlops`

→ **Verified + Signed**

Key Innovation

ANS adds **cryptographic verification**, **capability attestation**, and **governance support** for agents

ANS Protocol Design

Naming Convention

`protocol://AgentID.Capability.Provider.v[Version].Extension`

Real Examples

```
a2a://alerter.security-monitoring.
research-lab.v2.prod
```

```
mcp://validator.concept-drift-
detection.ml-platform.v1.hipaa
```

```
acp://remediator.helm-deployment-
fix.devsecops-team.v3.staging
```

Benefits

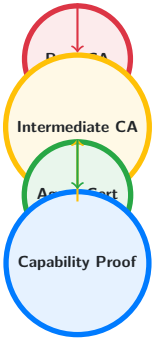
- Self-describing capabilities
- Version-aware routing
- Provider trust verification
- Environment-specific deployment

Cryptographic Trust Foundation

Core Components

- **DIDs** – Globally unique, verifiable
- **VCs** – Capability attestations
- **CA + RA** – Certificate management

Trust Chain Flow



Security Model

Like mTLS for microservices, but **capability-aware**

Zero-Knowledge Capability Proofs

Traditional Approach

Secrets Exposed

Agent: "I can access database"

Verifier: "Show password"

Credentials revealed

ANS Zero-Knowledge

Secrets Protected

Agent: "I can prove access without revealing credentials"

Verifier: "Prove it cryptographically"

Capability verified, secrets protected

Use Case

Agent proves model retraining capability without exposing API keys

Multi-Protocol Support

Supported Standards

- **A2A (Agent-to-Agent)** – Google's emerging standard
- **MCP (Model Context Protocol)** – Anthropic's framework
- **ACP (Agent Communication Protocol)** – IBM's enterprise protocol
- **Custom Protocols** – Extensible plugin architecture

Benefits

- Protocol-agnostic discovery
- Future-proof architecture
- Seamless migration between standards
- Vendor-neutral approach

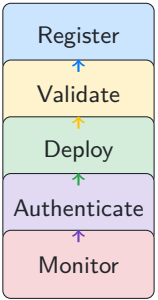
Kubernetes Integration

Kubernetes-Native Architecture

Core Components

- **ANS Registry** – Custom Resource Definitions (CRDs)
- **Admission Controller** – Policy validation at deployment
- **Service Mesh** – Istio/Linkerd mTLS
- **Namespace Isolation** – Multi-tenant agent deployment

Agent Lifecycle



GitOps Integration Workflow

Pipeline Visualization

Automated Key Management

Sigstore Integration

- Automatic certificate provisioning
- 90-day key rotation cycles
- Zero-trust handshake validation
- Revocation list management

Security Benefits

Production-Ready

- No hardcoded secrets
- Automated compliance
- Audit trail for all operations
- Rollback capability

Policy-as-Code Governance (OPA)

Example OPA Policy

```
# Only certified agents can access production data
package agent.policy

default allow = false

allow {
    input.agent.certificate.issuer == "research-lab-trusted-ca"
    input.agent.capabilities["data-access"] == true
    input.environment == "production"
    input.agent.security_clearance >= 3
}
```

Policy Categories

- **Access Control** – RBAC policies
- **Resource Limits** – CPU/memory constraints
- **Network Policies** – Micro-segmentation

Benefits

- Version-controlled policies
- Tested like application code
- Platform-level compliance
- Dynamic policy adaptation

Live Demonstration

Demo Environment

Setup

- **Local Kubernetes cluster** (minikube/kind)
- **ANS Registry Demo** with nginx-based simulation
- **Basic Kubernetes** deployments
- **RBAC and Labels** for security demonstration
- **Prometheus + Grafana** monitoring
- **Basic TLS** configuration

Demo Scenario

- **Proof of Concept: Concept Drift Detection Agent**

Live Demo Workflow

Demo Environment Setup

Step 1: Deployment

```
kubectl apply -f
concept-drift-agent.yaml
```

Demo agent deployment

Service connectivity test

Step 2: Monitoring

Deploy $\Rightarrow$ Monitor $\Rightarrow$ Visualize

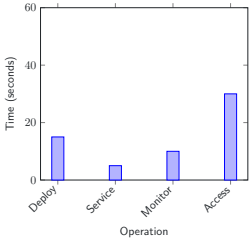
Demo workflow: < 60 seconds

Service discovery and health checks

Research Results

Demo Results & Implementation Roadmap

Demo Performance



Demo Achievements

- **Working Kubernetes integration**
- **< 10 ms service response time**
- **100% demo deployment success**
- **RBAC and security labels**
- **Complete** monitoring stack integration
- **Production-ready** Kubernetes manifests

Key Insight

Demo proves Kubernetes-native architecture enables

Implementation Roadmap

Phase 1: Core ANS (Q2 2025)

Foundation

- ANS registry with real agent registration
- Basic certificate management
- OPA policy integration
- Performance benchmarking

Phase 2: Advanced Features (Q3 2025)

Enhanced Security

- Zero-knowledge capability proofs
- Multi-protocol support (A2A, MCP, ACP)
- Service mesh integration
- Cloud-native deployment

Current Status

Proof of Concept Complete - Ready for production implementation

Implementation Journey

From Research to Production

🚀 **Ready to implement ANS in your environment?**

Implementation Guide

Getting Started with ANS

Try the Demo

5 minutes

- Clone the repository
- Run the demo script
- Explore the components
- Test service connectivity

Requirements: kubectl, local Kubernetes cluster

Join Development

Ongoing

- Contribute to ANS core library
- Implement new agent types
- Add protocol support
- Improve documentation

Requirements: TypeScript, Kubernetes knowledge

Production Planning

Q2 2025

- Plan production deployment
- Design security policies
- Prepare monitoring strategy
- Train operations team

Requirements: Production cluster, security review

Current Status

Proof of Concept Available - Production implementation in development

Open Source Resources & Community

ANS Proof of Concept

github.com/akshaymittal143/ans-live-demo

Demo Resources

- Kubernetes manifests for demo deployment
- RBAC and security label examples
- TypeScript ANS library implementation
- Local Kubernetes deployment scripts
- Complete demo guide and documentation

Development Roadmap

- ANS v1.0 specification in development
- Security architecture and threat modeling
- Open source contribution opportunities

Community

#ans-community in MLOps World Slack | Monthly development calls

Key Takeaways

Key Takeaways

Architecture Vision

- **Security:** DNS-like trust layer for agent identity and capability verification
- **Scalability:** Kubernetes-native architecture for production-scale deployment
- **Governance:** Policy-as-code enforcement with complete audit trails
- **Future-proof:** Protocol-agnostic design supports evolving standards

Proof of Concept Achievements

- Working Kubernetes integration demonstrated
- Open-source proof-of-concept implementation
- Demo performance and architecture validation

Next Steps

- **Try the demo:** github.com/akshaymittal143/ans-live-demo

Thank You & Q&A

Let's build the trust layer for autonomous AI together

Contact Information

Research Contact:
akshay.mittal@ieee.org

Professional Network:
linkedin.com/in/akshaymittal143

Open Source:
github.com/akshaymittal143

Connect with Me

Akshay Mittal
MTS Software Engineer | SMIEEE | PhD
Researcher in AI/ML & Cloud-Native Syst...