INDEPENDENT HANDS-ON PROJECT

Private Amazon EKS Platform Foundation

Private cluster access, managed nodes, KMS and controlled upgrades

Portfolio track	Independent Hands-on Engineering
Evidence level	Terraform fmt, init and validate passed
Author	Saurabh Dindokar

Evidence boundary: This is a public-safe personal implementation lab. It is not client production work, and no live deployment is claimed.

Overview

A Terraform foundation for a private-endpoint Amazon EKS cluster with managed nodes, KMS envelope encryption, control-plane logging, core managed add-ons and controlled rolling node updates.

Engineering Problem

Creating an EKS control plane is only the start of a platform. The foundation must define private access, node lifecycle, secrets encryption, audit visibility and a clear boundary between cluster creation and separately reviewed platform add-ons.

Architecture

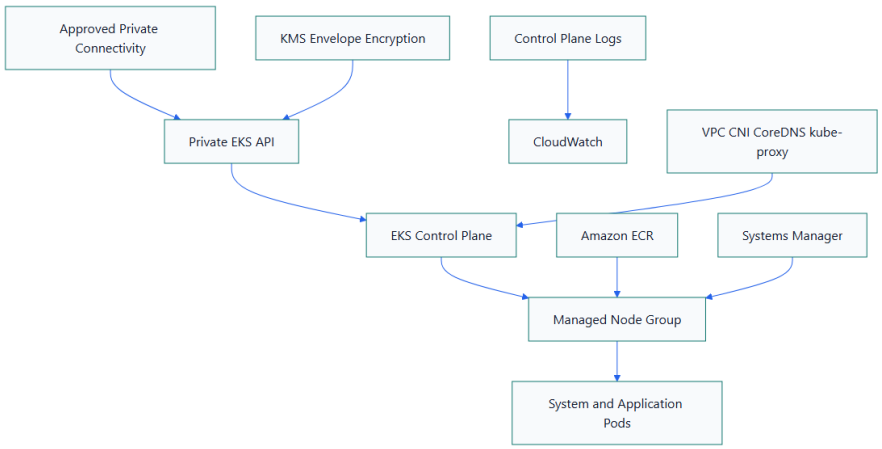

Figure 1. Private Amazon EKS Platform Foundation architecture

What I Implemented

- Configured a private EKS API endpoint that requires approved connectivity inside the VPC.
- Added a managed node group in private subnets with controlled rolling update settings.

- Enabled KMS envelope encryption for Kubernetes secrets and EKS control-plane logging.
- Managed core add-ons explicitly rather than treating their lifecycle as invisible.
- Documented the next platform layers: workload identity, ingress, DNS, certificates, autoscaling, policy, observability, backup and GitOps.

Important Technical Decisions

- Private API access reduces internet exposure but requires VPN, Direct Connect or a controlled in-VPC runner.
- Cluster creation and add-on lifecycle are separated so each platform capability can be reviewed and upgraded deliberately.
- Node replacement is treated as an operational change with drain and skew checks.

Security Controls

- The Kubernetes API endpoint is private.
- KMS protects Kubernetes secret data at rest.
- IAM, security groups and private worker subnets form explicit access boundaries.

Reliability and Operations

- Managed node rolling updates limit disruption.
- Control-plane logs support investigation of API, audit and authentication activity.
- Core add-on health and Kubernetes version skew are part of the upgrade runbook.

Cost and Cleanup Guardrails

- EKS control-plane and NAT Gateway charges continue while idle.
- The runbook requires cleanup checks for workload load balancers, EBS volumes, ENIs, NAT, EIPs and log groups.
- Optional platform add-ons should be selected with operational value and cost in mind.

Validation Evidence

The following local checks passed on 2026-08-13:

```
terraform fmt -check -recursive
terraform init -backend=false
terraform validate
```

Scope boundary: Format, initialization and validation passed on 2026-08-13. No plan, cluster deployment, pod scheduling, node drain or upgrade test is claimed. Ingress, GitOps, policy and observability add-ons are intentionally outside this foundation.

Delivery and Verification Runbook

- 1. Review IAM, KMS and private connectivity
- 2. Review Terraform plan
- 3. Create control plane and managed nodes

- 4. Verify logs and core add-ons
- 5. Test scheduling and node drain
- 6. Plan version upgrade and rollback
- 7. Destroy and inspect residual cloud resources

Technology Stack

Terraform | Amazon EKS | Kubernetes | Managed Node Groups | AWS KMS | IAM | CloudWatch | VPC CNI | CoreDNS

Key Learnings

- Infrastructure evidence must distinguish code validation, plan review and live deployment.
- Security, reliability, cost and cleanup decisions should be documented before apply.
- A useful platform lab includes verification and rollback thinking, not only resource declarations.