Move2Cloud

• WHITE PAPER — 2027 EDITION

CLOUD-NATIVE & KUBERNETES

Kubernetes in Production
The Complete Checklist

Running a cluster is not enough. This guide details the availability, security and observability controls that separate a development cluster from a platform genuinely ready for production.

11 pages · Self-assessment checklist, RBAC/network/container comparison and roadmap included

82%
of container users run Kubernetes in production (CNCF, 2026)

36%
cite security as the main barrier to adoption (CNCF)

25%
of cloud incidents involve a misconfiguration (IBM X-Force)

PREPARED BY
Move2Cloud — Cloud & Infrastructure Consulting, Paris

PUBLICATION
September 2026

CONTENTS

What you will find in this document

A six-step journey, from defining what "real" production readiness means to a concrete operational roadmap.

01 **Executive summary — the essentials on one page** p.3

02 **Why going live is the real test** p.4

03 **Availability: HA, updates, graceful shutdowns** p.5

04 **Security: RBAC, network, containers** p.6

05 **Observability and application health** p.7

06 **Resources, storage and autoscaling** p.8

07 **Self-assessment grid** p.9

08 **A 5-step roadmap** p.10

09 **Conclusion & about Move2Cloud** p.11

> **Who is this document for?** Platform, SRE, DevOps teams and CISOs preparing — or consolidating — a Kubernetes cluster intended to host critical workloads.

Copyright Move2Cloud — move2cloud.fr

EXECUTIVE SUMMARY

The essentials on one page

Kubernetes has become the de facto standard for running production workloads — but "it runs" and "it is production-ready" are two different things.

According to the 2026 edition of the annual CNCF survey, **82% of container users now run Kubernetes in production**, up from 66% in 2023. Technical adoption is no longer the main obstacle: **47% of organisations cite cultural change on the development side** as the leading barrier, ahead of training (36%) and security (36%).

This cultural shift has an operational flip side: many clusters go live with gaps that stay invisible as long as nothing breaks — no PodDisruptionBudget, overly permissive RBAC, poorly calibrated health probes, or undefined resource requests and limits. These are precisely the gaps that misconfiguration — rather than the exploitation of a vulnerability — turns into an incident: **25% of cloud security incidents involve a misconfigured service**, according to the IBM X-Force Threat Intelligence Index.

For a team preparing a production rollout — a first critical cluster or the consolidation of an existing one — the question is no longer whether Kubernetes is the right choice, but which availability, security and observability controls are genuinely in place before real traffic runs on it. This document offers a structured checklist to answer it.

82%
of container users run Kubernetes in production in 2026 (CNCF)

47%
cite team culture — not technology — as the leading barrier to adoption

25%
of cloud incidents stem from poorly managed configuration

For a team preparing a production rollout — a first critical cluster or the consolidation of an existing one — the question is no longer "is Kubernetes the right choice?" but "which availability, security and observability controls are genuinely in place before real traffic runs on it?". This document offers a structured checklist to answer it.

SECTION 02

Why going live is the real test

A Kubernetes cluster that works in a development environment mainly proves one thing: that the manifests apply without error. It says nothing about its ability to absorb a node upgrade, an availability-zone outage, a traffic spike or an unauthorised access attempt. Three gaps explain why so many production incidents originate in choices made — or not made — well before the first real deployment.

1. 1. Development hides partial failures

In development, a pod that restarts or a request that fails once often goes unnoticed. In production, those same symptoms signal insufficient replication, poorly calibrated health probes or improper shutdown handling — gaps that only surface under real load.

2. 2. Kubernetes' default security remains permissive

Kubernetes defaults favour ease of getting started, not production security: open RBAC, no network policies, containers running as root unless something forbids it. Nothing prevents a cluster from running perfectly while being broadly exposed.

3. 3. Operational complexity grows with scale

A cluster with three services is easy to watch with the naked eye. A cluster with a hundred services is not: without structured observability (metrics, logs, traces) and without automated sizing, teams lose the visibility needed to spot drift before it becomes an incident.

> *“ A Kubernetes cluster is not production-ready because it runs — it is ready because it was designed to survive what will go wrong. ”*

The following sections detail, domain by domain, the concrete controls that close these three gaps.

Copyright Move2Cloud — move2cloud.fr

SECTION 03

Availability: HA, updates, graceful shutdowns

Availability is not declared — it is built through a set of settings that determine how the cluster reacts to a failure or an update.

CONTROL	WHAT TO PUT IN PLACE
Multi-zone replication	Deploy at least 3 replicas per critical service, spread across several availability zones, with anti-affinity rules so that all the pods of a given service do not end up on the same node.
PodDisruptionBudget	Define a disruption budget for every critical workload, to prevent a node update or a maintenance operation from taking down too many replicas at once.
Progressive rollouts and graceful shutdown	Use progressive deployment strategies (rolling update, blue/green or canary) rather than brutal replacements, with explicitly defined maxUnavailable/maxSurge thresholds, and handle the application SIGTERM signal properly with a terminationGracePeriodSeconds consistent with the real time needed to close in-flight connections.

> **Key takeaway:** A well-designed high-availability setup is verified by testing it — deliberately taking down a node or a zone in a pre-production environment — rather than by assuming it is in place because the manifest declares several replicas.

The single-replica trap

A service deployed with a single replica has, by definition, no fault tolerance: any pod failure, any node update, any planned restart causes a service outage. It is the simplest configuration gap to fix, and one of the most frequent in clusters that have not yet gone through a production-readiness audit.

SECTION 04

Security: RBAC, network, containers

Kubernetes security plays out at three distinct levels — who can act on the cluster, who can talk to whom, and what a compromised container can reach.

LEVEL	PRINCIPLE	EXPECTED CONTROL	ANSWERS THE QUESTION
RBAC	Least privilege	Namespace-scoped roles rather than broad ClusterRoleBindings, dedicated service accounts per application, regular review of the permissions granted.	Who can act on the cluster
Network policies	Explicit rules	Define explicit NetworkPolicy objects rather than leaving east-west traffic completely open by default — a compromised pod must not be able to reach the rest of the cluster freely.	Who can communicate with whom
Container hardening and secrets	Non-root	Run containers as a non-root user, read-only root filesystem where possible, Linux capabilities reduced to the strict minimum, images scanned before deployment; never store secrets in clear text in manifests, use etcd encryption at rest and, ideally, an external secrets manager integrated with the cluster.	What a compromised container can reach

Common pitfall: According to the IBM X-Force Threat Intelligence Index, 25% of cloud security incidents stem from a poorly controlled configuration — not from a novel vulnerability. The four controls above cover the vast majority of the attack angles actually exploited against Kubernetes clusters.

The Kubernetes dashboard, a door left open far too often

A dashboard or an API server exposed without strong authentication or network restrictions remains one of the most documented configuration errors on Kubernetes. The simple rule: no administration interface should be reachable without authentication, whatever the perceived sensitivity of the cluster.

Copyright Move2Cloud — move2cloud.fr

SECTION 05

Observability and application health

Without structured observability, a cluster at scale becomes impossible to diagnose other than in reaction to an incident already under way.

Properly calibrated health probes

Liveness, readiness and startup probes are not interchangeable: liveness restarts a stuck pod, readiness withholds traffic, startup shields slow-starting apps. Mis-set thresholds cause restart loops or late detection.

Three signals, not just one

Reliable observability combines metrics (error rate, latency, saturation), centralized logs and distributed traces. One signal alone leaves a blind spot: metrics show a problem exists, logs and traces show where and why.

Alerting rather than passive dashboards

Automatic notification as soon as a critical threshold is crossed, without waiting for a manual report. The goal is not to observe everything, but to detect within minutes what a user would otherwise report first.

COMPONENT	ROLE
Metrics (Prometheus)	Real-time drift detection — CPU, memory, latency, error rate per service
Centralized logs	Reconstructing the exact context of an incident, searching across pods and nodes

Copyright Move2Cloud — move2cloud.fr

SECTION 06

Resources, storage and autoscaling

A cluster with no resource limits defined works fine until the day a single faulty pod consumes the capacity of all the others.

Requests and limits, on every container

Every container should declare requests (what it is guaranteed) and limits (what it cannot exceed) for CPU and memory. Without these declarations, the scheduler can neither place pods intelligently nor protect the cluster from a service that would consume all the capacity available on a node.

The autoscaling trio

Three autoscaling mechanisms work together, each with a distinct role: the Horizontal Pod Autoscaler adjusts the number of replicas according to load, the Vertical Pod Autoscaler adjusts a pod's requests/limits over time, and the Cluster Autoscaler adjusts the number of available nodes. Deploying them in isolation rarely produces the expected result — it is their coherent combination that makes the cluster genuinely elastic.

> **Key takeaway:** Setting memory limits too low causes restart loops (OOMKilled) that are hard to diagnose if the logs keep no trace of the overrun. Always size limits from real measurements, not from an initial estimate.

For stateful workloads, PersistentVolumeClaims must be tied to a storage class suited to the real workload (IOPS, latency), with a backup policy that has been tested — not merely configured. A backup plan that has never been restored in a test is not a reliable recovery plan.

Copyright Move2Cloud — move2cloud.fr

SECTION 07

Self-assessment grid

Before opening a service to real traffic, check these eight fundamental controls.

- [ ] **Does every critical service run with at least 3 replicas spread across multiple zones?** A single replica means a guaranteed outage at the first node failure.
- [ ] **Does a PodDisruptionBudget protect every critical workload during node updates?** Without one, a planned maintenance operation can bring down an entire service.
- [ ] **Are liveness and readiness probes calibrated on real measurements rather than default values?** Copy-pasted thresholds generate needless restarts or detection that comes too late.
- [ ] **Does RBAC enforce least privilege, with no broad ClusterRoleBinding?** An over-permissive service account turns a compromised pod into full cluster access.
- [ ] **Do NetworkPolicies restrict east-west traffic between namespaces?** Without them, a single compromised pod can freely reach the rest of the cluster.
- [ ] **Does every container declare CPU/memory requests and limits?** Without them, a failing pod can exhaust the capacity of an entire node.

> **How to read your answers:** Three "no" answers or more across these eight controls indicate that a structured production-readiness review should precede any exposure to critical traffic.

Copyright Move2Cloud — move2cloud.fr

SECTION 08

A 5-step roadmap

A progressive approach, designed to secure an existing cluster without interrupting the services it already hosts.

1. **Audit**
   Put every critical service through the self-assessment grid on page 9, and document the identified gaps precisely.

2. **Secure first**
   Start by fixing over-permissive RBAC and the absence of network policies — these are the gaps that expose the cluster most widely if a pod is compromised.

3. **Harden availability**
   Add the missing replicas, define PodDisruptionBudgets, and test a deliberate node shutdown in pre-production.

4. **Instrument**
   Roll out the metrics/logs/traces trio on the services that still lack it, with actionable alerts rather than passive dashboards.

5. **Automate and document**
   Put combined autoscaling in place (HPA/VPA/Cluster Autoscaler) and formalise the production-readiness checklist for every new service to come.

Copyright Move2Cloud — move2cloud.fr

CONCLUSION

A cluster ready for production, not just for the demo

The difference between a cluster that works and a cluster that is production-ready is not visible in a successful deployment — it shows in how the cluster absorbs a node failure, an intrusion attempt or an unexpected traffic spike.

As Kubernetes adoption grows and AI workloads are added to it, the eight controls detailed in this document remain the foundation on which to build a reliable platform, whatever the team's starting level of maturity.

Move2Cloud	is a consulting firm specialised in infrastructure, cloud and IT transformation. We support organisations in the audit, the securing and production rollout of their cloud environment.
Contact	Move2Cloud — Paris · move2cloud.fr
Next step	A production-readiness audit puts every critical service through the eight controls on page 9, with a prioritised remediation plan.

Sources

CNCF, « Kubernetes Established as the De Facto Operating System for AI as Production Use Hits 82% in 2025 » — CNCF Annual Cloud Native Survey, 2026 edition — IBM X-Force Threat Intelligence Index (summary via deepstrike.io, « Cloud Security Statistics 2026 ») — Wiz, « Cloud Data Security Snapshot 2025 » — Official Kubernetes documentation (RBAC, NetworkPolicy, PodDisruptionBudget, autoscaling)

Document written by Move2Cloud as an example and illustration, for educational purposes. The recommendations presented are general; their implementation must be adapted to each organisation's technical and regulatory context.