Amazon EKS Platform Architect Fieldbook

Amazon EKS Platform Architect Fieldbook

> A simple, production-focused guide using the Feynman technique: learn an idea, explain it plainly, apply it, and expose what you do not yet understand.

\newpage

00. The Amazon EKS platform architect role

Amazon Elastic Kubernetes Service (Amazon EKS) is AWS-managed Kubernetes. An EKS platform architect turns business needs into a safe, reliable container platform that development teams can actually use.

Explain it like I am five

Imagine an application is a busy restaurant. Containers are lunch boxes holding one small part of the meal. Kubernetes is the restaurant manager: it places the boxes, replaces broken ones, and opens more serving lines when a crowd arrives. EKS means AWS runs the manager's control room. You still decide who enters, where the boxes run, how they communicate, and what happens during a fire.

The architect's loop

1. Ask about users, business outcomes, data, scale, security, recovery, and cost.
2. Decide whether Kubernetes is justified; sometimes ECS, Lambda, or App Runner is simpler.
3. Choose the responsibility boundary: Auto Mode, managed node groups, Fargate, or self-managed capacity.
4. Design accounts, identity, networks, workloads, data, delivery, and operations.
5. Record trade-offs and rejected alternatives.
6. Prove the design with load tests, failure drills, restore tests, and cost data.

Evidence a strong architect produces

- A context and traffic-flow diagram.
- A compute-mode decision with reasons.
- IAM, EKS access entry, Kubernetes RBAC, and Pod Identity design.
- VPC, subnet, IP-capacity, endpoint, and network-policy plan.
- Resource, autoscaling, disruption, and topology policy.
- Source-to-ECR-to-admission supply-chain controls.
- SLOs, dashboards, alerts, runbooks, and an upgrade calendar.
- Tested RTO, RPO, backup, and regional recovery.
- Unit cost such as cost per order—not only a monthly invoice.

Feynman check

Can you explain who manages the control plane, who manages worker capacity, and who secures the application without saying “AWS handles everything”? If yes, you understand the first responsibility boundary.

\newpage

01. Kubernetes in simple terms

Kubernetes is a continuous correction machine. You declare **desired state**. Controllers compare it with **actual state** and keep trying to close the gap.

The smallest useful vocabulary

Object	Plain meaning
Pod	Smallest scheduled unit; normally one application container
Deployment	Keeps stateless Pods running and rolls out versions
StatefulSet	Stable names and ordered behavior for stateful Pods
DaemonSet	Runs one Pod on selected nodes
Job / CronJob	Finite or scheduled work
Service	Stable address for changing Pods
Gateway / Ingress	Routes application traffic
Namespace	Scope for names, quota, RBAC, and policy
ConfigMap / Secret	Configuration and sensitive values

Labels identify objects. Selectors choose objects. A wrong selector can send traffic to the wrong Pods or leave a controller managing nothing.

Requests, limits, and health

A resource **request** is what the scheduler reserves. It drives placement, capacity, and cost. A **limit** is an upper boundary. Measure both; guesses create waste, CPU throttling, or memory kills.

- Startup probe: “I am still starting.”
- Readiness probe: “Do not send me traffic yet.”
- Liveness probe: “I am stuck and a restart can fix me.”

Do not make liveness depend on a slow database. Restarting every Pod during a database incident makes the incident larger.

Feynman check

If three of five replicas disappear, a Deployment sees that actual state is smaller than desired state and creates three replacements. Kubernetes restores objects; it does not restore lost business data.

\newpage

02. How Amazon EKS works

An EKS cluster has an AWS-managed Kubernetes **control plane** and customer workloads running on worker compute.

The two halves

The control plane contains Kubernetes API servers and the cluster state store. AWS runs this across multiple Availability Zones. You pay a per-cluster fee and choose a supported Kubernetes version.

Worker compute can be EKS Auto Mode managed instances, EC2 managed node groups, AWS Fargate, or self-managed EC2. Kubernetes schedules Pods onto suitable capacity.

Shared responsibility in one sentence

AWS operates the EKS service and control-plane infrastructure; you own access, workload code and images, data, configuration, availability choices, and incident response. Your node responsibility changes with the compute model.

AWS building blocks around EKS

- VPC, subnets, route tables, security groups, endpoints, NAT, and DNS.
- IAM, EKS access entries, Kubernetes RBAC, and EKS Pod Identity.
- ECR, Inspector, KMS, Secrets Manager, GuardDuty, and CloudTrail.
- ALB/NLB, Route 53, CloudFront, WAF, and ACM.
- EBS, EFS, FSx, RDS/Aurora, DynamoDB, and ElastiCache.
- CloudWatch, ADOT, Amazon Managed Service for Prometheus, and Grafana.

One cluster or many?

Use business and security boundaries—not fashion. Separate clusters when a failure, trust boundary, region, lifecycle, or team autonomy requires it. Every extra cluster adds upgrades, add-ons, policy, delivery, and observability work.

Feynman check

EKS removes the job of building and repairing the Kubernetes control plane. It does not remove application architecture or platform engineering.

\newpage

03. Choose the EKS compute model

Choose the least operational responsibility that still satisfies the real requirement.

EKS Auto Mode

Auto Mode manages compute provisioning and scaling, Pod networking, network policy, load balancing, and block-storage integration. Its managed instances use Bottlerocket and are deliberately restricted. Start here for a new general platform when the supported model fits.

Managed node groups

AWS manages EC2 node-group lifecycle, update, drain, and replacement. You still choose AMIs, instance types, capacity types, labels, taints, scaling, add-ons, and upgrade timing. Use them when you need ordinary EC2-node control, custom agents, GPUs, or predictable groups.

Fargate

Each eligible Pod gets isolated serverless compute. There are no nodes to manage. It is useful for compatible bursty or isolated workloads, but DaemonSets, privileged containers, some storage/network features, and detailed placement needs can rule it out. Price it against the workload shape.

Karpenter and self-managed nodes

Karpenter provisions EC2 capacity directly from Pod requirements and can consolidate nodes. Auto Mode uses a managed Karpenter-based approach. Run your own Karpenter when you need its flexibility outside Auto Mode. Fully self-managed nodes carry the most patching and lifecycle responsibility.

Need	Sensible starting point
General workloads, minimal node operations	Auto Mode
Host agent, custom AMI, or controlled node groups	Managed node groups
Compatible Pods with per-Pod isolation	Fargate
Flexible EC2 provisioning with team-operated controller	Karpenter
Unusual node lifecycle requirement	Self-managed, only if justified

Feynman check

More control is not free. Name the exact capability that pays for every extra patch, upgrade, scaling, and on-call duty.

\newpage

04. Accounts, access, and workload identity

Identity has four connected layers: the AWS organization, IAM, EKS cluster access, and Kubernetes RBAC.

People

Federate people through IAM Identity Center. Use short-lived role sessions and groups. Avoid permanent IAM users and access keys.

EKS **access entries** associate an IAM principal with cluster access. EKS access policies can grant common permissions, while Kubernetes RBAC handles fine-grained namespaced roles. Keep platform administration separate from application deployment. Remove emergency access after the incident.

Pods

Use **EKS Pod Identity** or IAM Roles for Service Accounts (IRSA) to give a Kubernetes service account short-lived AWS credentials. Use a dedicated service account per responsibility. Do not put access keys in images, ConfigMaps, or Secrets.

Pod Identity uses the EKS Auth API and an agent on nodes; applications need a current AWS SDK. IRSA uses a cluster OIDC provider and web-identity federation. Choose a standard and understand its prerequisites.

Governance

- Separate production, non-production, security, networking, and log archive accounts where the organization needs those boundaries.
- Apply Service Control Policies as guardrails, not workload permissions.
- Centralize CloudTrail and AWS Config evidence in protected accounts.
- Use permission boundaries where delegated role creation needs a ceiling.
- Treat namespace isolation as useful but not automatically hostile-tenant safe.

Feynman check

IAM answers “may this AWS identity call this AWS API?” RBAC answers “may this authenticated identity change this Kubernetes object?” A secure design needs both answers.

\newpage

05. VPC networking and isolation

EKS networking connects Pods, nodes, load balancers, AWS services, the internet, and private data centers.

Address planning

With the Amazon VPC CNI, Pods normally receive VPC IP addresses. Estimate Pods per node, ENI limits, subnets, growth, peering, hybrid networks, and future clusters. Prefix delegation, IPv6, or secondary CIDRs can improve scale, but they do not excuse late planning.

Use at least two Availability Zones. Size private subnets for nodes and Pods and public subnets only for internet-facing load balancers when required.

Private clusters and egress

Restrict the Kubernetes API endpoint using private access and controlled public CIDRs, or private-only access when operations paths support it. Private nodes need designed routes to ECR, S3, STS, CloudWatch, and other services through VPC endpoints or NAT. “Private” does not mean “no dependencies.”

Three policy layers

1. Security groups control ENI-level flows.
2. Security Groups for Pods can give selected Pods AWS security-group rules.
3. Kubernetes NetworkPolicy controls Pod communication when an enforcing implementation is enabled. Start default-deny and allow required DNS, ingress, egress, identity, metrics, and data paths.

Troubleshooting order

Check DNS, Service/endpoints, NetworkPolicy, security groups, NACLs, routes, subnet IPs, NAT/endpoints, load-balancer health, and application listeners.

Feynman check

A Pod IP is like an apartment number in the VPC. The Service is the permanent reception desk. Network policies and security groups decide which visitors can reach which doors.

\newpage

06. Design workloads that survive

Kubernetes can replace a Pod only if the application is designed to be replaceable.

The safe workload recipe

- Keep web and worker compute stateless; externalize durable state.
- Use immutable image digests, not mutable `latest` tags.
- Run as non-root, drop capabilities, use seccomp, and prefer read-only filesystems.
- Set measured requests and intentional limits.
- Add startup, readiness, and careful liveness probes.
- Handle termination signals and allow enough graceful shutdown time.
- Use rolling, canary, or blue/green delivery with rollback evidence.
- Spread replicas across zones and nodes.
- Use a PodDisruptionBudget that protects service without blocking every drain.

Configuration and secrets

Keep ordinary environment configuration in ConfigMaps or external configuration. Store sensitive values in Secrets Manager or Parameter Store and retrieve them through a controlled integration. Kubernetes Secret values are base64-encoded, not magically encrypted by that encoding.

Back-pressure

Queues such as SQS separate bursty intake from slower processing. Workers must be idempotent because retries and duplicate delivery happen. Bound retries, send poison messages to a dead-letter queue, and protect downstream systems with timeouts and concurrency limits.

Feynman check

If deleting any one Pod causes lost orders, forced user logout, or manual repair, the application—not EKS—is holding essential state in the wrong place.

\newpage

07. Scheduling and autoscaling

Scaling is several feedback loops, not one magic switch.

Pod placement

The scheduler compares requests with available capacity. Node selectors and affinity attract Pods. Taints repel Pods unless they tolerate them. Topology spread distributes replicas across failure domains. Hard rules can protect isolation but also make Pods impossible to schedule.

Pod scaling

The Horizontal Pod Autoscaler changes replica count from CPU, memory, or custom metrics. Choose a metric that follows demand. Set minimums for availability and maximums from downstream capacity. The Vertical Pod Autoscaler helps right-size requests but may restart Pods in automatic modes.

Node scaling

Cluster Autoscaler changes managed node-group size. Karpenter selects and launches EC2 instances from Pod requirements. Auto Mode manages a Karpenter-based capacity layer through NodePools and NodeClasses. Pod scaling and node scaling are separate: new Pods can remain Pending while capacity arrives.

Capacity choices

Use On-Demand for reliable baseline capacity. Use Spot for replicated, checkpointed, queue-driven, or retryable work. Diversify instance types and zones. Graviton can improve price-performance when images support arm64.

Feynman check

Requests are the shopping list the scheduler reads. If the list is wrong, no autoscaler can buy the right shelves.

\newpage

08. Storage and data

Choose data services from access pattern, consistency, availability, recovery, and operations—not from what fits inside a Pod.

Kubernetes storage

A PersistentVolumeClaim asks for storage through a StorageClass and CSI driver. EBS provides block storage, usually tied to an Availability Zone. EFS provides regional shared NFS access. FSx provides managed file systems for specialized needs. Topology affects where a Pod can restart.

Managed data is often safer

Use RDS or Aurora for relational transactions, DynamoDB for known key-value access at large scale, ElastiCache for cache/session acceleration, S3 for objects, and OpenSearch for supported search/analytics use cases. Keeping a database outside Kubernetes removes database operations from the cluster but does not remove connection, backup, scaling, or failover design.

Backup and recovery

Replication keeps service available; backup lets you return to an earlier good state. Protect Kubernetes manifests, persistent data, ECR images, secrets, infrastructure code, DNS, and dependency configuration. AWS Backup and EBS snapshots cover eligible resources; tools such as Velero can protect Kubernetes objects. Always run restore tests.

RPO is tolerated data loss in time. RTO is tolerated restoration time. Write both before choosing the technology.

Feynman check

Three copies of corrupted data are three corrupted copies. Replication is not a time machine; backup and point-in-time recovery are.

\newpage

09. EKS security architecture

Security is a chain from human identity to build, registry, admission, runtime, network, data, and evidence.

Prevent

- Use federated short-lived human access and least-privilege RBAC.
- Give Pods scoped identity with Pod Identity or IRSA.
- Keep nodes private and restrict the API endpoint.
- Enforce Pod Security standards and safe security contexts.
- Use default-deny network policy and narrow security groups.
- Encrypt data with appropriate KMS controls and rotate secrets.
- Scan ECR images with Amazon Inspector and deploy immutable digests.
- Use admission policy to reject privileged, unsigned, unapproved, or incorrectly configured workloads.

Detect

Enable useful EKS control-plane logs: API, audit, authenticator, controller manager, and scheduler according to need. Centralize CloudTrail. GuardDuty EKS Protection can analyze audit activity and runtime signals where enabled. Security Hub aggregates supported findings. CloudWatch and SIEM rules need owners and response playbooks.

Respond

Preserve evidence, isolate the workload, revoke identity, rotate secrets, replace compromised nodes, redeploy known-good images, and test recovery. Do not debug a compromised node back into production.

Feynman check

The cluster is a building. IAM is the gate, RBAC is room access, network policy is the hallway rule, image admission checks deliveries, and logs are cameras. One lock is not a security architecture.

\newpage

10. Traffic, APIs, and global entry

Separate global edge traffic, VPC load balancing, Kubernetes routing, and service-to-service communication.

Outside to inside

Route 53 answers DNS. CloudFront can cache and protect global HTTP delivery. AWS WAF filters supported HTTP traffic and AWS Shield helps with DDoS protection. ACM provides certificates for supported endpoints.

The AWS Load Balancer Controller creates:

- Application Load Balancers for Layer 7 HTTP/HTTPS routing.
- Network Load Balancers for high-performance Layer 4 TCP/UDP/TLS patterns.

Gateway API gives role-oriented Kubernetes routing when supported by the chosen controller. Services provide stable discovery inside the cluster.

Inside the platform

Use timeouts, bounded retries with jitter, circuit breaking, idempotency, and graceful degradation. A service mesh can add traffic policy, workload identity, and telemetry, but it also adds proxies, upgrades, latency, cost, and a new failure surface. Adopt one only for a named requirement.

Hybrid and SaaS integration

Use Direct Connect or VPN for designed private hybrid connectivity. Use queues and events to protect slow ERP systems. Use PrivateLink for private service consumption without broad network routing. Never make a user request wait indefinitely for SAP, Salesforce, or Workday.

Feynman check

DNS tells visitors which building. The load balancer chooses a healthy door. The Kubernetes Service finds the right workers. Timeouts stop one slow room from trapping everyone.

\newpage

11. Multi-account and multi-cluster platform

A platform is a paved road: safe defaults, reusable services, and clear escape hatches. It is not one giant cluster controlled by one team.

Boundary choices

Use accounts for strong ownership, billing, quota, and policy separation. Use clusters for regional, trust, lifecycle, or failure boundaries. Use namespaces for lighter team and application separation. Keep production and non-production blast radii intentional.

Platform contract

Publish supported Kubernetes versions, compute models, ingress patterns, identity method, base policies, SLOs, upgrade cadence, incident ownership, backup expectations, and deprecation windows. Offer templates for Deployments, autoscaling, observability, Pod Identity, and delivery.

Reconciliation

Use infrastructure as code for accounts, VPCs, clusters, IAM, and add-ons. Use GitOps such as Argo CD or Flux for governed Kubernetes configuration. Avoid two controllers managing the same field. Roll global policy from audit to canary to enforcement.

Fleet operations

Standardize where sameness lowers risk; allow differences where region, regulation, workload, or lifecycle requires them. Track cluster inventory, versions, add-ons, policy exceptions, ownership, cost, and SLOs.

Feynman check

The platform team builds roads, signs, and guardrails. Application teams drive their cars. A paved road makes the safe path easy without taking away every destination.

\newpage

12. Delivery and software supply chain

A secure deployment proves what code became what image, who approved it, and what is running now.

The path

1. Developer changes reviewed source.
2. CI runs tests, dependency checks, and policy checks.
3. A controlled builder creates the image.
4. The image is scanned and stored in ECR.
5. The release references an immutable digest and provenance.
6. Admission policy accepts only approved artifacts and workload settings.
7. Progressive delivery sends limited traffic first.
8. SLO evidence decides promotion or rollback.

Use short-lived CI federation such as OIDC instead of repository access keys. Separate build, deploy, and approval permissions. Protect production branches and environments.

Rollouts

Set `maxUnavailable` and `maxSurge` from actual capacity. Readiness must reflect the user path. Canary or blue/green releases reduce blast radius only when metrics and rollback are reliable.

Database changes

Use expand-and-contract migrations: add compatible schema, deploy code that works with both versions, migrate data, then remove the old schema later.

Feynman check

An image tag is a label that can move. A digest is the parcel's fingerprint. For rollback and forensics, deploy the fingerprint.

\newpage

13. Observability and SRE

Observability should answer: Are users healthy? What changed? Where is the bottleneck? Who owns the response?

Signals

Use metrics, logs, traces, events, and profiles where useful. Start with request rate, errors, duration, and saturation. CloudWatch Container Insights can collect cluster/container signals. EKS control-plane logging captures API and audit evidence. ADOT can export OpenTelemetry data. Amazon Managed Service for Prometheus and Amazon Managed Grafana support Prometheus-style monitoring and dashboards.

SLOs

An SLI is a measurement such as successful checkout requests. An SLO is the target, such as 99.95% over 30 days. The error budget is the permitted bad portion. Alert on fast and slow error-budget burn, not every harmless spike.

Cost and signal quality

Unbounded log volume, metric cardinality, and trace sampling can be expensive. Define retention, redaction, ownership, sampling, and query needs. Never log credentials or unnecessary personal data.

Runbooks

An alert needs an owner, impact, dashboard, first checks, safe mitigation, escalation, and verification. Practice incident response and record learning without blame.

Feynman check

A dashboard is a car dashboard: it shows whether the journey is safe. An alert is the warning light that asks a person to act. A pile of logs is only raw material.

\newpage

14. Reliability, upgrades, and disaster recovery

Reliability comes from removing single failure assumptions and rehearsing change.

Availability

Spread nodes and Pods across Availability Zones. Use topology constraints, anti-affinity, readiness, and disruption budgets. Ensure databases, queues, DNS, certificates, identity, quotas, and external integrations meet the same user journey SLO.

EKS upgrades

EKS Kubernetes versions have a standard support period and then extended support. Extended support costs more; it is a bridge, not an upgrade strategy. Use EKS Upgrade Insights to find deprecated APIs. Test in a lower environment and upgrade one minor version at a time.

The control plane, EKS add-ons, node images, and applications have separate lifecycle work. Managed node groups are not automatically upgraded when the control plane changes. Roll nodes safely and confirm

workloads, PDBs, and capacity.

Disaster recovery

Select backup/restore, pilot light, warm standby, or active/active from RTO/RPO and business cost. Preserve IaC, Kubernetes objects, images, data, identity, secrets, DNS, and capacity. A green backup job is not proof; a timed restore is.

Feynman check

High availability handles expected local breakage while the system stays running. Disaster recovery rebuilds service after a larger loss. You need both when the business requires both.

\newpage

15. Cost, performance, and sustainability

The goal is not the smallest bill. It is the required outcome with the least waste and acceptable risk.

Main cost drivers

Count EKS cluster fees, extended-support fees, EC2 or Fargate compute, EBS/EFS, load balancers, NAT processing, cross-AZ and internet transfer, logs, metrics, security tools, support, and engineer time.

Reduce waste

- Right-size requests with measurements.
- Use HPA plus Karpenter, Auto Mode, or node scaling intentionally.
- Keep stable baseline on appropriate On-Demand or Savings Plans.
- Use diversified Spot for interruption-tolerant capacity.
- Evaluate Graviton and efficient images.
- Remove idle clusters, load balancers, volumes, snapshots, and ECR images.
- Use VPC endpoints and locality when they reduce unnecessary transfer/NAT cost.
- Control log retention, metric cardinality, and trace sampling.

Performance

Measure latency percentiles, throughput, errors, saturation, scheduling delay, image pull time, DNS, database connections, and dependency latency. Load-test autoscaling and recovery, not only steady state.

Unit economics

Tag and allocate by owner, environment, application, and cost center. Track cost per order, user, tenant, build, or request. A growing bill may be healthy when unit cost falls and business volume grows.

Feynman check

A half-empty bus may look cheap, but cost per passenger is high. Good platform economics packs work safely without making one crowded bus a single failure.

\newpage

16. Real-life scenario: MarketBridge

MarketBridge is a global retailer. Its customer channels need fast releases, but orders must still reach SAP, customer context comes from Salesforce, and employee changes come from Workday.

Business requirements

- Checkout availability: 99.95%; product browsing should degrade gracefully.
- Campaign traffic: ten times normal load within ten minutes.
- No order loss; SAP may be slow for hours.
- Production data remains in approved regions.
- RPO: five minutes. RTO: one hour for ordering.
- Teams need independent releases with central security guardrails.
- Measure cost per completed order.

Proposed architecture

Route 53 directs users to CloudFront. WAF protects public HTTP traffic. An ALB created by the AWS Load Balancer Controller routes to EKS.

Use one production cluster per active region/account and separate non-production. Start customer APIs on EKS Auto Mode because they need standard Kubernetes behavior but no node access. Keep a small managed node group only if an approved host-level integration agent truly needs it.

Pods use EKS Pod Identity. Checkout writes the order transaction to Aurora and publishes an event through an outbox pattern. SQS buffers work for the SAP adapter. Idempotent consumers and a DLQ prevent duplicates and poison messages. EventBridge distributes business events. Salesforce and Workday adapters have timeouts, rate limits, retry policies, and private connectivity where supported.

Store images in ECR, scan with Inspector, deploy digests, and enforce Pod Security plus admission policy. Use default-deny network policy. Secrets live in Secrets Manager with KMS. Centralize CloudTrail and EKS

audit logs.

Spread replicas and compute across three AZs. HPA scales Pods on request and queue metrics; Auto Mode supplies nodes. Use Spot only for retryable workers. Protect Aurora with backups and cross-region recovery aligned to RPO/RTO.

What happens when SAP stops?

Checkout still accepts orders after the durable order commit. SQS depth rises. Autoscaling is capped to protect SAP. Users see “order received,” not an endless spinner. Operators get an alert from queue age and error-budget impact. When SAP returns, idempotent workers drain the queue.

Decision record

Decision: Auto Mode for general workloads, asynchronous ERP integration, managed databases, and separate regional production clusters.

Rejected: one giant global cluster, synchronous SAP calls in checkout, permanent access keys, and active/active everything without business proof.

Prove it

Load-test the tenfold campaign, terminate nodes and Pods, simulate AZ and SAP failure, rotate secrets, roll back a bad image, restore into an isolated environment, and measure recovery and cost per order.

\newpage

17. Interview guide, assessment, and glossary

There is no separate official “Amazon EKS certification.” This fieldbook uses the official AWS Certified Solutions Architect – Associate question shape as a practice benchmark, then makes the content EKS-heavy.

Assessment facts

The official SAA-C03 exam currently has 65 questions in 130 minutes, including multiple-choice and multiple-response items. AWS reports 50 scored and 15 unscored questions and a scaled passing score of 720. Our mock has 65 original questions, uses a transparent 72% practice target, and is not copied exam content.

Interview answer frame: WORKLOAD

- **W**hy: users, outcome, SLO, RTO/RPO, regulation, budget.
- **O**wnership: accounts, teams, access, shared responsibility.
- **R**untime: Auto Mode, managed nodes, Fargate, or Karpenter.
- **K**ubernetes: workload objects, resources, probes, placement, disruption.
- **L**inks: VPC, traffic, identity, integrations, data.
- **O**perations: delivery, logs, metrics, alerts, upgrades, incident response.
- **A**vailability: zones, regions, backup, restore, failure tests.
- **D**ollars: allocation, unit cost, scaling, commitment, Spot, transfer.

Essential abbreviations

Term	Meaning
EKS	Elastic Kubernetes Service
EC2	Elastic Compute Cloud
ECR	Elastic Container Registry
VPC	Virtual Private Cloud
CNI / CSI	Container Network / Storage Interface
ENI	Elastic Network Interface
IAM / RBAC	AWS identity authorization / Kubernetes authorization
IRSA	IAM Roles for Service Accounts
ALB / NLB	Application / Network Load Balancer
HPA / VPA	Horizontal / Vertical Pod Autoscaler
PDB	PodDisruptionBudget
AZ	Availability Zone
HA / DR	High Availability / Disaster Recovery
RPO / RTO	Tolerated data loss / restoration time
SLI / SLO / SLA	Indicator / objective / agreement
IaC	Infrastructure as Code
CI/CD	Continuous Integration / Delivery
DLQ	Dead-Letter Queue
KMS	Key Management Service

Final Feynman challenge

Explain EKS to a colleague in two minutes. Draw the managed control plane, worker choices, VPC paths, human and Pod identity, delivery path, data, and failure domains. Then explain one trade-off in each. Anything you cannot explain simply is your next study target.

Official references

- Amazon EKS User Guide
- EKS best-practices guides
- EKS Auto Mode
- EKS security best practices
- EKS Kubernetes versions
- SAA-C03 exam guide