

KUBERNETES OPTIMIZATION ON AKS AND EKS

Coral product guide

How to connect a cluster, what Coral changes, and how you stay in control.

- **What Coral does**

Right-sizes CPU and memory requests, removes nodes you don't need, and moves work to cheaper machines.

- **What Coral never does**

Create machines itself, change your pod templates or Git sources, or read Secret objects or pod logs through the Kubernetes API.

- **You stay in control**

Nothing is resized until you switch on "Automatically optimize". Every engine has its own switch.

Node optimization and the Evictor start at install. For a careful start, see section 8. This guide describes Coral as released in October 2026.

- | | |
|--|--|
| 01 How Coral works | 02 Before you start |
| 03 Connect a cluster | 04 What gets installed |
| 05 The console | 06 Workload optimization |
| 07 Node optimization | 08 Stay in control |
| 09 Savings and figures | 10 Security and data |
| 11 Disconnect | |

SECTION 1

How Coral works

Coral rebalances your Kubernetes clusters. It right-sizes workloads and removes nodes you don't need, so you pay for what you use. Cost monitoring is always on.

1 Connect

One command installs the Coral agent in your cluster. The command asks before it changes anything in your cloud.

2 Watch

The agent reads usage every 30 seconds and sends 5-minute summaries. Figures appear within minutes.

3 Recommend

Coral works out the right size for every workload, the nodes you don't need, and what that saves.

4 Optimize

Coral empties spare nodes for your autoscaler to remove. When you switch on automation, it also applies the new sizes.

Two engines

ENGINE	WHAT IT DOES	HOW IT STARTS
Workload optimization	Right-sizes each container's CPU and memory requests, and can manage HPAs.	Recommendations only, until you switch on "Automatically optimize".
Node optimization	Empties nodes you don't need so your autoscaler can remove them. Rebalances onto cheaper machines when you ask.	On from install, together with the Evictor, which moves pods off lightly used nodes.

SECTION 2

Before you start

Check these before you connect a cluster.

YOU NEED	DETAILS
A supported cluster	AKS on Azure or EKS on AWS, on Kubernetes 1.30 or later, with at least one Linux node (amd64 or arm64). Other clusters connect as "Local", without cloud prices.
Linux nodes for optimization	Coral's parts run on Linux nodes. Windows nodes are listed and costed, but their workloads are not right-sized, rebalanced or live-migrated.
Nodes on 1.33 or later	Only for resizing running pods in place. On older nodes, Coral applies a new size by restarting pods one at a time, within their PodDisruptionBudget.
Cluster access	cluster-admin on the cluster, through a kubeconfig context.
Command-line tools	The Azure CLI (az) for AKS, or the AWS CLI (aws) for EKS, signed in. Helm 3.8 or later only if Karpenter has to be installed.
Cloud rights	AKS: Owner, or Contributor with User Access Administrator, on the cluster's subscription. EKS: the rights to create IAM roles and EKS Pod Identity associations. If Karpenter is installed, also SQS, EventBridge and cluster updates.
Network	Outbound HTTPS to Zoorik; to ghcr.io and pkg-containers.githubusercontent.com for the images; and to your cloud's APIs for node removal (Azure Resource Manager and Entra ID, or AWS STS and Auto Scaling).
A console account	Sign-in with Microsoft or Google, and permission to connect clusters.

SECTION 3

Connect a cluster

Connecting takes one command. `zoorikctl`, Zoorik's command-line tool, asks before every cloud change.

1 Sign in

Go to `zoorik.com/login`, choose Coral, and continue with Microsoft or Google.

2 Open "Connect cluster"

On the Overview page, click "Connect cluster". Node optimization and Workload optimization start switched on. Workload optimization only recommends until you switch on "Automatically optimize". For a careful start, read section 8 before you continue.

3 Install zoorikctl

```
# macOS
brew tap getzoorik/tap
brew trust getzoorik/tap
brew install zoorikctl

# Linux
curl -fsSL https://get.zoorik.com/linux | sh

# Windows (PowerShell)
irm https://get.zoorik.com/windows.ps1 | iex
```

4 Run the connect command

Copy it from the dialog. Add `--kube-context <name>` to pick the cluster. Its connect token works once and expires after 60 minutes.

```
zoorikctl cluster connect --connect-token="zoorik_v1_..." --api-region="us"
```

5 Approve the cloud changes

zoorikctl lists every cloud change and asks once. It then installs the agent and waits for it to register. The cluster appears on Overview, and its figures follow within minutes.

Review it first

Add `--dry-run` to print every cloud command and every object the agent setup installs, without changing anything.

What connect asks to change in your cloud

CHANGE	WHEN	WHAT IT SETS UP
Permission to remove nodes on AKS	Every AKS connect	A managed identity, <code>zoorik-agent-<cluster></code> , with the role "Zoorik Node Removal v1" on this cluster only. The role can read the cluster, its pools and machines, and delete machines. Connect also turns on workload identity and the OIDC issuer if they are off.
Permission to remove nodes on EKS	Every EKS connect	An IAM role, <code>zoorik-agent-<cluster></code> , bound with EKS Pod Identity. It can only terminate instances in this cluster's Auto Scaling groups. The Pod Identity Agent add-on is added if missing.
A node autoscaler on AKS	Only if the cluster has none	Turns on Node Auto Provisioning (NAP), which needs Azure CNI (not kubenet), and adds the node class <code>zoorik-base</code> .
A node autoscaler on EKS	Only if the cluster has none	Two IAM roles, an SQS queue with four EventBridge rules, discovery tags on the subnets and security group, an access entry, Pod Identity, Karpenter 1.8.1, the node classes <code>zoorik-base</code> and <code>default</code> , and a default NodePool.

- On AKS, the OIDC issuer can't be turned off again once it is on.
- On an EKS cluster that uses only `aws-auth`, installing Karpenter switches authentication to `API_AND_CONFIG_MAP`. That can't be undone.
- If you decline the cloud changes, the agent still installs. Pools run by an autoscaler are optimized as usual. Fixed-size pools wait until you run `zoorikctl cluster grant-node-deletion`.
- To set up an autoscaler later, run `zoorikctl cluster prepare`.

SECTION 4

What gets installed

Everything the agent setup installs is labelled `app.kubernetes.io/part-of=zoorik-agent`.

COMPONENT	WHAT IT DOES
zoorik-agent Deployment, namespace <code>zoorik-agent</code>	Reads the cluster, sends usage summaries and carries out Coral's changes. Runs as non-root with a read-only filesystem.
zoorik-agent-webhook Deployment, 2 replicas	Sizes new pods as they start, and fails open, so it never blocks a pod from starting. It also refuses hand edits and deletes of the HPAs Coral manages; change those through the policy or "Hand back".
zoorik-node DaemonSet, namespace <code>zoorik-node</code>	The live-migration daemon, on every Linux node: it checkpoints a pod and restores it on another node. It runs privileged (root, host PID, SYS_ADMIN) and places CRIU, the open-source checkpoint and restore tool, at <code>/usr/local/bin/criu</code> .
CRDs and admission policies	<code>recommendations.coral.zoorik.com</code> and <code>actionrequests.coral.zoorik.com</code> store Coral's recommendations and action requests. Admission policies limit what the agent may change.

Rebalancing pools are created later, and are labelled `coral.zoorik.com/owned-by=rebalancing`.

Updates

- The agent updates itself to each Zoorik release, and rolls back a failed update.
- It can't widen its own permissions. A release that needs more shows "Re-run connect" on the Dashboard's "Agent release" card.
- If Argo CD, Flux or Helm manages the agent, the agent doesn't update itself.

SECTION 5

The console

Overview lists every cluster. Pick a cluster to see its pages.

PAGE	WHAT IT IS FOR
Overview	Every connected cluster, its running cost, what is recoverable, and its score. "Connect cluster" starts here.
Dashboard	The cluster's status, capacity, nodes, pods, OOM kills and agent release.
Optimization constraints	Workload setups that make optimization risky (health probes, PodDisruptionBudgets, topology spread), with the fix for each. Advisory only.

WORKLOAD OPTIMIZATION

Optimization	Every workload's request against its recommendation, confidence, savings and status. Switch on "Automatically optimize" for one workload.
Scaling policies	Built-in and custom policies. Switch on "Automatically optimize" for a policy.
Impact	What resizing has already cut from requests, at list price.
Event log	Everything Coral did or saw, newest first.

NODE OPTIMIZATION

Node list	Every node and its cost, and the nodes held, with the reason for each.
Rebalancing	Plan, run and schedule moves to cheaper machines.

PAGE	WHAT IT IS FOR
Configuration	"Wait before removing empty nodes", and a switch for each node pool.
Container live migration	Pods moved by warm handover during node draining, and what each move saved. Checkpoint moves are not counted here.

CLUSTER

Settings	The Workload optimization, Node optimization and Evictor switches, and "What this cluster supports".
----------	--

While data arrives, pages show "Waiting for the agent's first report..." and then "Calculating". They update by themselves.

SECTION 6

Workload optimization

Coral sets each container's CPU and memory requests from what it really uses.

How sizes are worked out

- **CPU (balanced policy):** the 80th percentile (P80) of the last 3 days, plus 10%.
- **Memory (balanced policy):** the highest use in the last 8 days, plus 10%.
- A usage surge or an OOM kill raises the size within minutes.
- With a CPU-based HPA, Coral picks the request by replaying past load through your HPA's own settings.
- Your manifest is never changed. Coral sizes pods as they start. It applies a new size by resizing running pods in place on 1.33+ nodes, or else by restarting them one at a time.

Turn it on

WHERE	HOW	WHAT HAPPENS
On a policy (recommended)	Scaling policies → switch on "Automatically optimize" → "Start optimizing".	In the first 24 hours of data, cuts are capped at 10%, then 25%, then 35%. After that Coral needs at least 90% data coverage.
On one workload	Optimization → the workload's "Automatically optimize" → On.	Applies straight away, with no confidence wait.

Built-in policies

POLICY	USE IT FOR	SIZING
balanced	Most apps. The default for Deployments, DaemonSets and Argo Rollouts.	CPU P80 + 10%, memory peak + 10%
resiliency	Databases and stateful apps. The default for StatefulSets.	Like balanced; keeps Guaranteed pods Guaranteed
stability	Latency-sensitive apps.	CPU P90 + 15%, memory peak + 15%, one pod at a time
cost-savings	Dev and non-critical work.	CPU P70 + 10%
burstable	Jobs, CronJobs and spiky work. The default for Jobs and CronJobs.	CPU P60 of 8 days + 25%, memory peak + 20%
readonly	Recommendations only; never changes a workload. Always used for Zoorik's own namespace.	None

Good to know

- Every built-in policy except readonly has the Optimization type "Both": it sizes pods and manages HPAs. With automation on, Coral creates HPAs for eligible Deployments and Argo Rollouts, and takes over HPAs nothing else manages.
- On a built-in policy you can change only "Automatically optimize" and the Optimization type (Vertical or Both). Set it to "Vertical" to size pods only. Use "Duplicate..." for anything else.

Custom policies

- "Create scaling policy" starts from balanced. You can change the percentile (up to P99.9 or max), headroom, look-back (3 hours to 8 days), minimums and maximums, limits, startup CPU and rollout.
- Assign it by namespace names (patterns such as prod-*), namespace labels, workload kinds or workload labels.
- Custom policies are checked top to bottom, and the first match wins.

How a workload gets its policy

Coral uses the first of these that applies:

- 1 Zoorik's own namespace, and add-ons labelled `addonmanager.kubernetes.io/mode: Reconcile`, always get readonly. Other kube-system workloads, EKS add-ons included, follow the steps below.
- 2 A policy named in the workload's `coral.zoorik.com/configuration` annotation (see Per-workload settings).
- 3 A policy picked on the workload's page.
- 4 The first custom policy whose rules match.
- 5 The default for its kind.

Status words

STATUS	MEANING
Applied	Pods run at the recommended size.
Applying	Coral is rolling the new size out.
At next restart	The new size applies when the pod next restarts.
Waiting for confidence	Automation is on; Coral waits until it has enough data.
Recommendation only	Coral works out sizes but doesn't apply them.
No data yet	No usage has arrived for this workload yet.
Blocked	Coral can't apply right now; it retries by itself.
Stopped	Sizing is frozen by a safety hold, for example after repeated OOM kills.
Not supported	Coral can't size this workload; the reason is shown.

Per-workload settings

Add the `coral.zoorik.com/configuration` annotation to the workload. Set only what you want to change.

```
metadata:
  annotations:
    coral.zoorik.com/configuration: |
      scalingPolicyName: stability
      vertical:
        optimization: on
        cpu:
          target: p95
          min: 250m
        memory:
          max: 4Gi
        excludedContainers:
          - istio-proxy
      horizontal:
        optimization: off
```

A workload with this annotation is managed from its manifest, so its console controls are locked.

HPAs and VPAs

- Coral creates an HPA only for a Deployment or Argo Rollout that has none. The workload needs 24 hours of history, at least 100m CPU per pod, and 2 or more ready replicas. The new HPA's minReplicas is 2 or more.
- It takes over only HPAs nothing else manages. It keeps minReplicas, only raises maxReplicas, and restores your values on "Hand back".
- It never changes HPAs from Helm, Argo CD, Flux, KEDA or kubectl apply.
- With automation on, Coral switches an active VPA to Off, and restores it when you switch off.
- Switching automation off removes the HPAs Coral created and hands back the ones it took over.

SECTION 7

Node optimization

Coral empties nodes you don't need. Your autoscaler removes them. Coral never creates machines itself.

Consolidation

- Coral uses pod requests to work out how many nodes each pool needs. Every pool keeps at least one node.
- Before it moves anything, the agent checks on the live cluster that every pod on the node fits on another Ready node.
- It cordons and drains one node at a time. If a drain can't finish in 10 minutes, the node goes back into service.
- After "Wait before removing empty nodes" (5 minutes by default), Coral hands the empty node to your autoscaler, whose own timers decide when it goes. Coral deletes the machine itself only in fixed-size pools that nothing else manages.
- Pools that Karpenter or NAP already consolidates are left to them.

The Evictor

- Moves pods off lightly used nodes, in pools where empty nodes get removed. It never deletes a machine and never overrides a PodDisruptionBudget.
- Runs inside your cluster and checks every 60 seconds by default.
- On from install, with its own switch in Settings → General. Switching off Node optimization does not stop it.
- "Dry run" lists the nodes it would empty in the Event log, without moving any pod.

How pods are moved

Coral moves a pod in the first of these ways that works:

- 1 **Warm handover:** a replacement pod starts and becomes Ready on another node, then the old one is evicted.
- 2 **Live migration:** where both nodes passed a live-migration check, the pod is checkpointed and restored on the other node. Clients see a short pause, longer for pods with more memory.
- 3 **Eviction:** within the pod's PodDisruptionBudget.

Warm handover isn't possible for StatefulSets, Argo Rollouts, workloads with the Recreate strategy, pods with ReadWriteOnce volumes, GPU pods or HPA-scaled pods. Without live migration, these pods get a short restart.

Nodes held

A held node is one Coral won't empty, because a pod on it can't be moved safely. Node list → "What's holding them" shows each reason and its fix. The common ones:

REASON	FIX
The PodDisruptionBudget allows no disruption.	Add replicas or relax the PodDisruptionBudget.
A kube-system pod has no PodDisruptionBudget.	Add a PodDisruptionBudget that allows at least one disruption.
Data on the node's own disk (emptyDir or hostPath).	Move the data to a persistent volume, or mark the pod safe to evict.
Nothing would recreate the pod.	Run it under a Deployment or StatefulSet.
Marked do-not-disrupt, not safe to evict, or excluded.	Remove the marker if moving it is fine.

Rebalancing

Rebalancing replaces the nodes you choose with cheaper machines that still fit the work.

- **Needs:** Karpenter or NAP, Node optimization on, and the permission to remove nodes.
- **Steps:** "Prepare new plan" → choose nodes → "Generate plan" → compare current and rebalanced cost → "Rebalance". A plan expires after 1 hour.
- **How it runs:** Coral creates fixed-size NodePools named zoerik-rb-.... Your Karpenter or NAP launches the new machines. Then the old nodes are drained and removed one by one. The new pools stay after the run.
- **"Evict nodes gracefully"** is off by default. While it is off, pods still on a node after 20 minutes of normal eviction are deleted, even if their PodDisruptionBudget doesn't allow it. Switch it on to keep such a node instead.
- **Schedules:** a schedule runs every day at 02:00 UTC and starts a rebalance by itself when its savings target is met.

SECTION 8

Stay in control

Every engine has its own switch, and any pod or node can be protected.

Switches

SWITCH	WHERE	DEFAULT	WHEN IT IS OFF
Automatically optimize	Scaling policies, or a workload	Off	Recommendations only.
Workload optimization	Settings	On	Nothing is resized or scaled. Recommendations are still shown.

SWITCH	WHERE	DEFAULT	WHEN IT IS OFF
Node optimization	Settings	On	Coral plans no node removal, and steps already under way are cancelled. The Evictor keeps running.
Evictor	Settings → General	On	The Evictor moves no pods. Consolidation still drains nodes while Node optimization is on.
Node pool "Enabled"	Configuration → Node pools	On	Coral leaves that pool's nodes alone. A rebalance you start can still include them.
Wait before removing empty nodes	Configuration → Node scaling	5 minutes	Not a switch. Set it from 0 minutes to 7 days.

Labels and annotations

KEY	VALUE	TYPE, ON	EFFECT
coral.zoorik.com/ignore	"true"	Label: namespace, workload or pod template	Coral never changes its sizes or HPAs.
coral.zoorik.com/configuration	Settings	Annotation: workload	Per-workload settings (section 6).
karpenter.sh/do-not-disrupt	"true"	Annotation: pod or node	On a pod: never moved, and its node is kept. On a node: the node is kept.
cluster-autoscaler.kubernetes.io/safe-to-evict	"false"	Annotation: pod	Never moved, and its node is kept.
coral.zoorik.com/removal-disabled	"true"	Label or annotation: node or pod	Kept out of every Coral move.
coral.zoorik.com/live-migration	disabled	Annotation: pod	No live migration. The pod can still be moved by handover or eviction.
coral.zoorik.com/ignore-optimization-constraints	"true"	Annotation: pod template	Hidden from Optimization constraints only.

- A PodDisruptionBudget that allows no disruption keeps its pods, and their node, in place for consolidation and the Evictor. A rebalance respects it only when "Evict nodes gracefully" is on.
- The ignore label stops workload changes only (sizes and HPAs). It does not stop node moves.

A careful start

- 1 For recommendations only at first, switch off Node optimization in the Connect dialog. The Evictor still starts with the agent and can move pods, so do step 2 as soon as the cluster appears.
- 2 Open the Evictor's "Advanced settings" in Settings and tick "Dry run".
- 3 In Configuration → Node pools, switch off the pools you don't want touched.
- 4 Create a scaling policy for one namespace, set its Optimization type to "Vertical", and switch on "Automatically optimize".
- 5 Watch the Event log and the Impact page, then widen step by step: more namespaces, more node pools, and Node optimization back on.

Savings and figures

Every figure says where it comes from. Coral never shows a made-up number.

FIGURE	WHERE	WHAT IT MEANS
Running cost today, At the optimum, Recoverable	Overview	Monthly cost now, after optimization, and the difference.
Score	Overview	$10 \times \text{optimal} \div \text{current monthly cost}$. 10 means nothing is left to recover.
Estimated monthly savings	Optimization	Compute saved if every recommendation is applied.
Already removed	Impact	The list-price value of the requests resizing has already cut. It reaches your bill once the freed capacity lets nodes go.
Fleet cost, Unclaimed capacity, Nodes held	Node list	Every node's monthly price, the capacity no pod has claimed, and the nodes Coral can't empty.

- Prices are the cloud's published list price for the region, refreshed nightly. Spot nodes are priced at the spot rate.
- Discounts, reservations and savings plans are not included. A month is 730 hours.
- Until data exists, figures read "Calculating" or "Fetching prices", never a guess.

Security and data

The agent opens every connection itself. It sends summaries out, and Zoorik's replies carry its instructions, recommendations and updates.

What leaves your cluster

- Usage summaries per container: CPU, memory, OOM kills, probe failures and CPU pressure, in 5-minute windows.
- An inventory every 5 minutes: nodes, namespaces, workloads with their requests and limits, autoscalers, PodDisruptionBudgets, quotas, and up to 64 labels per workload and namespace.
- Events, with pod and node names: OOM kills, probe failures, spot interruptions, and Coral's own actions, including the pods the Evictor moved.
- A heartbeat every 60 seconds.
- At registration, the cloud account: the Azure subscription and resource group, or the AWS account and region.

What never leaves

- The agent can't read your Secrets or ConfigMaps, and has no pods/log, pods/exec or pods/portforward permission. It holds get on nodes/proxy to read kubelet metrics. That permission can also reach the kubelet's log and exec endpoints, which Coral never calls.
- No env var values, commands, image names or request payloads are sent. The only event text sent is the message when a pod can't be created because of a quota, a limit range or an admission policy.

How it connects

- The agent opens every connection to Zoorik itself, over HTTPS, and accepts none from outside the cluster. Inside it, the API server calls the webhook, and the node daemons talk to each other over mutual TLS.
- The connect token works once, for 60 minutes. The agent swaps it for a cluster credential that rotates by itself.
- Admission policies limit the agent's changes. For example, it may scale a Deployment by only one replica, and change only the HPAs and NodePools Coral manages.

The live-migration daemon

- `zoorik-node` runs privileged on every Linux node, so it can checkpoint and restore containers.
- A checkpoint holds the pod's memory, secrets included. It stays inside your cluster, moves between the two nodes over mutual TLS, and is deleted when the move ends.

Sign-in

Sign in with Microsoft or Google. Zoorik never sees your password. A session ends after 30 minutes of inactivity, or after 12 hours.

SECTION 11

Disconnect

Disconnect from the command line, with your own credentials.

```
zoorikctl cluster disconnect --kube-context <name>
```

- Add `--dry-run` to see everything it would remove first.
- Disconnect works even if the agent is down, and is safe to run again.
- It removes the agent, the webhook, the live-migration daemon, Coral's roles, policies and CRDs, and Coral's taints and annotations on nodes.
- It deletes the HPAs Coral created, hands back the HPAs it took over, and restores the VPAs it switched off.
- It offers to remove the cloud identity and role assignment it created.
- Running pods keep their current sizes until they restart.

What stays

- The node autoscaler set up during connect.
- Coral's `zoorik-base` node class and `zoorik-rb-...` rebalancing pools.
- The CRIU files on nodes, and the folder `/var/lib/zoorik/checkpoints`, which can hold a checkpoint from an interrupted move. Delete it.
- The "Zoorik Node Removal v1" role definition, workload identity, the OIDC issuer and the Pod Identity Agent add-on.
- The cluster's entry in the console, shown as "Not reporting", and the data Zoorik has already received.

Find your own number.

Run the free assessment on your own cluster.

zoorik.com/assessment →

Need help?

Pick a time at cal.com/zoorik/talk, or write to founders@zoorik.com. If a connect fails, `zoorikctl` prints the path of a redacted log to send us.