

APEX: **A**gentic **P**latform **E**ngineering

The Agentic Force Multiplier

Eng-Hwa Tan

Principal Specialist Solution Architect
AWS

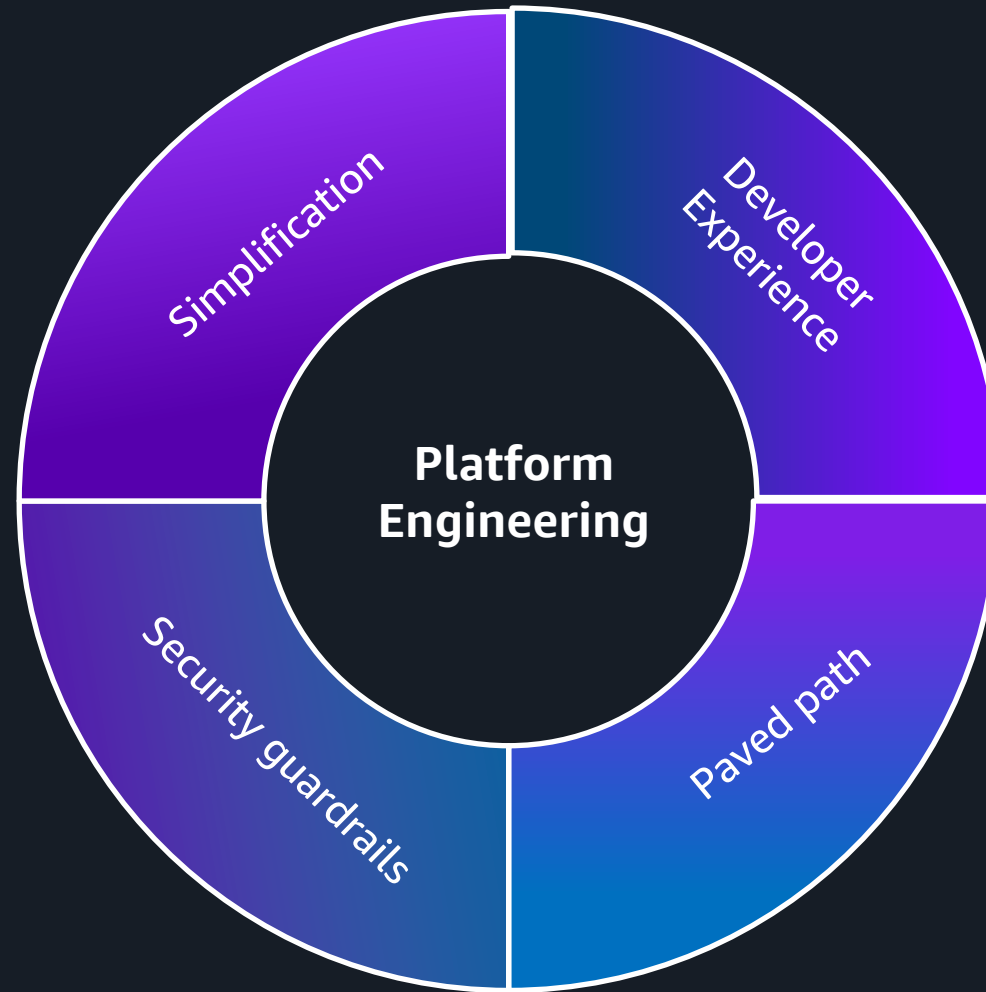
Masatoshi Hayashi

Senior Specialist Solution Architect
AWS

Unmanageable growth

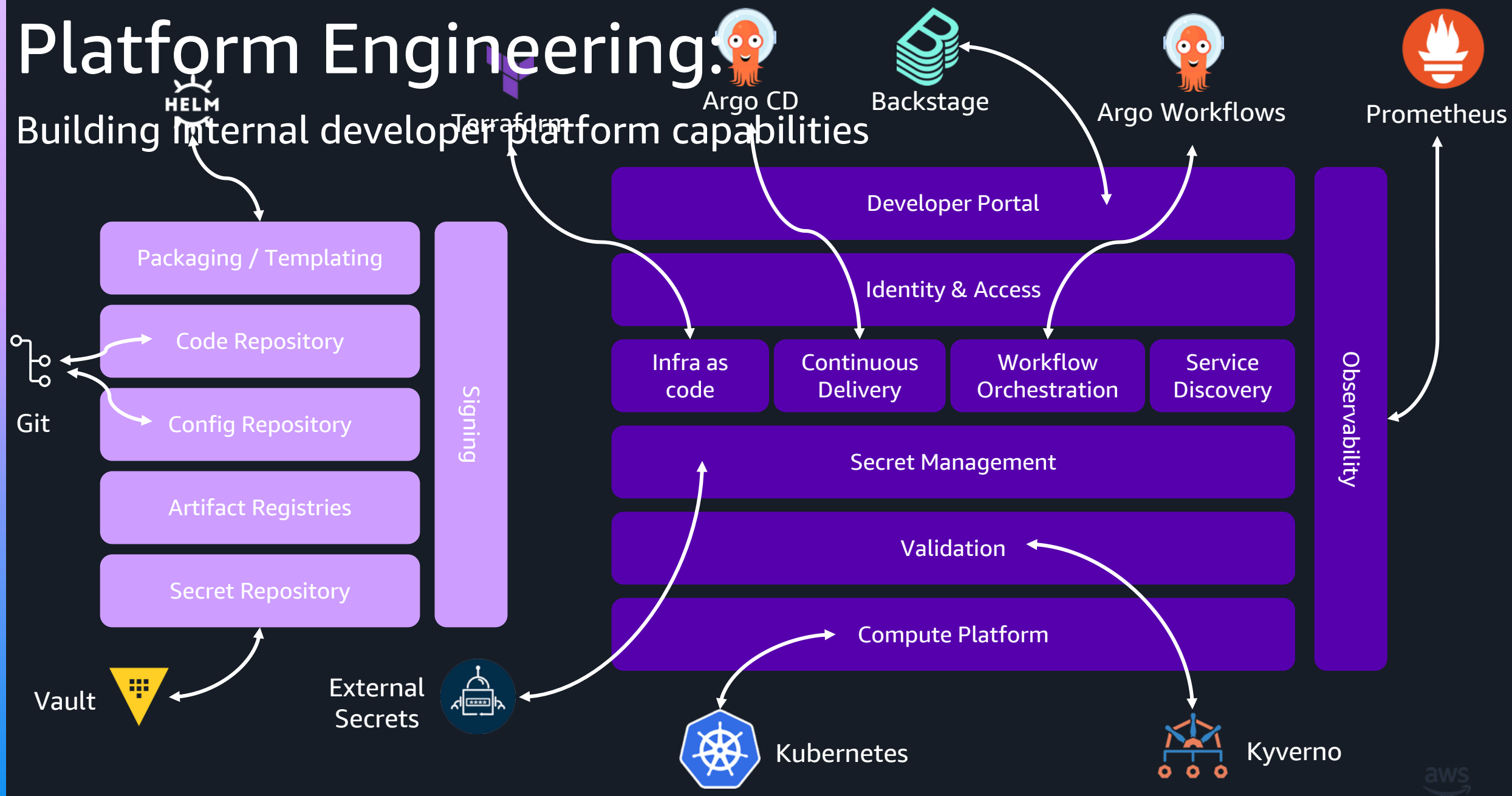


Platform Engineering

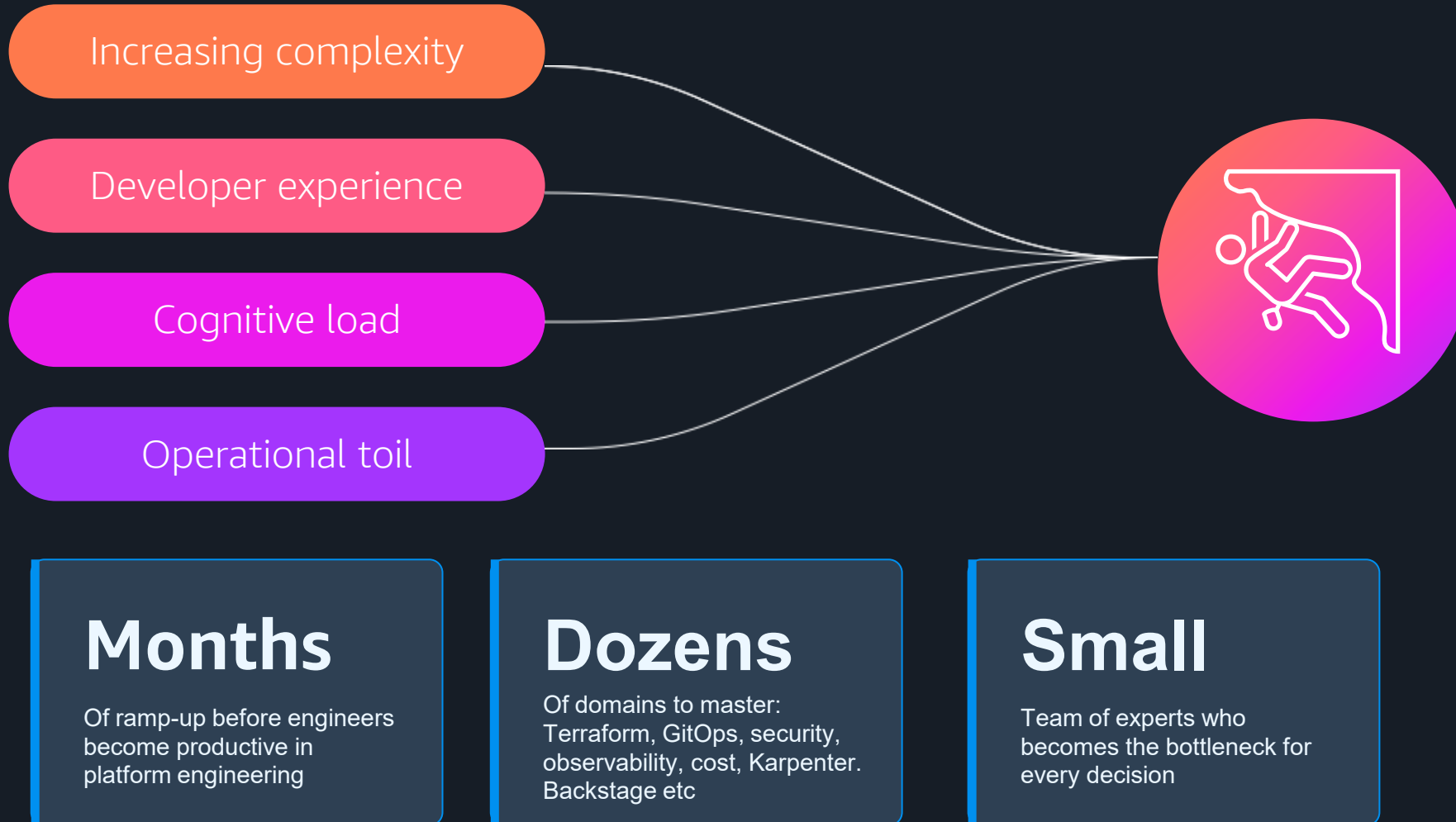


Platform Engineering:

Building Internal developer platform capabilities



The platform engineering bottleneck

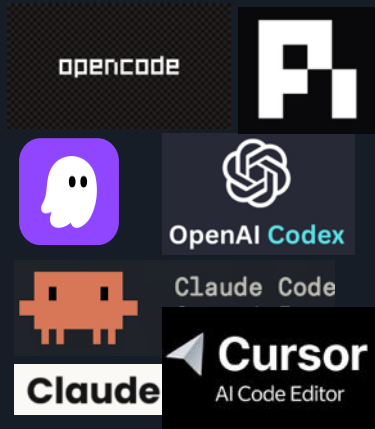


APEX: Scaling platform engineering with AI agents

APEX isn't another tool to adopt. It's a capability framework that turns scarce expertise into a scalable, compounding asset.

It pairs your engineers with AI agents that carry codified AWS expertise — architecture patterns, operational runbooks, security guardrails — so every engineer works as if a senior specialist is beside them.

Agents as a *force multiplier* for Platform Engineers



Agents



Platform Engineers

LAYER 1

Agent Harness

eg: Claude Code / Kiro CLI / Cursor / OpenCode

LAYER 2

Curated Skills

Expert-authored best practices, decision frameworks, Terraform patterns

github.com/aws-samples/sample-apex-skills

Agentic engineering

- 1 Engineer describes **intent** to Agent
- 2 Agent proposes implementation
- 3 Steering by SKILL.md, Agents.md etc, Human-in-the-Loop

Amplify and
Scale best practices

Architect: Design with AI-guided decisions

What the agent does

- Design guidance and trade-off analysis
- Service selection recommendations
- Topology decisions (VPC, subnets, AZs)
- Compute option evaluation

Decisions guided by skills

- Karpenter vs MNG vs Auto Mode
- Networking patterns (VPC CNI modes)
- Security posture (Pod Identity, IRSA)
- Cost optimization strategies

EXAMPLE PROMPT

"Design a multi-tenant EKS platform for 5 departments with cost visibility"

SKILLS ACTIVATED

OUTPUT

Architecture Decision Record (ADR)

Build: Generate and deploy IaC with guardrails

What the agent does

- Transform architecture into Terraform
- Deploy EKS cluster and add-ons
- Configure GitOps tooling (Argo CD)
- Apply security scanning (eg: Checkov)

Built-in validation

- terraform fmt and validate
- Checkov security scans
- OPA policy checks
- Human review gates

EXAMPLE PROMPT

"Generate Terraform for a private EKS cluster with Karpenter"

SKILLS ACTIVATED

GENERATED TERRAFORM SNIPPET

```
module "eks" {  
  source = "terraform-aws-modules/eks/aws"  
  version = "~> 20.0"  
  
  cluster_name      = "platform-eks"  
  cluster_version   = "1.31"  
  
  cluster_endpoint_public_access = false  
  enable_irs         = true  
}
```

OUTPUT

Working EKS platform via IaC



Operate: EKS Upgrade Readiness and Operation Review

EKS Upgrade Readiness Skill

- EKS Upgrade Readiness Prep
 - Readiness Scoring
- Well Architected Review based on Best Practices

EKS Operation Review

- Cluster Lifecycle
- Observability
- Networking
- Workload ..

Readiness Score: 71% — FAIR

Your cluster has several issues that need attention before upgrading to 1.31. The two critical blockers are: (1) Karpenter v0.37.0 is incompatible with EKS 1.31 and

EKS Operation Review Report

Cluster: test-old-cluster | Region: us-east-1 | Version: 1.30 | Platform: eks.63
Date: 2026-04-02
Account: 006*****

Executive Summary

test-old-cluster is an eksctl-managed EKS cluster running Kubernetes 1.30 on Extended Support, with 2 nodes across 2 AZs serving approximately 30 pods. The cluster demonstrates some good foundations — Karpenter is deployed with consolidation enabled, RBAC is appropriately scoped, Pod Disruption Budgets cover most workloads, CoreDNS is properly configured, and IP capacity is abundant.

However, the cluster has significant operational gaps across security and observability. The API server is publicly accessible to **0.0.0.0/0** with no private endpoint and all control plane logging disabled — meaning there is no audit trail and the API server is internet-exposed. The entire observability stack is absent: no metrics collection, no centralized logging, and no alerting. This means issues cannot be detected until users report them. The EBS CSI driver is in CrashLoopBackOff due to missing IAM credentials, impacting any EBS volume operations.

The most immediate cost concern is Extended Support: at \$0.60/hr vs \$0.10/hr, the cluster is paying approximately \$4,300/month extra. Upgrading to v1.32+ would eliminate this premium. Combined with the 13 RED findings (40.6% of rated items), this cluster requires urgent attention across security, observability, and version currency before it can be considered production-ready.

Maturity Score

Rating	Count	Percentage
GREEN	6	18.8%
AMBER	13	40.6%
RED	13	40.6%
UNKNOWN	5	--

Total items assessed: 37 (32 rated + 5 UNKNOWN)

HOW IT WORKS



OUTPUT

EKS Operation Review and Actionable items

Operate: Troubleshoot and optimize conversationally

What the agent does

- Diagnose operational issues
- Analyze costs and recommend savings
- Plan and validate upgrades
- Generate runbooks and documentation

Real-time capabilities

- Read pod status and events
- Query CloudWatch logs
- Inspect cluster configuration
- Check EKS Insights for issues

EXAMPLE PROMPTS

"Why are my pods stuck in Pending?"

"What cost optimizations can we make?"

"Prepare for EKS 1.31 upgrade"

MCP TOOLS IN ACTION

list_k8s_resources: Check pod status

get_cloudwatch_logs: Retrieve container logs

get_eks_insights: Identify cluster issues

OUTPUT

Operational runbooks + optimization actions

DEMO

Thank you!