

FOR PLATFORM AND INFRASTRUCTURE TEAMS

Coral FAQ

Short answers to the questions platform teams ask us most about how Coral sizes workloads, removes spare nodes and works with your autoscaler.

- **Pod sizing is opt-in**

Coral only recommends until you turn on **"Automatically optimize"** for a policy or a workload.

- **Node optimization is on from install**

Removing spare nodes starts at install, through node optimization and the Evictor (which empties nodes for your autoscaler to delete). Switch both off, put the Evictor in dry-run, or limit them to chosen node pools.

Numbers are product defaults. Answers describe Coral as released in October 2026.

01	How Coral sizes workloads	Q1–7	02	Safety, memory and rollback	Q8–12
03	Stateful and single-replica workloads	Q13–14	04	Working with HPA	Q15–17
05	Nodes and spare capacity	Q18–21	06	Autoscalers and traffic surges	Q22–23
07	Getting started	Q24			

SECTION 1

How Coral sizes workloads

1 Is the recommendation based on Average, P95, P99, Max, or a mix?

- **CPU:** a percentile of the last 3 days (P80 by default), plus 10% headroom.
- With a CPU-based HPA, the CPU request comes from replaying your HPA instead (question 16).
- **Memory:** the peak of the last 8 days, plus 10% headroom, because running out of memory kills the container.
- During a surge or CPU pressure, the CPU target is raised for a while, up to P99.
- You can change the target per workload or in a custom policy, up to P99.9 or max.

2 How much history is needed before the first recommendation?

- A first recommendation appears within the first hour.
- It is trusted after 24 hours of data. Jobs and CronJobs need 3 runs.
- The window then grows to 3 days for CPU and 8 days for memory.

3 CPU is 100m at night and 2000m from 10 AM to 2 PM. What does Coral recommend?

- Without an HPA, the base request sits near the night level: the 4 busy hours are only 17% of the day, and P80 leaves out the busiest 20%.
 - At 10 AM Coral detects the surge within minutes and raises the request. It drops back a few hours after the peak.
 - With a CPU-based HPA, Coral sizes the request by replaying load through your HPA instead (question 16).
 - To reserve 2000m all day, use the Stability policy or set a CPU minimum.
-

4 A workload is idle for 6 days and busy every Monday. Will Coral cut resources before Monday?

- **Memory: no.** Coral keeps last Monday's peak, because it looks back 8 days (unless more than about 10% of the data is missing).
 - **CPU: yes.** A short Monday peak doesn't hold the CPU request up, so it returns to the idle level after the peak.
 - On Monday Coral starts raising CPU within minutes. To be ready from the first minute, set a CPU minimum.
-

5 How are workloads handled that spike only at month-end or quarter-end?

- Coral looks back at most 8 days, so it doesn't remember last month's peak.
 - When the spike comes, Coral raises CPU within minutes, and raises memory after any OOM kill.
 - For known month-end workloads, set minimums or use a more conservative policy.
-

6 How does Coral learn Black Friday, Diwali or holiday traffic patterns?

- It doesn't forecast or use a calendar. It reacts to live usage.
 - Your HPA adds pods, your autoscaler adds nodes, and Coral raises requests that run hot.
 - Before a known event, set minimums on critical workloads.
 - For the event window, pause node optimization and the Evictor.
-

7 A workload needs 4 CPUs during startup but only 500m afterwards. How is it handled?

- By default Coral ignores the first 2 minutes of CPU after a container starts, so the request ends up near 500m.
 - Startup can still go above the request, using free CPU on the node, because Coral doesn't lower CPU limits on regular pods.
 - If startup must be fast, set a CPU minimum or use a policy that counts startup CPU.
 - Guaranteed StatefulSet pods and workloads with a CPU-based HPA are sized differently.
-

SECTION 2

Safety, memory and rollback

8 How does Coral decide a change is safe to apply automatically?

- Only workloads with "Automatically optimize" on are changed. Zoorik's own namespace is never changed.
 - Through a policy, cuts are capped in the first 24 hours of data (question 9). After that Coral needs at least 90% of the expected samples.
 - Every change respects your minimums, LimitRanges and quotas. Routine changes under 10% are skipped.
 - Pods are resized in place where possible. Otherwise they are replaced one at a time, within your PDBs.
-

9 Is there a warm-up period before automation?

- For pod sizing, yes. When you turn it on through a policy, cuts in the first 24 hours of data are capped at 10%, then 25%, then 35%.
- Workloads that already have 24 hours of good data get the full recommendation straight away.
- Turning on a single workload directly skips this warm-up.
- Node optimization has no warm-up. It starts at install.

10 Memory spikes can cause OOM kills. How much safety buffer is kept?

- Memory is set from the highest usage of the last 8 days, plus 10% (15 to 25% on some other policies).
- Coral never lowers a memory limit on regular pods, so a container with a limit is OOM-killed at the same point.
- A container with no memory limit can be evicted sooner if the node runs short of memory.
- If an OOM kill still happens, Coral raises memory straight away (next question).

11 What happens if an optimized pod starts getting OOM-killed?

- Coral raises that container's memory by 20% per OOM kill (at least 100 MiB), straight away.
- Crash-looping pods are recreated at the new size, and the extra margin fades over 24 hours.
- Raises stop at 2.5× the base size. If the kills look like a memory leak, Coral stops raising and holds that cap for 24 hours.

12 If optimization causes performance degradation, how is it rolled back automatically?

- OOM kills, failing probes and CPU pressure make Coral raise requests back up automatically.
- If a pod can't be scheduled at its new size, Coral caps the request to fit.
- A replacement pod that isn't Ready within 10 minutes goes back to its original size (except in StatefulSets).
- Coral doesn't read latency or error rates.
- To roll back by hand, turn automation off. New pods start at your manifest's size; running pods keep theirs until they restart.

SECTION 3

Stateful and single-replica workloads

13 How are databases, Kafka, Elasticsearch, Redis and StatefulSets optimized?

- Coral changes only CPU and memory, never app settings such as JVM heap or Redis maxmemory.
- StatefulSet pods are resized in place, or replaced one at a time, highest ordinal first, within the PDB.
- When a node is removed, a StatefulSet pod gets a short restart within its PDB. Nodes with local data or do-not-disrupt pods are kept.
- For a careful start, put databases on the Read-only policy and mark them do-not-disrupt.

14 Can optimization happen without downtime for applications that have only one replica?

- Usually, yes. Coral resizes the pod in place (Kubernetes 1.33+), or starts a new pod and waits until it's Ready before removing the old one.
- Where that isn't safe (for example Recreate strategy or a ReadWriteOnce volume), the new size waits for the pod's next natural restart.
- When a node is removed, a pod whose replacement can't start first gets a short restart. Mark it do-not-disrupt to prevent that.

Working with HPA

15 If the HPA wants to scale up while Coral wants to reduce CPU requests, which one wins?

- There is no conflict. The HPA decides how many pods run, and Coral decides how big each pod is.
- While the HPA is scaling and below its max, Coral holds back its surge and CPU-pressure raises (OOM raises still apply), and pauses pod replacements until every pod is Ready.
- For CPU-based HPAs, Coral picks the request by replaying load through your HPA's own settings (next question).

16 Could reducing CPU requests make the HPA scale more aggressively?

- It could, because HPA utilization is usage divided by request.
- So for workloads with a CPU-based HPA, Coral replays the last 3 days of load through your HPA's own settings.
- A smaller request wins only if its saving beats the cost of unserved load and extra scaling.

17 Does optimization ever modify HPA minReplicas or maxReplicas?

- Only on HPAs Coral manages: ones it creates for safe Deployments, with minReplicas of 2 or more, and ones you created by hand that it takes over.
- On an HPA it takes over, Coral keeps minReplicas, only raises maxReplicas, and restores your values when it lets go.
- An HPA that Coral creates can scale a steady Deployment down to its minimum.
- It never changes min or max on HPAs from Helm, Argo CD, Flux, KEDA or kubectl apply.

Nodes and spare capacity

18 If workload optimization reduces requests, how soon are nodes consolidated and removed?

- Within minutes of the running pods carrying the smaller requests. Coral checks every 5 minutes and drains nodes gradually, never in bulk.
- An emptied node waits 5 minutes (you can change this), then your autoscaler deletes it. Coral deletes it itself only in fixed-size pools that nothing else manages.
- Pools that Karpenter or NAP already consolidates are left to them.

19 How do you make sure node consolidation doesn't cause scheduling failures later?

- Before draining, Coral simulates the scheduler on the live cluster. Every pod must fit elsewhere, with its affinity, taints, spread, ports and zones.
- It respects PDBs, do-not-disrupt and your autoscaler's minimum pool size.
- Room for future growth is your autoscaler's job.
- Known gap: per-node disk-attach limits aren't checked yet.

20 How much spare capacity is kept for sudden workload growth?

- Coral keeps no idle spare nodes. The headroom sits inside each pod's request.
- Sudden growth is handled by your autoscaler adding nodes.
- To keep warm spare capacity, run a low-priority placeholder deployment. Put it on the Read-only policy so Coral doesn't shrink it.

21 During cluster upgrades, is optimization paused or does it continue?

- It continues, but steps around the upgrade. It skips cordoned, NotReady or deleting nodes, and never drains a node someone else cordoned.
- We recommend pausing node optimization and the Evictor during the upgrade window.

SECTION 6

Autoscalers and traffic surges

22 How does Coral interact with Cluster Autoscaler or Karpenter decisions?

- Coral never changes your autoscaler's settings and never adds nodes for demand.
- Karpenter, NAP or the Cluster Autoscaler deletes the nodes Coral empties, within your autoscaler's minimums.
- Pools that Karpenter or NAP already consolidates are left to them. Coral deletes VMs itself only in fixed-size pools that nothing else manages.

23 After node and workload optimization move us from larger machines to smaller ones, how do we get back to bigger machines in a sudden surge?

- Coral never adds capacity for demand. Scale-up belongs to your Karpenter, NAP or Cluster Autoscaler, and Coral doesn't change or fight it.
- In a surge, the HPA adds pods, pending pods trigger your autoscaler, and Karpenter or NAP can pick bigger machines.
- If you have no autoscaler, we set up NAP (AKS) or Karpenter (EKS) during onboarding, with your approval.
- If one can't be set up (for example AKS with kubernetes networking), nothing adds nodes until an autoscaler exists.
- Coral moves workloads to smaller machine types only with Karpenter or NAP. With only the Cluster Autoscaler, Coral removes nodes but never changes their size.

SECTION 7

Getting started

24 Can optimization be restricted to one namespace or application first?

- Yes. Create a policy with automation on and assign it to a namespace or by labels. Everything else stays recommendation-only.
- That policy can also create HPAs. Turn HPA optimization off in it if you want sizing only.
- Node optimization is scoped by node pool, not namespace. Enable only the pools you want, and start with the Evictor in dry-run.

Find your own number.

Run the free assessment on your own cluster.

zoorik.com/assessment →

Talk to the builders.

Pick a time at cal.com/zoorik/talk, or write to founders@zoorik.com.