Friday, 4:47 PM

PagerDuty fires. Production payment service is down.

Someone scaled the deployment manually to **50 replicas** to handle a traffic spike.

ArgoCD detected drift. Auto-synced it back to **10 replicas**.

The app crashed under load. Customers couldn't pay.

ArgoCD did exactly what we told it to do. That was the problem.



ArgoCon
NORTH AMERICA

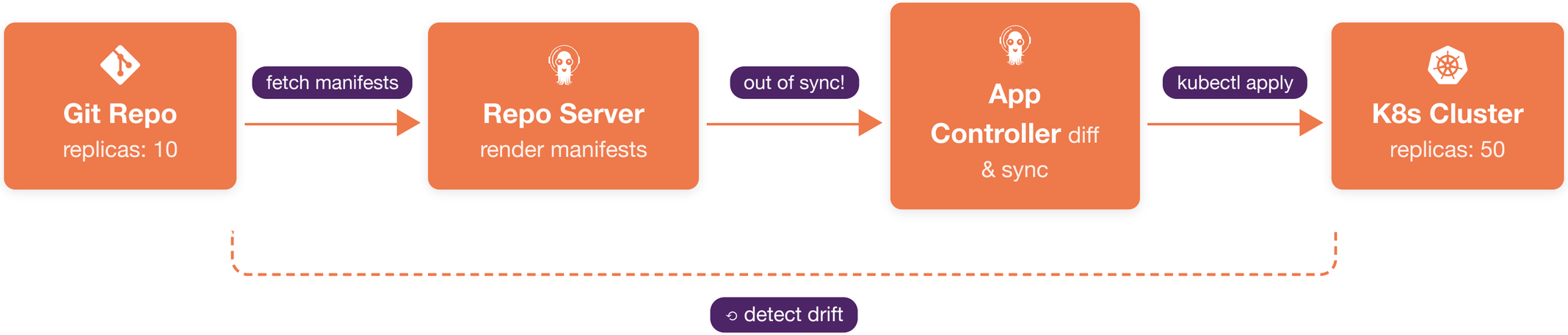
Beyond GitOps: Building Intelligent Drift Detection and Auto-Remediation in Argo CD

ArgoCon North America 2026 · Salt Lake City

Shibi Ramachandran

Ram Mohan Rao Chukka

Stage 1: ArgoCD Out of the Box v2.0+



Git is the single source of truth
Automatic sync on drift detection
Every drift is treated the same - sync it back, no questions asked

The Problem

Production Outages

Rigid sync rules cause crashes under unexpected load or legitimate scaling.

Security & Compliance

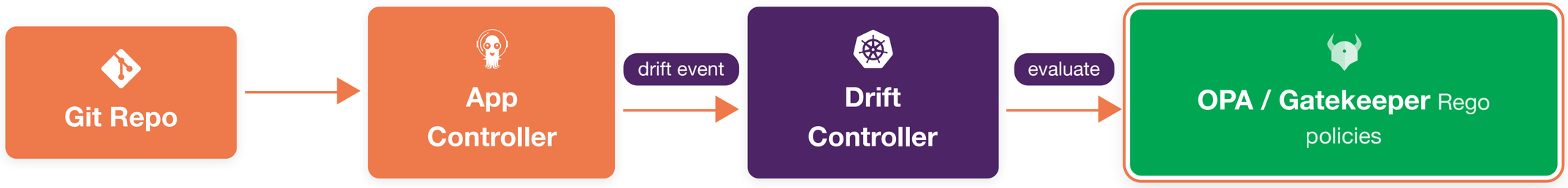
Unauthorized changes bypass the entire GitOps model - the system has no way to detect intent.

No Room for Nuance

Every drift is treated equally - no way to distinguish between legitimate changes and actual problems.

Stage 2: Policy-Driven with OPA v2.12+

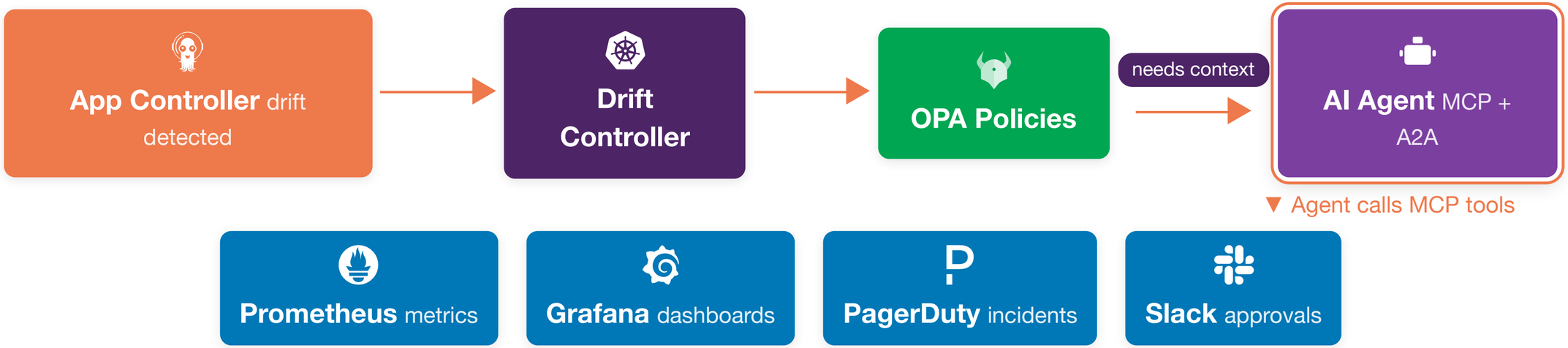
"Teach the system the rules"



Rego Policies - platform teams define drift rules without code changes
Sync Windows - OPA output schedules deferred syncs to maintenance windows
Custom Health Checks - health status now reflects policy verdict, not just diff
Policies can't learn from patterns - still a static rulebook

Stage 3: AI-Powered Intelligence v3.0+

"Let the system reason about drift"



AI queries Prometheus: "*Is this scaling due to actual load?*"
Checks PagerDuty: "*Is there an active incident justifying this change?*"
Asks Slack for human approval when confidence is low

The Protocols: MCP & A2A

Model Context Protocol

AI to Tools - A standard interface for LLMs to call external tools

- Exposes `tools` as typed JSON-RPC functions
- Agent calls `query_prometheus("rate(http_requests[5m])")`
- Server returns structured data - not HTML, not dashboards
- Same MCP server works with any LLM provider

Agent-to-Agent (A2A)

Agent to Agent - Agents discover and delegate to each other

- Drift Agent delegates to `Security Agent` for RBAC analysis
- Security Agent delegates to `Compliance Agent` for audit
- Each agent has its own MCP tools and specialization
- Agent Cards advertise capabilities for discovery

Friday 4:47 PM: ConfigMap drift during production hours



DEMO

Replaying the Friday Incident

Same drift, completely different outcome with Stage 3

App Controller Detects drift: `spec.replicas` changed from **10** → **50** on `payment-service`

Drift Controller OPA policy says: "*replicas on HPA-managed deployments → DEFER to AI Agent*"

AI Agent → MCP Calls `query_prometheus("rate(http_requests_total{service='payment'}[5m])")`
→ *Result: 12,400 req/s (4× normal)*

AI Agent → MCP Calls `check_pagerduty(service='payment-service')`
→ *Result: Active incident INC-4421 "Traffic spike from partner API"*

AI Decision **ALLOW drift** - Traffic is 4x normal with active incident. Scaling is justified.
Log decision, suppress sync, notify #platform-ops on Slack.

No outage. No rollback. No 3 AM page. The system understood context.

Guardrails: Trusting AI in Production

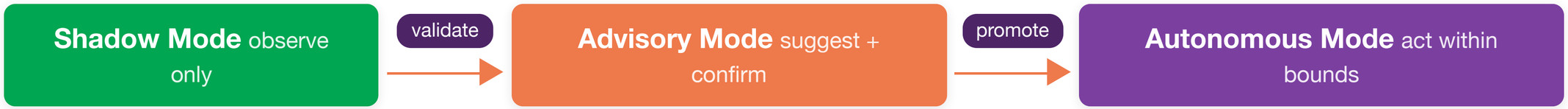
Safety Boundaries

- AI can **ALLOW** or **DEFER** - never **DELETE**
- Confidence threshold: below 80% -> ask human
- OPA policies act as hard guardrails

  "Never auto-approve image tag changes"
- Shadow mode: run alongside existing sync for 30 days

Observability

- Every decision logged with full reasoning chain
- Grafana dashboard: drift decisions / hour
- Alert on AI not-equal-to OPA disagreements
- Weekly accuracy review: was the AI right?



The Evolution

Basic Detection →

Stage 1 v2.0+

ArgoCD detects drift -> Auto-sync without understanding context

Limitation: no nuance, binary response to all changes

Policy-Driven Decisions →

Stage 2 v2.12+

OPA policies define rules per resource -> Auto-remediate, escalate, or ignore

Breakthrough: policy-driven decisions without code changes

Agentic Reconciliation →

Stage 3 v3.0+

AI Agent queries Prometheus, PagerDuty, Slack via MCP -> context-aware decisions

*Breakthrough: understands **why** drift happened, not just **what** changed*

From **"what changed?"** > **"should we fix it?"** > **"why did it change, and what's the right thing to do?"**

Key Takeaways

1. **Not all drift is equal** - scaling a deployment isn't the same as deleting a secret. Your tooling should understand the difference and respond accordingly.

2. **ArgoCD already has the building blocks** - Custom Health Checks, Resource Hooks, ApplicationSets, and Sync Windows. You don't need to replace it - you need to extend it.

3. **Guardrails are non-negotiable** - start with OPA policies. Only bring in AI for the ambiguous cases. And always: shadow mode first, human-in-the-loop, hard boundaries.



ArgoCon
NORTH AMERICA

Thank You!

Beyond GitOps: Building Intelligent Drift Detection and Auto-Remediation in ArgoCD

Rate Session

Connect

Shibi Ramachandran

Senior Software Engineer