What we are doing for the AWS monitoring right now? We don't have anything. Not really.

If you truly have “nothing” in AWS monitoring right now, that’s a gap, not a preference. You’re flying blind. When something breaks, you’ll find out from customers or a random Slack message, and you’ll waste hours proving what happened.

Here’s what you should be doing immediately, in a practical order.

1) Minimum baseline (same day)

- **CloudWatch Metrics** + **Alarms** for the basics:
  - **ALB/NLB**: 5XX, target 5XX, target response time, healthy host count
  - **ECS/EKS**: CPU/memory utilization, task/pod restarts, desired vs running, node disk pressure
  - **Lambda**: errors, throttles, duration p95/p99, concurrent executions
  - **RDS**: CPU, free storage, connections, read/write latency
  - **SQS**: queue depth, age of oldest message
- **Notifications**: send alarms to **SNS** → **Slack/PagerDuty** (whatever you use).
- **CloudTrail enabled everywhere** (management events at minimum). Without it, you can’t answer “who changed what”.

2) Centralize logs (1–2 days)

Pick one:

- **CloudWatch Logs** as the starting point (fastest).
- Or **Splunk / Datadog / OpenSearch** if that’s your standard.

Then enforce:

- **ALB access logs** → **S3**
- **VPC Flow Logs** → **CloudWatch/S3**
- **CloudWatch Logs retention** set (don’t leave it “Never expire”).
- **EKS/ECS app logs** shipped consistently (FireLens/Fluent Bit/OTel Collector).

3) Dashboards that answer real questions (2–3 days)

Build 3 dashboards:

- **Service health**: traffic, latency, error rate, saturation (CPU/mem), dependency health
- **Infra health**: nodes/instances, disk, network errors, autoscaling activity
- **Deployment/change view**: deploy markers, config changes, scaling events, incident timeline

If you can’t see **RPS + p95 latency + error rate + saturation** for every tier, you’re not monitoring. You’re collecting noise.

4) Tracing + SLOs (next step, not first)

Once metrics and logs exist:

- Add **OpenTelemetry** tracing for key services.
- Define **SLOs** (ex: “99.9% of requests < 500ms”, “5xx < 0.1%”) and alert on **SLO burn rate**, not raw CPU.

5) Guardrails (quietly prevent future pain)

- Standard **alarm module** in Terraform (same alarms everywhere).
- **Tagging** standard for service/env/team so dashboards and alerts group correctly.
- **Runbooks** linked in every alert (or the alert is useless).

What I need from you (answer in one line each, no long story)

1. What runs today: **EKS / ECS / EC2 / Lambda / RDS** (which ones)?
2. Where do you want alerts: **Slack / PagerDuty / Email**?
3. Do you already have **Splunk/Datadog** or do we start with **CloudWatch only**?

If you reply with those 3 lines, I’ll give you a clean “week-1 monitoring plan” with exact alarms and thresholds per service type.

in terms of general AWS account health or cluster health, we don't have any monitors set up, which is kind of what we're trying to address here. Okay. So we need to figure it out for AWS health monitors, right, for our clusters, what we require to monitor related with our infrastructure side. It's not for app side, right?

Got it. This is **infrastructure health only**, not application observability. And yes, you currently have **zero AWS-level health coverage**. That's the problem you're fixing.

I'll be very direct and scoped.

What "AWS / cluster health monitoring" actually means

This is about answering **one question** during an incident:

> *Is the cloud and the cluster healthy, or is this an app problem?*

Right now, you can't answer that.

So we focus on **account**, **region**, **cluster**, **node**, and **control-plane** health. Nothing app-specific.

Layer 1: AWS Account & Regional Health (must-have)

These are non-negotiable. They detect AWS-side issues and risky changes.

Required monitors

- **AWS Health events**
  - Service outages
  - Scheduled maintenance
  - Degraded performance
- **CloudTrail**
  - Enabled in all regions
  - Alert on:
    - IAM changes
    - Security group / NACL changes
    - EKS cluster updates
- **Service Quotas**

  - EC2 instance limits
  - EIP limits
  - Load balancer limits

Outcome

You immediately know:

- “AWS is having an issue”
- “Someone changed infra”
- “We’re about to hit a hard limit”

Layer 2: Cluster Control Plane Health (EKS-specific)

This is **not app monitoring**. This is Kubernetes itself.

Required monitors

- **EKS Control Plane**
  - API server availability
  - API request throttling
  - Auth failures
- **Cluster Autoscaler**
  - Failed scale-ups
  - Unschedulable pods
- **Core system pods (kube-system)**
  - CoreDNS
  - kube-proxy
  - CNI pods
  - Metrics server

Outcome

You know if:

- The API server is unhealthy
- Scheduling is broken
- DNS or networking inside the cluster is degraded

Layer 3: Node & Capacity Health (where most outages hide)

This is where infra teams usually fail.

Required monitors

- **Node readiness**
    - NotReady nodes
- **Node pressure**
    - DiskPressure
    - MemoryPressure
    - PIDPressure
- **EC2 backing nodes**
    - CPU steal
    - Disk IO wait
    - Network errors
- **Auto Scaling Groups**
    - Failed launches
    - Termination storms

Outcome

You can answer:

- “Do we have enough capacity?”
- “Are nodes dying or unhealthy?”
- “Is autoscaling actually working?”

Layer 4: Network & Ingress Health (infra, not app)

Still infrastructure. Still required.

Required monitors

- **Load balancers (ALB/NLB)**
    - Healthy target count
    - 5xx at LB level
- **VPC**
    - NAT gateway errors
    - Route table changes

- **CNI**
  - IP exhaustion
  - ENI attach failures

Outcome

You catch:

- Traffic can’t enter the cluster
- Pods can’t get IPs
- AWS networking is the bottleneck

What NOT to include (yet)

Be explicit about this to stakeholders:

- ❌ App error rates
- ❌ Business SLIs
- ❌ Request latency per endpoint
- ❌ Custom app dashboards

Those come **after** infra is visible.

Minimal tooling (don’t over-engineer)

You can do **all of this with**:

- CloudWatch metrics + alarms
- AWS Health
- CloudTrail
- One notification channel (Slack or PagerDuty)

No Splunk, no Datadog required to start.

One-sentence summary you can use with leadership

> “Right now we have no visibility into AWS account, region, or Kubernetes cluster health. We’re putting baseline infrastructure monitors in place so we can distinguish AWS or cluster failures from application issues.”

If you want, next I can:

- Give you a **checklist of exact alarms** (by name and threshold), or
- Design a **Terraform module** for “infra-only AWS & EKS health monitoring”

Say which one you want.