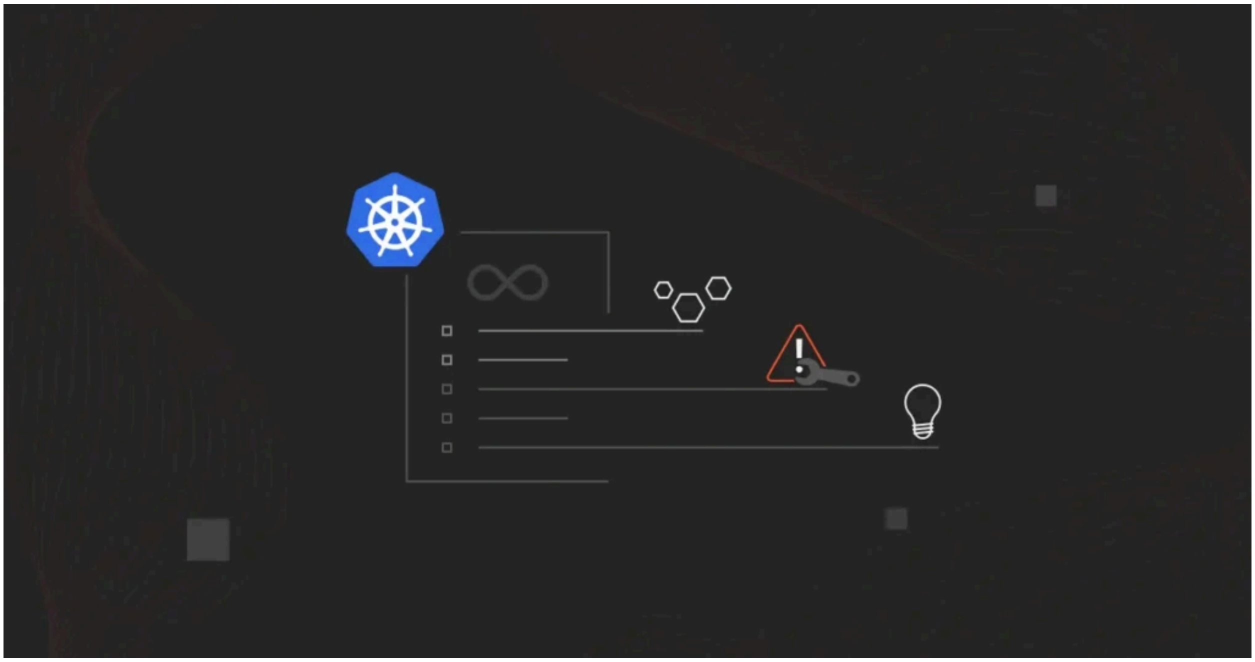
February 25, 2025, 16 min read

25 Kubernetes Interview Questions You Must Know In 2025

Want to ace those Kubernetes interview questions? Dive in to find tips, tricks, and the kind of mindset interviewers (and future colleagues) are looking for.

By: Cody Slingerland



We'd love to interview you!

It's exciting to see this message pop up in your inbox. It's also nerve-wracking. Suddenly, impostor syndrome kicks in — am I ready?

That's because an interview isn't just about showing competence; it's your chance to prove your expertise. And the thought of not getting a call back can feel like you don't measure up, leading to self-doubt.

Whether it is your first time or the umpteenth, remember that every interview is a learning opportunity that brings you one step closer to your goals.

To help you embrace the challenge and trust in the skills and experience you've built, we've compiled a list of solid Kubernetes interview questions to help you prepare and ace your next session.

Why Is Kubernetes Knowledge So Essential For Modern DevOps?

More companies are embracing microservices and cloud-native technologies. While these solutions offer scalability, they can become complex if not managed properly.

That's where Kubernetes comes in. It provides a powerful platform for orchestrating microservices and containers across clusters.

By automating deployment, scaling, and orchestration, Kubernetes ensures your applications run smoothly, efficiently, and reliably.

This automation allows businesses to scale fast, roll out updates seamlessly, and maintain high availability. And if you can do this, well, you're in high demand.

Companies know that skilled Kubernetes engineers streamline operations, minimize downtime, and drive innovation — directly impacting their bottom line. And they're more than willing to back that up with a six-figure paycheck.

How To Prepare For Your Next Kubernetes Engineer Job Interview

You might be eager to jump straight into common **Kubernetes interview questions** or the troubleshooting challenges you'll likely face.

But remember, interviewers aren't just assessing your theoretical knowledge and hands-on experience. They're also evaluating your soft skills.

With that in mind, here are some key interview preparation tips to practice for that Kubernetes engineer role.

- **Start by mastering containerization** — especially Docker — and dive deep into Kubernetes fundamentals like pods, deployments, services, and persistent storage.
- **Earning certifications** such as Certified Kubernetes Administrator (CKA) or Certified Kubernetes Application Developer (CKAD) can boost your resume and validate your expertise.
- **Hands-on experience is just as crucial.** Work on personal projects, contribute to open-source Kubernetes initiatives, or set up your own cluster to gain real-world insights.

- **Also, feel free to ask them questions** so you can learn about their setup and specific use cases and tailor your answers.

Most importantly, **be ready to clearly communicate your hands-on experiences with K8s** and how you approach complex system architecture. Strong technical skills and articulate explanations will help you stand out in this competitive job market.

And now for the juicy part…

Top Kubernetes Interview Questions (And Tips To Answer Them Like A Pro)

Here's the critical part. Whenever possible — unless explicitly told otherwise — back up your knowledge with real-life examples.

Sharing how you've applied Kubernetes concepts to solve real problems or improve containerized operations proves you can move beyond theory and deliver practical solutions.

This not only highlights your expertise but reinforces your value as a problem-solver and key contributor to your team and organization.

What is Kubernetes, and why do you think it is so popular today?

What interviewers are looking for here is a clear and concise definition of Kubernetes, your understanding of why companies are adopting it, and your awareness of real-world use cases.

Kubernetes (K8s) is an open-source container orchestration platform that automates deploying, scaling, and managing containerized applications. It abstracts away infrastructure complexities, allowing teams to focus on building and running applications efficiently.

Its popularity stems from simplifying microservices architectures, managing scalability, and enabling cloud-agnostic deployments.

Kubernetes follows a master-worker node architecture.

1. Control Plane (Master Node):

   - **API Server**: The front end of Kubernetes, handling all requests.
   - **Controller Manager**: Ensures desired state by managing controllers (e.g., replication, node health).
   - **Scheduler**: Assigns pods to nodes based on resource availability and constraints.
   - **etcd**: A distributed key-value store that holds cluster state and configurations.

2. Worker Nodes:

   - **Kubelet**: Communicates with the API server, ensuring pods run as expected.
   - **Container Runtime**: Runs containers (e.g., Docker, containerd).
   - **Kube Proxy**: Manages networking and routes traffic to pods.

This architecture ensures scalability, self-healing, and automation, making Kubernetes a resilient platform.

Overall, this is your opportunity to demonstrate your understanding of the major components of Kubernetes, their roles, and, crucially, how these components work together.

What feature do you wish Kubernetes had? And what’s most frustrating about using K8s?

Another way to approach this question is, “What are some Kubernetes pain points your team has encountered, and how have you worked around them?”

Potential answer:

YAML complexity and debugging issues can be challenging. Debugging failed deployments often requires checking multiple logs, events, and networking configurations, which can be cumbersome, especially in large-scale environments.

exist, a native Kubernetes cost management tool would be a game-changer.

Explain the role of a master node — what happens when a master fails, and how do you get it back up?

The master node (control plane) manages the cluster's desired state, scheduling, and coordination. It consists of the API server, controller manager, scheduler, and etcd.

If a single master node fails in a non-HA setup, the cluster becomes unmanageable, but running workloads continue until nodes or pods require updates.

To recover, you would:

- Check etcd health and restore from a backup if necessary.
- Recreate the master node with the same configurations.
- Rejoin worker nodes to the cluster if disconnected.

For high availability (HA) clusters, multiple master nodes are used, and failure is mitigated via leader election mechanisms and redundant etcd instances.

What is a pod in Kubernetes?

A pod is the smallest deployable unit in Kubernetes, representing one or more tightly coupled containers that share networking and storage. Containers in a pod communicate via localhost and can share volumes.

Pods are ephemeral, so if a pod fails, Kubernetes replaces it with a new one. To ensure persistence, we use ReplicaSets, Deployments, or StatefulSets for automatic pod management.

You want to share your knowledge of service types (and their use cases) and handling internal and external traffic.

How do you tie a service to a pod or a set of pods in Kubernetes?

A Kubernetes Service ties to pods using selectors and labels. First, assign a label to the pods:

labels:

 app: my-app

Then, define a Service that selects pods with that label:

selector:

 app: my-app

This enables Kubernetes to dynamically associate the service with the correct pods, even as they scale.

What is Kubelet? What is Kubectl?

Kubelet runs on each worker node, ensuring pods are running and reporting back to the control plane. It interacts with the container runtime to start and stop containers. Meanwhile, Kubectl is the command-line tool for interacting with Kubernetes clusters, used for deploying applications, managing resources, and troubleshooting.

You can also share one or a few of your go-to kubectl commands for troubleshooting pod issues.

Would you use Docker Swarm over Kubernetes, and why?

Docker, Kubernetes, and OpenShift.

Related Reads:

10 Top Kubernetes Alternatives (And Should You Switch?)

13+ Docker Alternatives To Consider in 2025

What are some examples of Kubernetes security measures you've used and loved?

Here, you'll want to share some Kubernetes architecture best practices you know and have used to improve its security. These include:

- Role-Based Access Control (RBAC) to manage permissions more efficiently
- Network policies to restrict pod-to-pod communication
- Pod Security Standards (PSS) to enforce best practices like non-root containers
- Secrets management via Kubernetes Secrets, external vaults like HashiCorp Vault
- Image scanning with tools like Trivy or Anchore
- Pod security policies and admission controllers for pre-deployment checks

See additional K8s security best practices here.

Explain the concept of Ingress

Ingress is a Kubernetes API object that manages external access to services, typically via HTTP/HTTPS. Instead of exposing services with LoadBalancer or NodePort, Ingress routes traffic based on hostnames, paths, or TLS settings.

Share an example use case, such as how a single Ingress controller (e.g., Nginx, Traefik) can manage traffic for multiple services and what rules you'd use.

not a K8s best practice because:

- It **clutters resources**, making them challenging to manage.
- It **increases security risks**, as all resources share the same space.
- It **limits access control**, as RBAC rules apply at the namespace level.

Instead, you'll want to organize workloads into namespaces to improve security, scalability, and maintainability.

What does the Kube-scheduler do?

This is meant to prod your understanding of scheduling mechanisms and the factors that affect scheduling. The Kube-scheduler assigns pods to nodes based on resource availability, affinity rules, and constraints. It considers:

- CPU & memory requests/limits
- Node taints & tolerations
- Pod affinity/anti-affinity
- Custom scheduling policies

If a pod cannot be scheduled, it remains in a Pending state.

What kinds of services would you consider a natural fit to host on Kubernetes?

Show your understanding of diverse workloads by discussing these top K8s use cases:

- Microservices architectures (e.g., REST APIs, GraphQL services)
- Event-driven applications (Kafka consumers, message queues)

How does a Horizontal Pod Autoscaler (HPA) work?

HPA automatically scales pods based on CPU/memory utilization or custom metrics. It monitors pod usage and adjusts the replica count dynamically.

Give an example HPA definition to support your answer.

How does Kubernetes handle rolling back deployments?

Kubernetes allows easy rollbacks via kubectl. It keeps a history of ReplicaSets, enabling you to revert to a previous state. Some K8s rollback strategies include:

- **Recreate** deletes old pods before creating new ones.
- **RollingUpdate** gradually replaces old pods.
- **Canary Deployments** gradually shift traffic to new versions.

How do applications in Kubernetes accept traffic from clients?

Here, the interviewer wants to hear your understanding of Kubernetes networking option and best practices for exposing services.

Applications in Kubernetes receive traffic through Services and Ingress Controllers:

- **ClusterIP** (default) caters to internal communication within the cluster.
- **NodePort** exposes a service on a static port (30000-32767) on each node.
- **LoadBalancer** provisions a cloud-based external IP for direct access.
- **Ingress** routes external HTTP/HTTPS traffic using hostnames and paths.

When a node fails, Kubernetes automatically:

- Marks the node as NotReady after missing heartbeat signals.
- Evicts pods from the failing node if PodDisruptionBudgets allows it.
- Reschedules workloads onto healthy nodes based on resource availability.

For resiliency and high availability (HA):

- Use multi-node control planes to avoid single points of failure.
- Enable Pod Anti-Affinity to distribute workloads across nodes.
- Implement self-healing mechanisms via liveness probes and auto-scaling.

Overall, the goal here is to demonstrate you understand how to design and maintain a reliable K8s environment.

What is the difference between a StatefulSet and a Deployment? When would you use one over the other?

Deployments are used for stateless applications where pod identity doesn’t matter. You can use it for web services, APIs, and microservices. StatefulSets is used for stateful applications (where pod identity and persistent storage are crucial, such as databases and message queues). Some ideal use cases include PostgreSQL and MySQL databases, Kafka, and Zookeeper.

StatefulSets ensure:

- Stable network identity (each pod gets a consistent hostname).
- Ordered scaling and termination (pods start/stop sequentially).

CRDs empower you to define custom Kubernetes objects, extending Kubernetes beyond built-in resources like Pods and Services. CRDs enable custom controllers and Operators, automating complex application management.

Have any experience with CRDs in production? Be sure to share it.

How does Kubernetes facilitate load balancing for applications?

Kubernetes provides three levels of load balancing:

- **Internal Load Balancing (ClusterIP):** This distributes traffic to pods within a cluster.
- **External Load Balancing (LoadBalancer Service):** It provisions cloud-based external load balancers.
- **Ingress Load Balancing:** Routes traffic based on hostnames and paths.

For pod-to-pod balancing, Kube-proxy implements IPTables mode (the default mode and efficient packet forwarding) and IPVS mode, which offer better performance for large-scale clusters.

How would you address issues with network policies blocking communication between pods?

Here’s a structured debugging approach to showcase your understanding of Kubernetes Network Policies (use commands).

- Check existing network policies
- Describe the network policy

What steps would you take if a node becomes NotReady in a Kubernetes cluster?

Highlight your understanding of Kubelet (node lifecycle) and how to systematically troubleshoot node issues with these steps:

- Check the node status
- Describe the node for events/logs
- Check Kubelet logs for errors
- Verify system resource availability
- Restart Kubelet
- Remove the node from the cluster if it’s unresponsive
- Scale the cluster up if needed. Cluster Autoscaler may replace the failed node if it’s a cloud-based cluster.

How would you troubleshoot issues with Persistent Volumes not being mounted correctly?

Use your understanding of PVC, PV, and storage classes to:

- Check the PVC status
- Describe the PVC for errors
- Verify the storage class
- Check the underlying cloud storage. If using AWS, check EBS Attachments, and if using Azure, verify Azure Disk provisioning.
- Inspect the pod logs for mount failures
- Manually test volume mounting

- Describe the Ingress resource
- Verify the Ingress Controller is running. Restart it if it's down.
- Check service backend mapping.
- Inspect the Ingress logs
- Manually test routing with curl

How do you ensure your Kubernetes environment is cost-efficient?

Picture this: six out of ten respondents in our State of Cloud Cost survey reported that their cloud costs are higher than they should be.

And if you think K8s has something to do with it, you are right!

Rising Kubernetes costs are becoming a major headache for many teams, making it challenging to keep spending in check without sacrificing scalability or innovation.

It doesn't help that Kubernetes lacks a robust, built-in cost management solution.

On top of that, many cost management tools fail to deliver the accuracy, detail, and control you need to keep costs in check — without slowing down engineering velocity or hindering scalability.

Not CloudZero.

Be The Most Cost-Conscious Kubernetes Engineer In The Room With CloudZero

Want to build cost-efficient solutions from day one? CloudZero helps you do just that — while ensuring you stand out as a financially-mindful innovator.

costs per Pod, per Namespace, per Feature, per Team, and more. You'll know exactly where your money is going — so you can stop the bleeding before Finance says, "Stop, we're overspending!"

Business-first insights: Go beyond numbers — understand which features and products drive costs. CloudZero ties your Kubernetes spend directly to business metrics (like cost per deployment, feature, or customer) making chargebacks and showbacks more precise and actionable.

Your single source of truth: Unlike tools focusing only on Kubernetes, CloudZero connects both in-cluster and external cloud costs, giving you a complete view of your cloud spend — critical for informed, strategic decisions.

Real-time anomaly detection: Receive real-time alerts straight in your inbox or Slack. You'll catch unusual spending patterns early — not a month and a fat bill later.

Scalable and intuitive: Whether you're a fast-growing startup or a global enterprise, CloudZero scales with you, turning complex cost data into clear, actionable business insights.

Cost-conscious engineers at Shutterstock, Coinbase, and Drift trust CloudZero to control their cloud spend. Drift saved $2.4 million, Upstart cut over $20 million, and Skyscanner found enough savings in one week to pay for their entire year's subscription. Schedule a demo today and see how CloudZero can help you become the most profitable engineer in the room.

Bonus Kubernetes Interview Questions For Practice

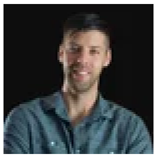
What is the role of etcd in Kubernetes, and how does it ensure data consistency?

What are taints and tolerations, and how are they applied in scheduling?

How do you troubleshoot if the POD is not getting scheduled?

How would you debug a CrashLoopBackOff error in a pod?

What steps would you take to diagnose and resolve DNS issues in a Kubernetes cluster?



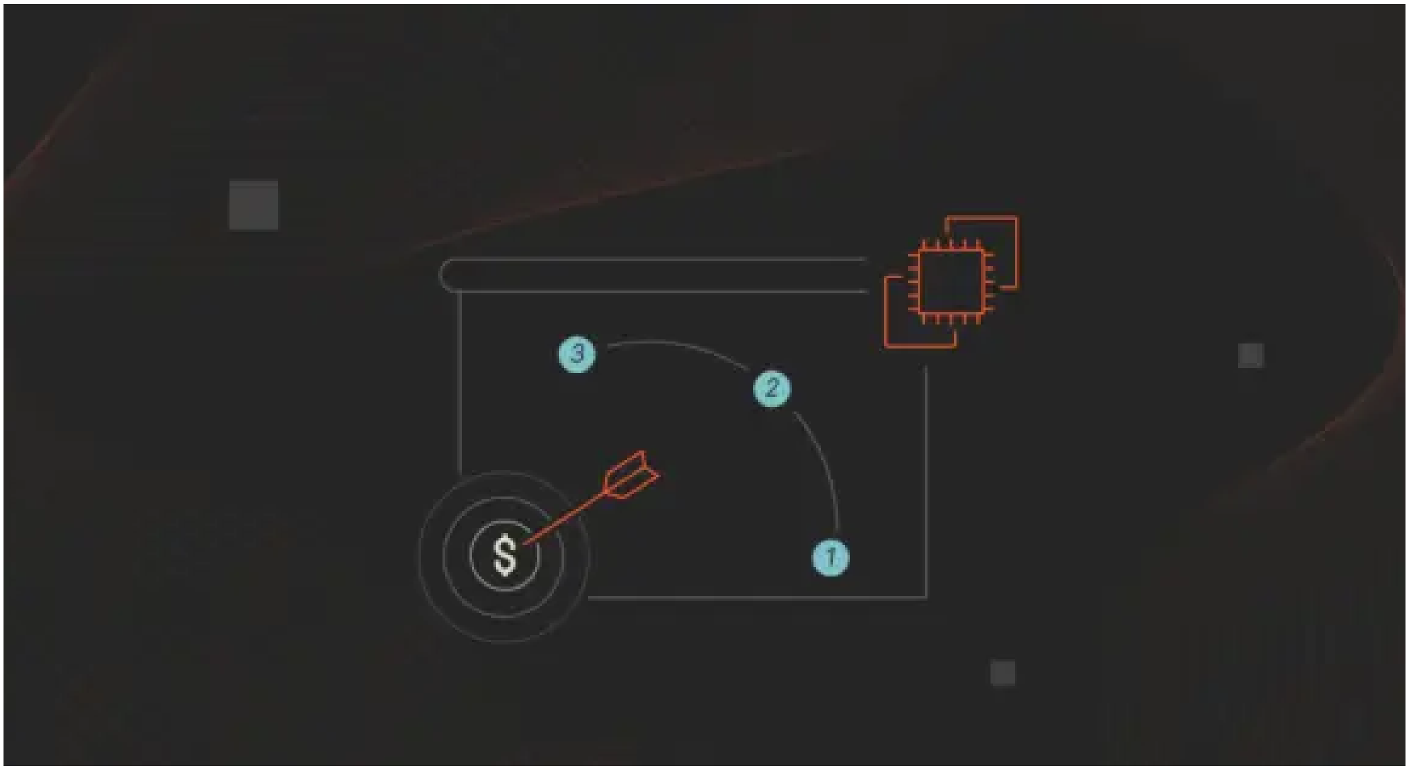
Author: Cody Slingerland

Cody Slingerland, a FinOps certified practitioner, is an avid content creator with over 10 years of experience creating content for SaaS and technology companies. Cody collaborates with internal team members and subject matter experts to create expert-written content on the CloudZero blog.

in

Suggested Articles

See more



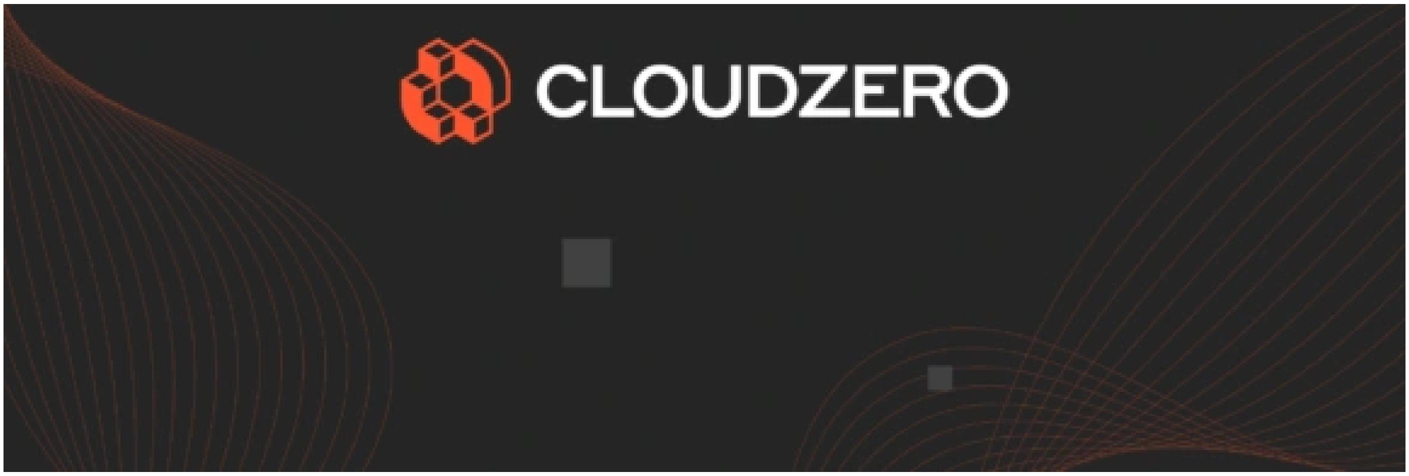
Blog

November 20, 2023

3 Simple EC2 Strategies To Ensure Cost Efficiency

The State Of AI Costs In 2025

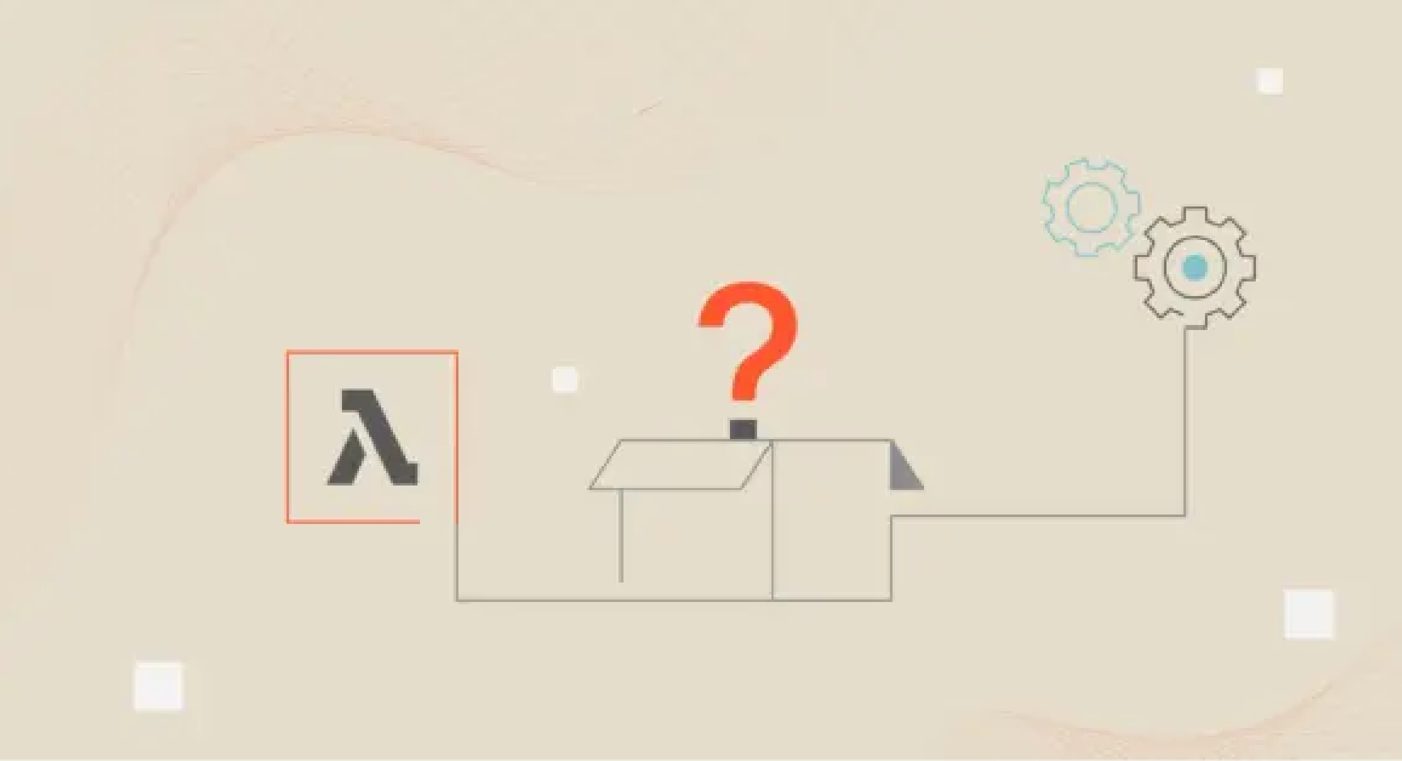
Learn More



Blog

November 13, 2023

Why Automatic Cloud Cost Optimization Isn’t Enough



Solutions

- Data Normalization
- Cost Allocation
- Shared Cost
- Cost Per Customer
- Unit Cost
- Kubernetes Visibility
- Cost Optimization
- Budget Management
- For Engineering
- For FinOps
- For Finance
- For DevOps

Platform

- Why Change?
- Overview
- Anomaly Detection
- Explorer
- Analytics
- Budgeting And Forecasting
- Optimization Insights
- Dimensions
- Demo
- Tour
- FinOps Enablement
- Customer Success
- Pricing

Integrations

- AnyCost

Resources

- Blog

Datadog

Kubernetes

Databricks

MongoDB

New Relic

All Integrations

Free Cost Assessment

AWS Service Comparison

Academy

Documentation

FAQ

State Of Cloud Cost 2024

All Resources

Company

About

Leadership

Customers

Careers

Contact

Cookie Policy

The Best Tools By Category

Best Cloud Cost Management Tools

Best AWS Cost Optimization Tools

Best AWS Monitoring Tools

Best Azure Cost Management Tools

Best GCP Cost Optimization Tools

Best Multi-Cloud Management Tools

Cloud Cost Guides

The Cloud Cost Playbook

Cloud Cost Optimization 101

The Ultimate Guide To Cost-Efficient Cloud Spending

The Modern Guide To Managing Cloud Costs

FinOps 101: An Intro To The Basics Of FinOps

The State Of AI Costs In 2025

Learn More

CLOUDZERO

The Top Cloud Service Providers

The Cost Of Cloud Computing

Cloud Storage Pricing Comparison

Types Of Cloud Computing

FinOps™ CERTIFIED Platform

AICPA SOC aicpa.org/soc4so SOC for Service Organizations | Service Organizations

SPRING 2025 High Performer ENTERPRISE

SPRING 2025 Momentum Leader

SPRING 2025 Grid Leader

SPRING 2025 Best Support MID-MARKET

FinOps Foundation Premier Member

Terms Of Use Privacy Policy Security