

# **Agent Name Service (ANS) in Action**

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


**Akshay Mittal** — PhD Scholar

**MLOps World | GenAI Summit 2025**




# 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

# 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

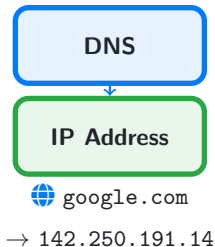
Data validation → Verified agents  
Model training →  
Capability-attested  
Deployment → Policy-enforced  
Monitoring → Real-time  
remediation

## Key Innovation

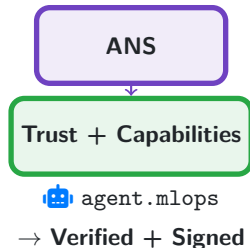
 **Trust Layer:** Every agent interaction is cryptographically verified and capability-attested

# DNS vs ANS — The Missing Trust Layer

## DNS (1987)



## ANS (2025)



## 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

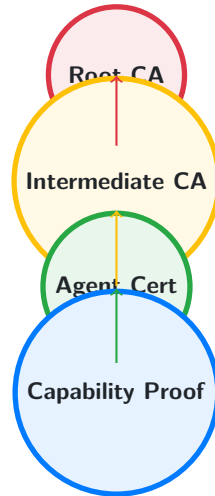
-  Self-describing capabilities
-  Version-aware routing
-  Provider trust verification
-  Environment-specific deployment

# Cryptographic Trust Foundation

## Core Components





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

## Trust Chain Flow







# 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

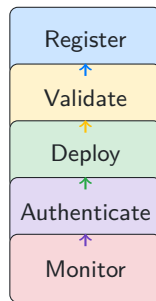


# Kubernetes-Native Architecture

## 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

# Live Demo Workflow

## Demo Environment Setup

Kubernetes Cluster

ANS Registry

RBAC & Labels

Monitoring

## Step 1: Deployment

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

Demo agent deployment  
Service connectivity test

## Step 2: Monitoring

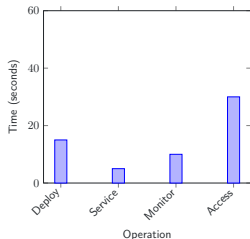
⚡ Deploy ⇒ Monitor ⇒ Visualize  
Demo workflow: < 60 seconds  
Service discovery and health checks

## Live Demo





[!\[\]\(4fe57c3593bf1b21d272ae7ac8dfaf77\_img.jpg\) Watch the Complete Demo](#)

# 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


# 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](https://github.com/akshaymittal143/ans-live-demo)

 **Join development:** Contribute to ANS core library

## Let's build the trust layer for autonomous AI together

### Contact Information

#### **Research Contact:**

[akshay.mittal@ieee.org](mailto:akshay.mittal@ieee.org)

#### **Professional Network:**

[linkedin.com/in/akshaymittal143](https://linkedin.com/in/akshaymittal143)

#### **Open Source:**

[github.com/akshaymittal143](https://github.com/akshaymittal143)

### Connect with Me



**Akshay Mittal**

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

