zsoftly

Enterprise EKS Auto Mode Migration Roadmap

From On-Premises to Fully Managed Kubernetes with EKS Auto Mode

A Comprehensive Guide by ZSoftly Technologies Inc.

December 2025

Notices

This document is provided for informational purposes only. It represents ZSoftly Technologies Inc.'s current product offerings and practices as of the date of publication, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of ZSoftly's services.

This document does not create any warranties, representations, contractual commitments, conditions, or assurances from ZSoftly Technologies Inc., its affiliates, suppliers, or licensors.

Trademarks

- Amazon Web Services, AWS, Amazon EKS, and related services are trademarks of Amazon.com, Inc. or its affiliates.
- Kubernetes is a registered trademark of The Linux Foundation.
- SigNoz, Falco, Trivy, and ArgoCD are open-source projects under their respective licenses.
- All other trademarks are the property of their respective owners.

Third-Party Content

This document references third-party open-source tools (SigNoz, Falco, Trivy, ArgoCD). ZSoftly Technologies Inc. is not responsible for the content, accuracy, or functionality of these third-party tools. Users should review the respective project documentation and licenses.

Table of Contents

Executive Summary

Organizations running Kubernetes on-premises face real challenges: aging infrastructure, scaling limitations, security compliance burdens, and the constant operational overhead of cluster management. This guide provides a proven roadmap for migrating to **Amazon EKS Auto Mode**. You get fully managed infrastructure, automated node provisioning, and production-ready security with open-source tooling.

Why EKS Auto Mode?

EKS Auto Mode changes how you manage Kubernetes. AWS now manages **Karpenter, ALB Controller, and EBS CSI Driver off-cluster**, removing the operational burden of maintaining these critical components. Combined with **Bottlerocket OS** and **open-source observability and security tools**, you get enterprise capabilities without vendor lock-in.

Key Benefits

Benefit	Impact
Control Plane Management	Karpenter, ALB Controller, EBS CSI managed by AWS off-cluster
Node Provisioning	30-60 seconds with automatic GPU detection and repair
Operating System	Bottlerocket with SELinux, read-only root filesystem
Observability	SigNoz (open-source APM) with OpenTelemetry native support
Runtime Security	Falco (CNCF graduated) + Trivy for image scanning
GitOps	EKS Capability for ArgoCD (fully managed)
Disaster Recovery	Cross-region failover with 30-minute RTO

Target Outcomes

- **Infrastructure as Code**: 100% coverage
- **Deployment Frequency**: Multiple deployments per day
- **Change Failure Rate**: < 5%
- **Mean Time to Recovery**: < 30 minutes
- **Vendor Lock-in**: Minimized with open-source tooling

Who Should Read This Guide

- **CTOs & VP Engineering**: Strategic business case and ROI framework

- **Platform Engineers**: Technical architecture and implementation details
- **DevOps Teams**: GitOps workflows and operational procedures
- **Security Teams**: Open-source security stack and compliance

---

The Challenge

On-Premises Limitations

Organizations operating Kubernetes on-premises commonly face:

Infrastructure Constraints

- Hardware refresh cycles creating technical debt
- Limited capacity for sudden scaling demands
- High capital expenditure for peak capacity planning
- Data center operational overhead

Operational Burden

- Kubernetes version upgrades requiring significant planning
- Node patching and security remediation delays
- Managing Karpenter, Ingress Controllers, CSI drivers
- Limited observability across distributed workloads

Vendor Lock-in Concerns

- Expensive proprietary APM tools (Dynatrace, Datadog)
- Costly runtime security solutions (CrowdStrike, Sysdig)
- Limited flexibility for multi-cloud strategies

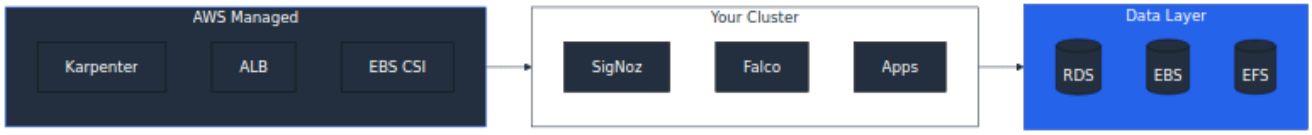
Business Impact of Current State

Challenge	Business Impact
Slow deployments	Delayed time-to-market for new features
Manual scaling	Missed SLAs during demand spikes
Expensive tooling	High operational costs, vendor dependency
Security gaps	Increased risk exposure and compliance failures

Solution Architecture Overview

EKS Auto Mode: What AWS Manages

With EKS Auto Mode, critical Kubernetes components run **off-cluster in the AWS control plane**:



EKS Auto Mode Managed Components:

Component	Managed by AWS	Notes
Karpenter	Yes	Node provisioning, off-cluster
AWS Load Balancer Controller	Yes	ALB/NLB ingress, off-cluster
EBS CSI Driver	Yes	Persistent volumes, off-cluster
VPC CNI	Yes	Pod networking
CoreDNS	Yes	Cluster DNS
kube-proxy	Yes	Service networking
ArgoCD	Optional	EKS Capability (opt-in)
Node Auto Repair	Yes	10-min GPU failure detection

Self-Managed Components (deploy via ArgoCD):

Component	Purpose	Enterprise Alternative
External DNS	Route53 DNS records	-
Cert Manager	TLS certificate automation	-
SigNoz	APM, logs, traces	Datadog, Dynatrace
Falco	Runtime security	CrowdStrike
Trivy	Image vulnerability scanning	Prisma Cloud, Snyk

Data Persistence (Outside Cluster):

Service	Use Case	Backup
RDS	Relational databases (PostgreSQL, MySQL)	Automated snapshots
EBS	Block storage for stateful pods	EBS snapshots
EFS	Shared file storage across pods	AWS Backup

Multi-Account Strategy

Account	Purpose	Environments	Regions
Non-Prod	Development and testing	dev, qat	ca-central-1
Prod	Staging and production	stg, prod	ca-central-1, ca-west-1 (DR)

Bottlerocket: Secure Node Operating System

EKS Auto Mode uses **Bottlerocket** exclusively. This purpose-built Linux OS for containers provides:

Feature	Benefit
Read-only root filesystem	Immutable infrastructure, prevents tampering
SELinux mandatory access controls	Enhanced process isolation
No SSH/SSM access	Reduced attack surface
Automatic security updates	API-driven updates, no manual patching
Minimal footprint	Faster boot times, smaller attack surface

Built-in NodePools

EKS Auto Mode provides two built-in NodePools:

NodePool	Purpose	Taints
`system`	Cluster-critical applications	`CriticalAddonsOnly`
`general-purpose`	Standard workloads	None
Custom	GPU, Spot, specialized workloads	User-defined

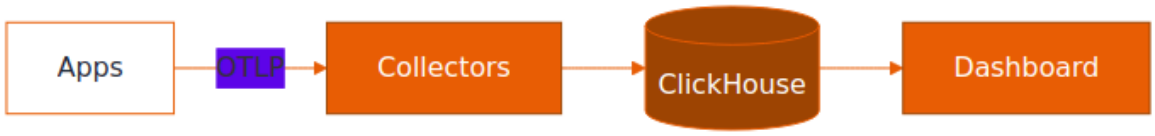
Open-Source Technology Stack

Observability: SigNoz

Why SigNoz over Datadog/Dynatrace?

Feature	SigNoz	Proprietary APM
Cost	Open-source (self-hosted)	$$$$ per host/container
OpenTelemetry	Native support	Varies, often proprietary agents
Vendor lock-in	None	High
Data ownership	Full control	Cloud-only
Kubernetes native	K8s-Infra Helm chart	Yes, but expensive
Real-time tracing	Yes (OTLP protocol)	Yes

SigNoz Architecture:



K8s-Infra Helm Chart Capabilities:

- Tails and parses container logs in real-time
- Gathers distributed traces from Kubernetes workloads
- Collects kubelet and host metrics
- Gathers cluster-level metrics from API server
- Out-of-the-box Kubernetes dashboards

Enterprise Alternatives:

For organizations requiring vendor-supported solutions:

- **Datadog** - Full-featured APM with strong dashboards, per-host licensing (~$30-50/host/mo)
- **Dynatrace** - AI-powered root cause analysis, per-host licensing (~$30-50/host/mo)

Security: Falco + Trivy

Why Falco over CrowdStrike?

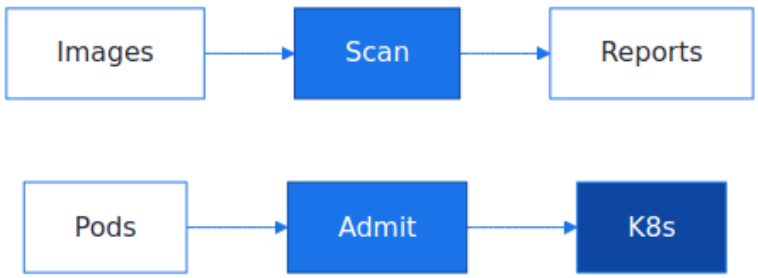
Feature	Falco	CrowdStrike
Cost	Open-source	$$$$ per endpoint
CNCF Status	Graduated project	Proprietary
Linux/K8s focus	Native	Endpoint-first
Deployment	Self-hosted	Cloud-only SaaS
Air-gapped	Supported	Not supported
Customization	Full rule control	Black box
eBPF support	Native	Yes

Falco Runtime Security Architecture:

Falco Detection Capabilities:

- Container escape attempts
- Privilege escalation
- Cryptomining detection
- Unexpected network connections
- File system tampering
- Kubernetes audit events

Trivy Security Scanning:



Enterprise Alternatives:

For organizations requiring vendor-supported security solutions:

- **CrowdStrike Falcon** - Industry-leading EDR/XDR with Kubernetes support, managed threat intelligence, 24/7 SOC (~$20-40/node/mo)
- **Prisma Cloud** - Palo Alto's CNAPP platform, image scanning + runtime protection
- **Sysdig Secure** - Kubernetes-native security with Falco compatibility

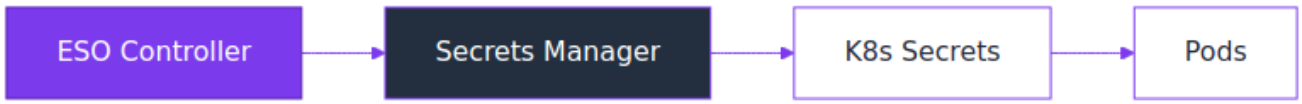
Secrets Management: External Secrets Operator

Why External Secrets Operator (ESO)?

Kubernetes Secrets stored in Git (even encrypted) create security and operational challenges. External Secrets Operator solves this by syncing secrets from AWS Secrets Manager directly into Kubernetes, keeping sensitive data out of your repositories.

Feature	Native K8s Secrets	External Secrets Operator
Secret Storage	etcd (in cluster)	AWS Secrets Manager
Git Repository	Secrets in Git	Only references in Git
Rotation	Manual redeploy	Automatic sync on change
Audit Trail	Limited	CloudTrail integration
Cross-Environment	Copy/paste	Same secret, different permissions
Encryption at Rest	KMS (cluster)	KMS (AWS-managed)

ESO Architecture:



How It Works:

1. **Store secrets in AWS Secrets Manager** - Database credentials, API keys, certificates
2. **Create ExternalSecret resources** - Reference the secret path in Secrets Manager
3. **ESO syncs automatically** - Controller creates/updates Kubernetes Secrets
4. **Pods consume normally** - Mount as environment variables or volumes

Secret Organization Strategy:

Environment	Secrets Manager Path	Access
dev	`/eks/dev/app-name/secret`	Dev cluster IAM role
qat	`/eks/qat/app-name/secret`	QAT cluster IAM role
stg	`/eks/stg/app-name/secret`	Prod account, STG role
prod	`/eks/prod/app-name/secret`	Prod account, Prod role

Benefits:

- **Zero secrets in Git** - Only ExternalSecret manifests referencing paths
- **Automatic rotation** - Update Secrets Manager, ESO syncs within minutes
- **Least-privilege access** - Each cluster only accesses its environment's secrets
- **Audit compliance** - CloudTrail logs all secret access

Implementation Roadmap

Timeline (5-6 Months)

Phase	Description	Duration
1	AWS Foundation	Weeks 1-2
2	EKS Auto Mode Clusters	Weeks 3-4
3	GitOps with ArgoCD	Weeks 4-5
4	Open-Source Observability (SigNoz)	Weeks 5-6
5	Open-Source Security (Falco + Trivy)	Weeks 6-7
6	Metrics Server & HPA	Weeks 7-8
7	CI/CD Pipeline	Weeks 8-9
8	Application Migration (5 Apps)	Weeks 9-16
9	Disaster Recovery	Weeks 16-18
10	Handover & Training	Weeks 18-20

Base Duration: 20 weeks (5 months) for 5 applications. Additional apps add ~1-2 weeks each, extending to 24+ weeks for larger portfolios.

Phase 1: AWS Foundation (Weeks 1-2)

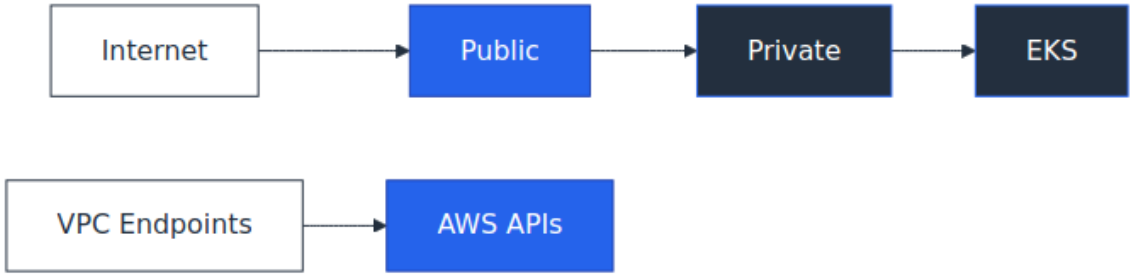
Objectives:

- Establish multi-account structure
- Configure networking foundation
- Set up IAM roles with least-privilege policies

Key Deliverables:

- AWS Organizations setup with Service Control Policies (SCPs)
- VPC per environment with proper CIDR planning
- VPC Endpoints for private AWS API access
- Cross-account IAM roles for deployments
- KMS keys for EBS encryption

Network Architecture:



Secure Access to Private Cluster:

The EKS cluster uses **private endpoints only**—the Kubernetes API server is not exposed to the public internet. Developers and CI/CD pipelines access the cluster through secure VPN connectivity.

VPN Options:

Solution	Type	Best For
WireGuard	Open-source	Cost-conscious, self-managed, high performance
Tailscale	SaaS (WireGuard-based)	Easy setup, mesh networking, SSO integration
Zscaler ZPA	Enterprise SaaS	Zero trust, compliance, identity-aware access
Palo Alto GlobalProtect	Enterprise SaaS	Existing Palo Alto customers
AWS Client VPN	AWS Native	AWS-only environments

Open-Source Recommendation: WireGuard

- Kernel-level performance (faster than OpenVPN)
- Simple configuration with public key cryptography
- Deploy on EC2 instance or container in VPC
- No licensing costs

Enterprise Recommendation: Zscaler ZPA

- Zero trust network access (ZTNA)
- Identity-aware policies (Okta, Azure AD integration)
- No inbound firewall rules required
- Compliance-ready (SOC 2, HIPAA)

Phase 2: EKS Auto Mode Clusters (Weeks 3-4)

Objectives:

- Deploy EKS Auto Mode clusters
- Configure built-in NodePools
- Enable managed add-ons

What EKS Auto Mode Provides:

Component	Status	Notes
Karpenter	Managed off-cluster	No deployment needed
ALB Controller	Managed off-cluster	No deployment needed
EBS CSI Driver	Managed off-cluster	No deployment needed
Node Auto Repair	Built-in	10-minute GPU failure detection
Bottlerocket AMI	Automatic	No AMI pipeline needed

Cluster Configuration:

Setting	Value
Kubernetes Version	1.31, 1.32, 1.33 (recommended), 1.34
Compute Mode	Auto Mode
Authentication	EKS Access Entries
Endpoint Access	Private + Public (restricted)
Logging	API, Audit, Authenticator
Encryption	Secrets encrypted with KMS

NodePool Configuration:

Custom NodePools can be created for specialized workloads such as GPU instances (g5.xlarge, g5.2xlarge, p4d.24xlarge) with on-demand capacity type, CPU limits of 1000, and consolidation policies.

Deliverables:

- EKS Auto Mode cluster per environment
- Custom NodePools for GPU/specialized workloads

- CloudWatch log groups configured
- EKS access entries for admin roles

Phase 3: GitOps with ArgoCD (Weeks 4-5)

Objectives:

- Deploy ArgoCD (managed or self-hosted)
- Implement app-of-apps pattern
- Configure environment-specific deployments

Option A: EKS Capability for ArgoCD (Recommended)

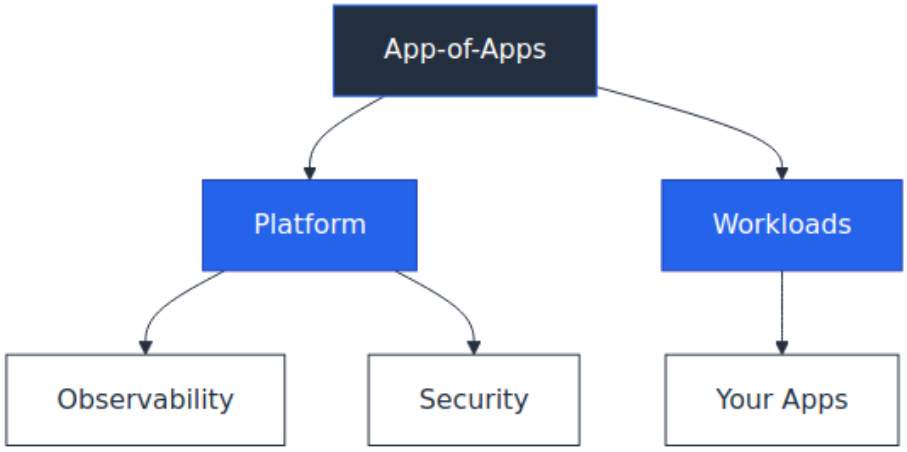
AWS manages ArgoCD in the control plane:

- No ArgoCD pods on worker nodes
- AWS handles Git/Helm registry access
- Automatic scaling and updates

Option B: Self-Hosted ArgoCD

Deploy via Helm for full customization control.

App-of-Apps Pattern:



Phase 4: Open-Source Observability (Weeks 5-6)

Objectives:

- Deploy SigNoz for unified observability
- Configure OpenTelemetry collectors
- Set up dashboards and alerting

SigNoz Deployment via ArgoCD:

SigNoz is deployed using the official Helm chart from charts.signoz.io with AWS cloud configuration, cluster name settings, and OpenTelemetry collector endpoint configuration. The deployment uses automated sync with prune and self-heal enabled.

K8s-Infra Collectors:

The K8s-Infra Helm chart is deployed alongside SigNoz to collect infrastructure metrics, logs, and traces from all Kubernetes workloads with AWS cloud configuration.

Included Capabilities:

- Pod-level CPU/memory usage, restarts, saturation
- P90/P99 latency, request throughput, error rates
- Unified correlation of logs, traces, and metrics
- Custom dashboards for EKS Auto Mode
- Real-time distributed tracing

Enterprise Alternatives:

For organizations requiring vendor-supported APM:

- **Datadog** - Deploy Datadog Agent DaemonSet via Helm
- **Dynatrace** - Deploy OneAgent Operator for auto-instrumentation

---

Phase 5: Open-Source Security (Weeks 6-7)

Objectives:

- Deploy Falco for runtime security
- Deploy Trivy for image scanning
- Configure admission control

Falco Deployment:

Falco is deployed via ArgoCD using the official Helm chart with modern eBPF driver, Falcosidekick for alert routing to SigNoz, and custom rules for detecting shell access in containers.

Trivy Operator Deployment:

Trivy Operator is deployed via ArgoCD to automatically scan all container images in the cluster, reporting CRITICAL and HIGH severity vulnerabilities with ConfigAudit scanning enabled.

Admission Controller (Optional):

Enable webhook-based admission control to block vulnerable images at deploy time with a Fail policy.

Phase 6: Metrics Server & HPA (Weeks 7-8)

Objectives:

- Deploy Metrics Server for resource metrics
- Configure Horizontal Pod Autoscaler (HPA) for data-driven scaling
- Establish scaling policies per application

Why Metrics Server + HPA:

EKS Auto Mode manages node scaling via Karpenter, but **pod scaling** requires Metrics Server and HPA for data-driven autoscaling based on your organization's actual traffic patterns.

Metrics Server Deployment:

Metrics Server is deployed via ArgoCD to collect CPU and memory metrics from kubelets, enabling HPA to make scaling decisions based on real-time resource utilization.

HPA Configuration per Application:

Each application Helm chart includes HPA configuration with:

- Min/max replica counts based on workload requirements
- CPU threshold (e.g., scale at 70% CPU utilization)
- Memory threshold (optional)
- Custom metrics from SigNoz (requests per second, latency)

Scaling Architecture:

Layer	Component	Scaling Trigger
Pods	HPA	CPU/Memory metrics from Metrics Server
Nodes	Karpenter	Pending pods (managed by EKS Auto Mode)

Phase 7: CI/CD Pipeline (Weeks 8-9)

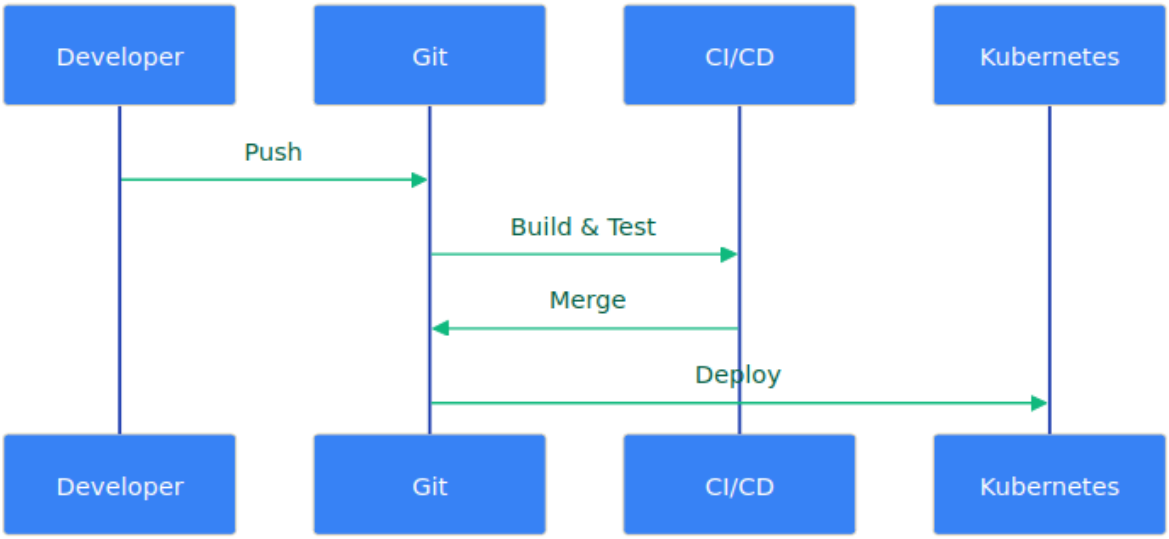
Objectives:

- Configure GitLab/GitHub Actions for infrastructure
- Implement promotion workflows
- Set up security scanning in pipeline
- Configure private ECR registry with cross-region replication

Pipeline Stages:

The CI/CD pipeline consists of four stages: validate (terraform fmt, validate, trivy scan), build (terraform plan), deploy (terraform apply), and security (trivy image scan).

GitOps Deployment Flow:



Private ECR Registry Setup

All container images are stored in **AWS ECR private repositories**, created and managed via Terraform. This ensures consistent image management, security scanning, and cross-region availability for disaster recovery.

ECR Infrastructure (Terraform):

Resource	Purpose	Configuration
ECR Repositories	Private container image storage	One per application/service
Lifecycle Policies	Automatic cleanup of old images	Retain last 30 tagged images
Repository Policies	IAM access control	CI/CD and EKS node access
Replication Rules	Cross-region image sync (prod only)	ca-central-1 → ca-west-1
Image Scanning	Vulnerability detection on push	Enhanced scanning enabled

ECR Cross-Region Replication (Prod):

For production, ECR replication is configured between ca-central-1 (primary) and ca-west-1 (DR). Images pushed to the primary region automatically replicate to the DR region, ensuring container images are available for failover.

CI/CD Image Pipeline:

1. **Build**: Docker image built from Dockerfile
2. **Scan**: Trivy scans image for CRITICAL/HIGH vulnerabilities
3. **Push**: Image pushed to ECR ca-central-1 with semantic version tag
4. **Replicate**: ECR automatically replicates to ca-west-1 (prod only)
5. **Deploy**: ArgoCD detects new image tag, syncs to cluster

IAM Policies:

Principal	Access Level	Purpose
CI/CD Role	Push/Pull	Build and push images during pipeline
EKS Node Role	Pull Only	Nodes pull images for pod deployments
Developer Role	Pull Only	Local development with remote images

Lifecycle Policy:

ECR lifecycle policies automatically clean up untagged images and retain only the most recent tagged versions, reducing storage costs and maintaining a clean registry.

Phase 8: Application Migration (Weeks 9-16)

Objectives:

- Migrate 5 organization applications to EKS via Helm charts
- Implement data-driven autoscaling with HPA
- Deploy via GitOps with environment promotion

Scope: 5 Applications

This phase migrates 5 company applications. Timeline scales with application count:

Applications	Duration	Notes
5 apps	6-7 weeks	Base scope
10 apps	10-12 weeks	+1 week per app
15+ apps	14-18 weeks	Parallel team recommended

Per-Application Helm Chart Includes:

- Deployment with resource requests/limits
- Service and Ingress configuration
- HPA with CPU/memory thresholds
- ConfigMaps and Secrets (via External Secrets Operator)
- Health checks (liveness, readiness, startup probes)
- OpenTelemetry auto-instrumentation for SigNoz
- Network policies

Migration Approach: Strangler Fig Pattern

1. Create Helm chart for application
2. Deploy to dev → validate with SigNoz metrics
3. Promote to qat → load testing, tune HPA thresholds
4. Promote to stg → production-like validation
5. Promote to prod → DNS cutover, monitor
6. Decommission on-prem instance

Per-Application Checklist:

- Containerize application (Dockerfile)
- Create Helm chart with values per environment
- Configure HPA with org-specific scaling thresholds

- Add OpenTelemetry instrumentation
- Validate in dev → qat → stg → prod
- DNS cutover and decommission legacy

Migration Pitfalls & Risk Mitigation

Common challenges encountered during Kubernetes migrations and how to avoid them:

Database Migration Timing:

Pitfall	Impact	Mitigation
Migrating DB before app is ready	Extended downtime, rollback needed	Keep DB on-prem until app validated in EKS
Big-bang database cutover	High risk, long rollback time	Use read replicas, gradual traffic shift
Ignoring connection pool limits	Pod scaling exhausts DB connections	Configure PgBouncer or RDS Proxy

StatefulSet Challenges:

- **Persistent Volume migration** - EBS volumes are AZ-specific; plan for data migration
- **Pod identity** - StatefulSets expect stable network identities; test DNS resolution
- **Ordered scaling** - Pods scale sequentially; factor into HPA response time

DNS & Traffic Cutover:

Issue	Symptom	Solution
High TTL on DNS records	Traffic goes to old infra	Lower TTL to 60s weeks before cutover
No rollback plan	Stuck with broken deployment	Keep on-prem running during validation
Missing health checks	Bad pods receive traffic	Implement readiness probes properly

Rollback Strategy:

Every application migration should have a documented rollback procedure:

1. **Pre-cutover**: On-prem and EKS running in parallel
2. **Cutover**: DNS points to EKS, on-prem remains available
3. **Validation**: 24-48 hours monitoring in production
4. **Decommission**: Only after successful validation period
5. **Emergency rollback**: DNS revert to on-prem (< 5 min)

Resource Sizing Mistakes:

Mistake	Result	Prevention
No resource requests/limits	Noisy neighbor, OOM kills	Always set requests = limits
Copying on-prem sizing	Over-provisioned, wasted spend	Profile in dev/qat, tune in stg
Ignoring memory leaks	Gradual degradation, restarts	Monitor with SigNoz, set limits

Security Oversights:

- **Running as root** - Use `securityContext.runAsNonRoot: true`
- **Missing network policies** - Default allow-all; implement deny-by-default
- **Secrets in environment vars** - Visible in pod describe; use volume mounts
- **No pod security standards** - Enable Pod Security Admission (restricted)

Phase 9: Disaster Recovery (Weeks 16-18)

Objectives:

- Prepare DR infrastructure as code
- Configure cross-region data replication
- Establish failover procedures

DR Strategy: Cold Standby

Only **ca-central-1** runs at all times. The DR region (ca-west-1) uses cold standby—no running compute resources. Infrastructure is defined in Terraform and apps in ArgoCD, ready to deploy on demand. Only the EKS control plane runs continuously to enable rapid cluster bootstrapping during failover.

What's Pre-Configured (Git-Tracked):

Component	Status	Location
Terraform modules	Ready to apply	`terraform/prod/ca-west-1/`
ArgoCD app-of-apps	Ready to sync	`gitops/environments/dr/`
ECR images	Auto-replicated	ca-central-1 → ca-west-1
ECR repositories	Terraform-managed	Created via IaC in both regions
RDS snapshots	Daily replication	Automated cross-region copy
EFS backups	AWS Backup	Cross-region vault

Multi-Region Architecture (Prod Account):

RTO: 30 Minutes

Failover procedure:

1. Detect ca-central-1 failure (Route53 health checks)
2. Run `terraform apply` to provision ca-west-1 compute nodes and networking (~10 min)
3. ArgoCD syncs app-of-apps from Git (~5 min)
4. Restore RDS from latest snapshot (~10 min)
5. Update Route53 to ca-west-1 endpoints
6. Verify application health

Note: EKS control plane already running in DR region enables rapid node provisioning.

RPO: 24 Hours (daily RDS snapshots, continuous ECR replication)

Phase 10: Handover & Training (Weeks 18-20)

Documentation Deliverables:

Document	Purpose
Architecture Guide	System design and components
Operations Guide	Day-to-day procedures
Runbooks	Troubleshooting with SigNoz, Falco alerts
Upgrade Guide	EKS Auto Mode upgrades
DR Playbook	Disaster recovery procedures

Training Topics:

- EKS Auto Mode cluster management
- HPA tuning and scaling policies
- SigNoz dashboards and alerting
- Falco rule customization

- Trivy vulnerability remediation
- ArgoCD GitOps workflows
- Incident response with open-source tools

Investment Framework

Infrastructure Costs (Monthly Estimate)

Assumptions:

- Region: ca-central-1 (primary), ca-west-1 (DR)
- 5 containerized applications
- On-Demand pricing (before Savings Plans)
- Prices as of December 2025

Compute - EC2 Node Pools

Environment	Instance Type	Nodes	vCPU	RAM	$/hr	Monthly
dev	m7i.large	2	2	8 GB	$0.101	$147
qat	m7i.large	2	2	8 GB	$0.101	$147
stg	m7i.xlarge	2	4	16 GB	$0.202	$295
prod	m7i.xlarge	3	4	16 GB	$0.202	$442
prod (HA)	m7i.2xlarge	2	8	32 GB	$0.404	$590
Total		**11**				**$1,621**

DR uses cold standby - no running compute until failover

EKS Control Plane

Cluster	Type	Monthly
dev	EKS Auto Mode	$73
qat	EKS Auto Mode	$73
stg	EKS Auto Mode	$73
prod	EKS Auto Mode	$73
DR	EKS Auto Mode	$73
Total		**$365**

EKS Auto Mode: $0.10/hr per cluster ($0.10 × 730 hrs = $73/cluster/month)

Storage - EBS Volumes

Component	Size	Type	$/GB/mo	Monthly
Node root volumes	11×50 GB	gp3	$0.096	$53
App PVCs (Non-Prod)	200 GB	gp3	$0.096	$19
App PVCs (Prod)	500 GB	gp3	$0.096	$48
SigNoz ClickHouse	500 GB	gp3	$0.096	$48
Total				**$168**

Networking

Component	Quantity	Unit Cost	Monthly
ALB (Non-Prod: dev, qat)	2	$22.50 + LCU	$60
ALB (Prod: stg, prod)	2	$22.50 + LCU	$80
NAT Gateway (Non-Prod)	2	$45 + data	$110
NAT Gateway (Prod)	2	$45 + data	$110
VPC Endpoints (Non-Prod)	6	$7.50 each	$45
VPC Endpoints (Prod)	6	$7.50 each	$45
VPC Endpoints (DR - ECR only)	2	$7.50 each	$15
Data Transfer (inter-AZ)	~500 GB	$0.01/GB	$5
Data Transfer (internet)	~200 GB	$0.09/GB	$18
Total			**$488**

DR uses cold standby - no ALB or NAT Gateway until failover. Only VPC Endpoints for ECR replication.

Observability & Monitoring

Component	Details	Monthly
CloudWatch Logs	50 GB ingestion	$25
CloudWatch Metrics	Custom metrics (100)	$30
SigNoz (self-hosted)	ClickHouse storage only	$0*
Falco (self-hosted)	No additional cost	$0
Total		**$55**

SigNoz storage included in EBS costs above

Cost Summary

Category	Non-Prod	Prod	DR	Total
Compute	$294	$1,327	$0	$1,621
EKS Control	$146	$146	$73	$365
Storage	$65	$103	$0	$168
Networking	$215	$258	$15	$488
Monitoring	$25	$30	$0	$55
Subtotal	**$745**	**$1,864**	**$88**	**$2,697**

Total Monthly: ~$2,697 (On-Demand pricing)

With Compute Savings Plans (1-year, no upfront): **~$1,750/mo** (35% savings)

DR uses cold standby strategy - only EKS control plane ($73) and minimal VPC endpoints ($15) run continuously. Full DR infrastructure deploys on-demand during failover.

Open-Source vs. Proprietary Savings

Tool	Proprietary Cost Basis	Open-Source	Monthly Savings
APM	Datadog: ~$100/host (APM + Infra + profiler) × 11	SigNoz	$1,100
Log Management	Splunk: ~$150/GB/day × 10 GB/day	SigNoz	$1,500
Runtime Security	Sysdig: ~$50/node × 11 nodes	Falco	$550
Image Scanning	Snyk Container: ~$80/developer × 10	Trivy	$800
GitOps	Harness: ~$100/service × 5	ArgoCD	$500
Total			**$4,450/mo**

Annual Savings: ~$53,000 with open-source tooling

Proprietary pricing based on typical enterprise contracts with full feature suites (December 2025). Actual costs vary by vendor negotiation and feature selection.

Cost Optimization Strategies

Beyond open-source tooling savings, EKS Auto Mode enables additional cost optimization through intelligent node management and AWS pricing models.

Spot Instances for Non-Production:

Environment	Instance Strategy	Savings	Risk Level
dev	100% Spot	Up to 90%	Acceptable
qat	80% Spot / 20% OD	Up to 70%	Low
stg	50% Spot / 50% OD	Up to 45%	Very Low
prod	On-Demand + RI	30-40% (RI)	None

Karpenter Consolidation (Managed by EKS Auto Mode):

EKS Auto Mode's managed Karpenter automatically consolidates workloads to reduce node count:

- **Bin packing** - Efficiently schedules pods to maximize node utilization
- **Node consolidation** - Removes underutilized nodes during low-traffic periods
- **Right-sizing** - Selects optimal instance types based on workload requirements
- **Spot interruption handling** - Gracefully migrates pods before Spot termination

Reserved Capacity for Production:

For predictable production workloads, combine On-Demand with Savings Plans:

Commitment Type	Discount	Flexibility	Best For
Compute Savings Plans	Up to 66%	Any instance type	Variable workloads
EC2 Instance Savings	Up to 72%	Specific instance	Stable, predictable loads
Reserved Instances	Up to 75%	Specific instance/AZ	Baseline capacity

Cost Monitoring:

- **AWS Cost Explorer** - Tag-based cost allocation per environment/application
- **Kubecost** (open-source) - Kubernetes-native cost visibility per namespace/workload
- **SigNoz dashboards** - Correlate cost with performance metrics

Estimated Monthly Savings with Optimization:

Strategy	Monthly Savings
Open-source tooling	$4,450
Spot instances (non-prod)	$800
Karpenter consolidation	$400
Savings Plans (prod)	$600
Total Additional Savings	**$6,250/mo**

Success Criteria

Metric	Target
Node Provisioning Time	< 60 seconds (EKS Auto Mode)
Deployment Frequency	Multiple per day
Change Failure Rate	< 5%
Mean Time to Recovery	< 30 minutes
Infrastructure as Code Coverage	100%
Open-Source Tooling	100% for observability/security
Vendor Lock-in	Minimized

Why ZSoftly

Our Expertise

EKS Auto Mode Specialists

- Early adopters of EKS Auto Mode since GA (December 2024)
- Certified Kubernetes Administrators (CKA)
- AWS Container Services expertise

Open-Source Champions

- SigNoz deployment experience
- Falco rule customization experts
- OpenTelemetry implementation specialists

Canadian-Based Team

- Local presence for Canadian enterprises
- Understanding of Canadian compliance requirements
- Responsive support in your timezone

Our Services

Service	Description
EKS Auto Mode Migration	Full migration from on-prem or legacy EKS
Open-Source Observability	SigNoz deployment and configuration
Open-Source Security	Falco + Trivy implementation
GitOps Implementation	ArgoCD setup and app-of-apps patterns
Training & Support	Team enablement and ongoing support

Learn more: zsoftly.com/services/containers

Next Steps

1. Discovery Workshop

Schedule a discovery session to review your current architecture and migration requirements.

2. Assessment

Receive a detailed assessment of your EKS Auto Mode readiness and recommended approach.

3. Proof of Concept

Deploy a development cluster with SigNoz, Falco, and sample application.

4. Engagement

Finalize scope and begin your migration journey.

Ready to Start Your EKS Migration?

Our team of AWS-certified Kubernetes specialists is ready to help you plan and execute your migration to EKS Auto Mode.

Get a Free Assessment:

- Review your current architecture
- Identify migration complexity
- Estimate timeline and costs
- Receive a customized roadmap

Book a Call: zsoftly.com/book-call

Contact Us

ZSoftly Technologies Inc.

Contact	Link
Website	zsoftly.com
Container Services	zsoftly.com/services/containers
Contact Form	zsoftly.com/contact
Book a Call	zsoftly.com/book-call

Appendix: Compliance Control Mapping

This architecture supports common compliance frameworks. The table below maps key controls to specific components.

SOC 2 Type II Controls

Control Area	Requirement	Implementation
CC6.1 - Access	Logical access controls	EKS Access Entries, IAM roles, RBAC
CC6.2 - Authentication	User authentication	AWS SSO/Okta integration, OIDC
CC6.3 - Authorization	Role-based access	Kubernetes RBAC, namespace isolation
CC6.6 - Boundaries	System boundaries protected	VPC, Security Groups, Network Policies
CC6.7 - Data Transfer	Encrypted data transmission	TLS 1.3 (ALB), mTLS (service mesh optional)
CC6.8 - Malware	Malware prevention	Falco runtime detection, Trivy image scanning
CC7.1 - Monitoring	Security event detection	Falco alerts, CloudWatch, SigNoz
CC7.2 - Anomalies	Anomaly identification	Falco behavioral rules, SigNoz alerting

PCI-DSS v4.0 Requirements

Requirement	Description	Implementation
1.4	Network segmentation	VPC subnets, Network Policies, namespace isolation
2.2	Secure configurations	Bottlerocket hardened OS, Pod Security Standards
3.5	Protect stored data	KMS encryption (EBS, Secrets Manager, RDS)
4.1	Encrypt transmissions	TLS everywhere, ALB HTTPS, encrypted EBS
5.2	Anti-malware	Falco runtime security, Trivy vulnerability scan
6.3	Secure development	GitOps review process, automated security scanning
8.3	Strong authentication	AWS SSO, MFA enforcement, short-lived credentials
10.2	Audit logging	CloudTrail, EKS audit logs, Falco events
11.5	Change detection	ArgoCD drift detection, Falco file integrity

HIPAA Technical Safeguards

Safeguard	Requirement	Implementation
Access Control	Unique user identification	IAM users, OIDC identity, audit trails
Audit Controls	Record system activity	CloudTrail, EKS audit logs, SigNoz traces
Integrity Controls	Data integrity mechanisms	Falco file monitoring, Git-based IaC
Transmission Security	Encrypted PHI transmission	TLS 1.3, VPN for cluster access

Shared Responsibility Note

Layer	AWS Responsibility	Your Responsibility
Control Plane	EKS control plane security	Access policies, audit log review
Node Security	Bottlerocket OS patches	Pod security policies, runtime monitoring
Network	VPC infrastructure	Security groups, network policies
Data	Encryption infrastructure	Key management, data classification

Sources

- EKS Auto Mode Best Practices
- Under the Hood: Amazon EKS Auto Mode
- EKS Auto Mode Security Overview
- SigNoz Kubernetes Observability
- Falco Cloud Native Runtime Security
- Trivy Security Scanner
- EKS Blueprints with Auto Mode

Document Version: 1.0 - December 2025

zsoftly.com | Contact Us