APPETIZER LABS

Kubernetes Production Readiness Checklist

This checklist makes twenty-four statements about a cluster that already serves production, grouped into the eight areas we use to define “production-ready”. You tick what is true today. Next to every statement that stays open is the first step you can take without a budget and without us.

Reviewed: 2026-08-17 https://appetizers.io/en/kubernetes-production-readiness-checklist/

Who it is for

For the people who run the cluster and for the people who answer for it: platform and operations teams, engineering leads, and whoever has to decide whether the next application is allowed anywhere near it. Work through it with two or three people — someone from operations, someone from development, and whoever gets woken up at night.

How to use it

Only tick what is already true — not what is in a ticket, planned, or sitting in a repository nobody has touched for months. Answer for production, not for the best environment you own. If the room disagrees about a statement it counts as open: two people answering it differently is itself a finding.

What depends on your version (Reviewed: 2026-08-17)

Some of these depend on the Kubernetes version you run: APIs disappear with minor releases, and what still works in one version is gone in the next. So the notes on this page say when something changed — not which versions still get patches today. The project's own release page answers that, and it answers it more currently than a page of ours ever could. The date on each note is when we last reviewed the criteria.

How these criteria were reviewed

The items come from running clusters in production and were read back against the project's published documentation; the sources are below. They were reviewed by the person who also does the work — no external body was involved, and this is our standard rather than an industry one. If you think an item is wrong, tell us: a reasoned correction changes the list.

Kubernetes — Releases, patch support and release cadence
https://kubernetes.io/releases/

Kubernetes documentation — Deprecated API migration guide
https://kubernetes.io/docs/reference/using-api/deprecation-guide/

Kubernetes documentation — Pod Security Admission
https://kubernetes.io/docs/concepts/security/pod-security-admission/

Kubernetes documentation — Configure Liveness, Readiness and Startup Probes
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

01

Upgrades are planned work

- [ ] You know, without looking it up, which minor version your cluster runs and when its patch support ends.

IF THIS ONE IS OPEN

Put the cluster version and its end of support in the same document as your other deadlines. On a managed offering its own schedule applies as well — and that one is usually the earlier of the two.

VERSION NOTE

The project ships a minor release roughly every four months and patches each one for about fourteen months. Managed providers set their own, often shorter, deadlines on top of that.

- [ ] Before every upgrade you check your manifests for APIs removed in the target version — with a tool, not from memory.

IF THIS ONE IS OPEN

Run pluto or kubent once against what you have today. What comes out is the work list for your next upgrade, and it does not get shorter by waiting.

VERSION NOTE

APIs go away with minor releases: PodSecurityPolicy, policy/v1beta1 PodDisruptionBudget and batch/v1beta1 CronJob were removed in 1.25. What your target version drops is in its own migration notes.

- [ ] An upgrade runs first where a failure wakes nobody, including the add-ons that have to move with it.

IF THIS ONE IS OPEN

Write down your add-ons — CNI, ingress controller, cert-manager, CSI drivers, monitoring — and the cluster versions each supports. The narrowest range decides your upgrade path, not the cluster.

02

The desired state lives in Git

- [ ] Every object in production comes from a repository, and nobody changes anything there by hand for good.

IF THIS ONE IS OPEN

Compare what runs in the cluster with what the repository says once. Whatever differs is either drift or an intention nobody wrote down — both belong on the same list.

- [ ] Differences between the repository and the cluster are noticed without anyone going looking for them.

IF THIS ONE IS OPEN

Turn on the drift view in your delivery tooling and decide who reads it. A view that belongs to nobody is not a view.

- [ ] Credentials are not sitting in the repository in the clear; the path from the secret store into the cluster is automated and written down.

IF THIS ONE IS OPEN

Search the repository for secret manifests with values embedded in them. If you find some, the first move is not rotation but deciding where secrets come from in future.

03

Recovery has been rehearsed

- [ ] Losing a node has been rehearsed: you drained one during working hours and watched what briefly stopped answering.

IF THIS ONE IS OPEN

Schedule draining a node in a quiet hour and watch. If a service drops out, it is missing replicas, a PodDisruptionBudget, or spreading across failure domains.

- [ ] A restore from backup has been rehearsed into a fresh cluster, with the date and the outcome written down.

IF THIS ONE IS OPEN

Take the most recent backup and restore a single namespace into a test environment. How long that takes is your actual recovery time; everything else is an estimate.

- [ ] State outside etcd is accounted for: volumes, databases, and everything a manifest does not bring back.

IF THIS ONE IS OPEN

Go through your StatefulSets and PersistentVolumeClaims and note beside each who backs the data up and where to. Anything left blank is unprotected, however often the cluster itself is backed up.

04

Access is answerable

- [ ] People authenticate to the cluster as themselves, not through a shared account or a kubeconfig passed around.

  IF THIS ONE IS OPEN

  Count the kubeconfig files with administrative rights in circulation. Each one is an entrance with no name on it, and it survives the person leaving.

- [ ] Who may deploy to production is answerable in one sentence and provable from the roles in the cluster.

  IF THIS ONE IS OPEN

  List the bindings to the cluster administrator role and write down why each exists. Whatever cannot be justified is the first candidate for removal.

- [ ] Workloads run under a policy that rules out privileged containers, and exceptions are justified one by one.

  IF THIS ONE IS OPEN

  Put Pod Security Admission on one namespace in warning mode first and read what it flags. Enforce once that list is short.

  VERSION NOTE

  PodSecurityPolicy was removed in 1.25 and Pod Security Admission is the built-in successor. Clusters that only switched the old policy off at the time usually have no policy here at all today.

05

Observability answers questions

- [ ] Every important service has a view from the user's side — error rate, latency, saturation — not only utilisation per pod.

  IF THIS ONE IS OPEN

  Take your most important service and answer, from the views you already have, whether it is working for users right now. If that takes several dashboards, exactly one view is missing.

- [ ] Logs are central, searchable and kept for a decided period; nobody needs access to a node to look something up.

  IF THIS ONE IS OPEN

  Find the last incident in your logs without connecting to a node. Where that fails, either the collection or the retention is missing — and both are discovered too late by definition.

- [ ] You can see how much capacity is free and whether requested and actually used resources have drifted apart.

  IF THIS ONE IS OPEN

  Compare requested resources against real usage for two or three workloads. The gap explains your cloud bill and your scheduling at the same time.

06

Every alert has a recipient

- [ ] Every alert names a recipient and a runbook; alerts with neither are switched off rather than muted.

  IF THIS ONE IS OPEN

  Go through the alerting rules and delete the ones nobody acted on last quarter. That is not a loss of safety, it is the end of a habit.

- [ ] Alerts report symptoms somebody notices, rather than every threshold that was crossed.

  IF THIS ONE IS OPEN

  Count the alerts from the last on-call week and mark which ones led to an action. The ratio tells you whether your rotation is working or just awake.

- [ ] The on-call rotation is named, reachable and practised: the escalation path and the stand-in are written down.

IF THIS ONE IS OPEN

Ask the next person on call what they do first when the ingress controller fails. If the answer is a name rather than a step, the runbook is missing.

07

Workloads arrive through a documented path

- [ ] A new workload gets its namespace, resource settings, network rules and delivery path through a documented procedure rather than by asking around.

  IF THIS ONE IS OPEN

  Write down what actually happened during the last onboarding and turn it into a template. The second run shows you what the first version left out.

- [ ] Every workload declares its resource requests, and whatever has to stay available runs more than once and spread out.

  IF THIS ONE IS OPEN

  Find the deployments with no resource requests. Without them the scheduler plans blind, and the first traffic spike distributes the consequences across your nodes at random.

- [ ] Probes tell the truth: readiness checks dependencies, liveness only restarts what a restart fixes, and slow starts have a startup probe.

  IF THIS ONE IS OPEN

  Look at the containers with the most restarts. A liveness probe that fires on an overloaded dependency amplifies an incident instead of resolving it.

08

At least two people can do it

- [ ] For every item on this list there are two people who can carry it out — demonstrably, because the second one has done it once.

  IF THIS ONE IS OPEN

  Take the step you find hardest to answer for and have the second person carry it out in the next fortnight, with the first one sitting beside them.

- [ ] Runbooks live where people look during an incident, and the last person to use one had not written it.

  IF THIS ONE IS OPEN

  Let the next small incident be handled by someone who does not know the runbook, with the experienced person watching. The questions they ask are your gaps.

- [ ] Cluster version, add-on versions, ownership per namespace and open exceptions live in one place that is current.

  IF THIS ONE IS OPEN

  Create one page holding those four facts and carry it forward at the next upgrade. If it is still accurate at the one after that, you have a procedure rather than a note.

Reading the result

Count the open items, not the ticked ones. Where they sit matters more than how many there are: three open items in recovery weigh more than three spread across eight areas, because together they describe an outage you do not come back out of.

> **0 to 3 open**
>
> Isolated gaps, usually in one place. That is maintenance: take the open items one at a time, in the order of the areas. And read the paragraph below — an empty list is not the same as production-ready.

4 to 10 open

A visible pattern. Check first whether the open items cluster in one or two areas. If they do, there is usually a shared cause, and fixing that is cheaper than ten separate fixes.

11 or more open

The cluster runs, but it is not carrying anything yet. Here the order matters more than the volume: first the way back — recovery, rollback, access — then the way forward. Automating first means automating a state you cannot restore.

What it does not do

This list is not an audit, not a certificate and not a sign-off. It cannot see your cluster, does not know your traffic and has no idea what an outage costs you — it only knows what you click. Every box ticked means these questions find nothing more. It does not mean your cluster is production-ready, because the questions missing here are precisely the ones nobody can ask without looking at the environment.

What to do with the open items

Next to every statement is the first step that needs no budget — start there, in the order of the areas. If you want a second opinion, write to info@appetizers.io with the open areas and a few sentences about your environment. The answer is allowed to be that you do not need us for it.