Are You Really a Kubernetes Expert? Take This 10-Question Challenge to Find Out

Dheeman Das Follow 8 min read · 6 days ago

🧠 Introduction

Kubernetes has become the backbone of modern cloud-native applications. But let's face it — just deploying a few Pods and Services doesn't make someone a Kubernetes expert.

Whether you're a developer, DevOps engineer, SRE, or architect, mastering Kubernetes means understanding its deeper workings — from container orchestration and scaling to networking, probes, and

troubleshooting.

So here's a quick challenge:

> Can you answer 10 questions that test your real-world Kubernetes knowledge?

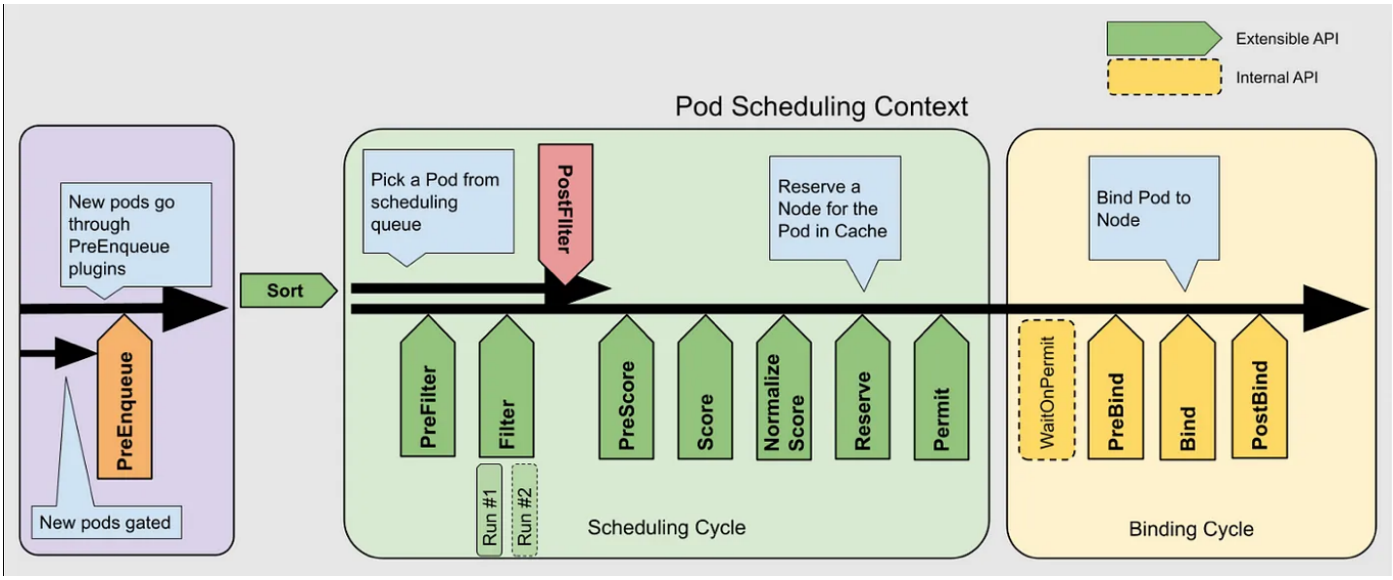
This isn't your average multiple-choice quiz — it's a **hands-on knowledge check** designed to separate casual users from true pros.

Let's see how much you really know about Kubernetes.
👇 Scroll down and start the quiz!

1. In a cluster with two nodes, one with pods and the other without, which node will a new pod be scheduled to?

🧠 Situation:

- You have **two shelves** (nodes).
- **Shelf A** already has some boxes (pods).
- **Shelf B** is completely **empty**.
- A **new box** (pod) comes in.
- Now, you ask: **Which shelf will it go to?**


Extensible API
Internal API
Pod Scheduling Context
New pods go through PreEnqueue plugins
PreEnqueue
New pods gated
Sort
Pick a Pod from scheduling queue
PostFilter
Reserve a Node for the Pod in Cache
Bind Pod to Node
PreFilter
Filter
Run #1
Run #2
PreScore
Score
Normalize Score
Reserve
Permit
WaitOnPermit
PreBind
Bind
PostBind
Scheduling Cycle
Binding Cycle

📦 How Kubernetes Thinks:

Kubernetes acts like a **smart organizer**. When a new box (pod) needs to be placed:

1. It checks **both shelves**.
2. It looks at **available space**, **resources (CPU/memory)**, and a few other conditions.
3. If **Shelf B is empty** and has **more free space**, Kubernetes will **usually choose the empty shelf**.

Why?

Because it tries to:

- **Balance the load** across shelves.
- Use resources **efficiently**.
- Avoid putting too much on one shelf while the other one is empty.

✅ So, in simple words:

> If you have two computers (nodes), and one already has apps (pods) running while the other is doing nothing, Kubernetes will typically put the next app (pod) on the **empty one** — to keep things balanced.

🛠️ Caveat:

This assumes both nodes are **healthy**, **ready**, and **have enough resources**. If the empty node is not working or doesn't meet the pod's needs, Kubernetes **won't use it**, even if it's empty.

2. If an application running in a container encounters an OOM (Out-of-Memory) error, will the container restart, or will the entire Pod be recreated?

🧠 Situation:

- You're running an **application** inside a **container**, and it's part of a **Pod** in Kubernetes.
- The app uses **more memory than allowed**, and hits an **OOM (Out-of-Memory) error**.

🚨 What happens next?

🔄 Will the container restart?

✅ **Yes** — Kubernetes will **restart the container** inside the pod.

🏗️ Will the whole Pod be recreated?

❌ **No**, the **Pod is not recreated** (unless there's some other issue).

🔧 Why is that?

Think of a Pod like a **house**, and a container like an **appliance inside it (like a microwave)**.

- If the **microwave blows a fuse** (OOM error), you **replace or restart the microwave**, but you don't **rebuild the entire house**.
- Similarly, Kubernetes will **restart the container** that failed, **not the whole pod**, unless something more serious happens.

✅ In short:

> If a container hits an OOM error, **only that container restarts, not the whole pod**. The pod remains the same — Kubernetes just gives the container another shot.

3. Can application configurations such as environment variables or ConfigMap updates be applied dynamically without recreating the Pod?

🔴 For Environment Variables:

❌ **No**, you **cannot change environment variables dynamically** for a running pod.

- When you define environment variables in a pod spec, they get **injected only at pod startup**.
- To apply changes, you need to **recreate the Pod** (either manually or through a deployment rollout).

➡️ Think of it like baking a cake with some ingredients (env vars). Once baked, you can't change the sugar level — you need to bake a new cake.

🟡 For ConfigMaps:

It depends on **how your application consumes them:**

✅ Mounted as Volumes (Recommended Way):

- If your ConfigMap is mounted as a **volume**, and you **update the ConfigMap**, the **file contents in the Pod can update** automatically **(usually within a few seconds)**.
- BUT: the **application must re-read the file** to reflect the changes.

✅ **Pod restart not needed, if your app watches the file.**

❌ Used as Environment Variables:

- If you use ConfigMap values as **env vars**, they behave **just like normal env vars.**
- So: ❌ You **must recreate the Pod** to see the changes.

4. Is a Pod stable once created, even if the user takes no further action?

🟡 The honest answer:

"It depends."
A Pod is **not guaranteed to be permanently stable** just because it's created. Here's why:

✅ When is a Pod stable?

A Pod is *stable* **only as long as:**

- The **node it's running on stays healthy**

- The **container inside doesn't crash**
- There are **no external forces** (like eviction or scaling events)
- It's not affected by **resource pressure** (memory, CPU, disk)

🧠 TL;DR:

> A pod **can run stably** for a while if nothing goes wrong, but it is **not inherently stable or self-healing.**
> Use controllers (like Deployments) to ensure **long-term stability and auto-recovery.**

5. Can a Service of type ClusterIP ensure load balancing for TCP traffic?

🔷 What is a ClusterIP Service?

- It's the **default type** of Service in Kubernetes.
- It exposes a group of pods **internally** (within the cluster).
- Other services or pods inside the cluster can **access it via a stable virtual IP.**

🔄 Does it Load Balance?

✅ **Yes**, it **automatically load balances** traffic across all healthy pod endpoints **behind the service.**

This applies to:

- ✅ TCP traffic

- ✅ **UDP traffic**
- ✅ **HTTP/HTTPS traffic** (since they're built on TCP)

⚙️ How it works (TCP context):

1. You create a `Service` with `ClusterIP` type and point it to a `selector`.
2. Kubernetes keeps track of all the **pods matching the selector**.
3. When a pod inside the cluster sends a **TCP request to the service IP**, Kubernetes (via **kube-proxy**) uses **round-robin** or similar strategy to **load balance** that TCP connection to one of the backend pods.

🧠 TL;DR:

> Yes, a `ClusterIP` service **does load balancing** for **TCP traffic**, across all matching pods inside the cluster. It's automatic, internal, and built-in.

6. How should application logs be collected, and is there a risk of losing logs?

📤 1. Standard Practice: Write logs to stdout/stderr

- Your app should **not write to files inside the container** (like `/var/log/app.log`).
- Instead, it should log to **standard output (stdout)** and **standard error (stderr)**.
- This is because Kubernetes captures these logs **automatically** via the container runtime.

🧠 TL;DR:

> ✔️ **Log to stdout/stderr,**
> ❌ **Don't rely on local logs inside the container,**
> 🛡️ **Use centralized log collection tools** (like Fluent Bit, Loki, etc.) to ensure logs are saved and searchable even if pods crash or nodes go down.

7. If an HTTP Server Pod's `livenessProbe` is functioning correctly, does it mean the application is problem-free?

🔍 **No**, a passing `livenessProbe` does **not necessarily mean** the application is **problem-free**.

Let's explain that clearly:

✅ What a `livenessProbe` actually checks:

- It **only checks if the container is alive** — i.e., it's not stuck, crashed, or frozen.
- If the liveness check passes, **Kubernetes assumes the app is still running** — but it doesn't mean everything is working perfectly.

📍 Real-world example:

Imagine your app has an HTTP endpoint `/healthz` used by `livenessProbe`.

- The endpoint simply returns 200 OK if the server is running.

- BUT… your app's real business logic (like DB queries or user APIs) might be:
- Returning 500 errors
- Failing to connect to the database
- Timing out due to internal issues

> Kubernetes **won't know any of that** unless your liveness probe is smart enough to check those too — which it usually isn't (and shouldn't).

✅ Best Practice:

- Use **livenessProbe** to check:
  👉 *Is the process running and not stuck?*
- Use **readinessProbe** to check:
  👉 *Can the app serve traffic right now?*
- Use **external monitoring** or application-level health checks to track **real issues**.

🧠 TL;DR:

> ✅ A working `livenessProbe` just means the container is **alive**.
> ❌ It does **not** mean the application is problem-free, healthy, or ready to serve real user requests.

8. How can an application scale to handle traffic fluctuations?

✅ The answer: Auto-scaling mechanisms

In Kubernetes, the most common and effective way to scale applications is by using:

🔷 1. Horizontal Pod Autoscaler (HPA)

> Automatically **adds or removes pods** based on resource usage like **CPU**, **memory**, or **custom metrics**.

🛠️ How it works:

- If traffic increases ➡️ CPU goes up ➡️ HPA adds more pods.
- If traffic decreases ➡️ CPU drops ➡️ HPA reduces pods.

🔷 2. Cluster Autoscaler

> Dynamically **adds/removes nodes** in the cluster based on pod demands.

- If there aren't enough nodes to run new pods, Kubernetes will request the cloud provider to **add nodes**.
- When nodes are underused, they can be **terminated** to save cost.

🧠 Used together with HPA = full-stack scaling.

🔷 3. Vertical Pod Autoscaler (VPA)

> Adjusts **CPU and memory requests/limits** of pods.

- Less common in high-fluctuation traffic cases.
- More useful for optimizing stable workloads (e.g., batch jobs, cron jobs).

🔷 4. Custom Metrics Autoscaler

> You can also scale based on **application-specific metrics** like:

- Request rate (RPS)
- Queue length
- Latency
- Business metrics (e.g., number of active users)

This uses the **Kubernetes Metrics Server** or **Prometheus Adapter**.

9. When you execute `kubectl exec -it <pod> -- bash`, are you logging into the pod?

✅ **Yes**, when you run:

```
bash
```

```
CopyEdit
```

```
kubectl exec -it <pod> -- bash
```

You are essentially **"logging into" the pod's container**, but let's explain **what that really means.**

✅ So are you logging in?

✔️ **Yes — in practical terms:**
You're getting a **shell inside the container** running in the pod — so it **feels like you're logged in**, similar to SSH.

❌ **But technically:**
You're **not logging into the pod itself**, you're running a command (`bash`) **inside one container of the pod.** You're not touching the pod object; you're entering **one of its containers.**

10. How would you troubleshoot if a container in a Pod repeatedly exits and restarts?

🧠 TL;DR Troubleshooting Flow:

1. 🔍 `kubectl get pods` → Check status and restarts
2. 📝 `kubectl describe pod` → Look for exit codes, events
3. 📜 `kubectl logs --previous` → Analyze crash logs
4. ⚙️ Check config, env, entrypoint, resources
5. 🧪 Use `kubectl exec` if possible
6. 🛠️ Fix, redeploy, and monitor

🏁 Conclusion

So, how did you do?

If you aced the quiz — congratulations! You've likely spent some serious time in the trenches with Kubernetes. If you stumbled on a few questions, that's okay too — Kubernetes is vast, and there's always more to learn.

Whether you're preparing for a certification, a job interview, or just leveling up your cloud-native skills, the key is **continuous learning and hands-on practice.**

✅ Stay curious.
✅ Keep exploring.
✅ And remember: even the best engineers are always learning.

If you found this quiz helpful, share it with your network and challenge your colleagues too — let's see who the real Kubernetes experts are! 🚀

DevOps Kubernetes AWS Azure Terraform

Written by Dheeman Das

255 followers · 6 following

Follow

🚀 DevOps Engineer | Passionate about CI/CD, Kubernetes, Azure, AWS & the intersection of DevOps with AI/ML | Sharing cloud-native insights & automation tips

Responses (2)

Rox

What are your thoughts?

Sergey
13 hours ago

It's not the questions for expert.

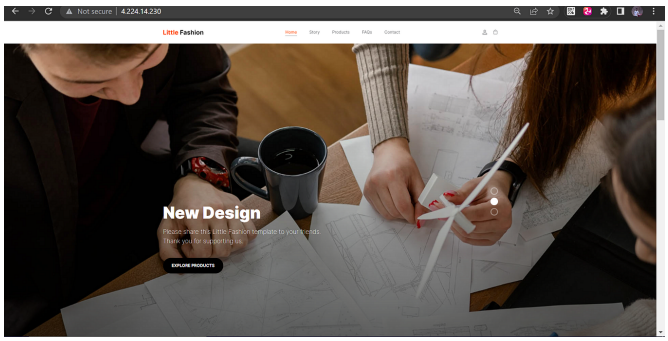
3 Reply

Steve Atkinson
2 days ago

What a great set of questions! I think the correct answer to 9 should be no... you are not execing to the pod - this is covered in the explanation but if I asked that question of a candidate I would expect them to say no. I'm giving myself 9.5 out of 10 because of that!

1 1 reply Reply

More from Dheeman Das



Dheeman Das

Deploy Simple Static Web Application on Azure Kubernet...

Today, we'll look at Azure Kubernetes Service (AKS)—a revolutionary tool for...

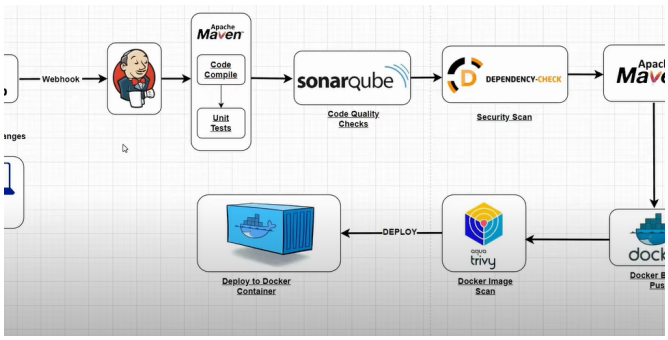
Feb 5, 2024 68

Dheeman Das

Essential Linux Commands Every DevOps Engineer Should Master

Table of Contents

May 22 2



Dheeman Das

🚀 Ensuring High Availability in Kubernetes with Pod Resources...

As containerized applications scale, reliability and resource optimization...

Jul 3 1

Dheeman Das

End To End DevOps CI/CD Project 🔥🔥

Step 1 :- Launch an ec2 instance (t2.medium, ubuntu) and login to it and...

Jul 10, 2023 157 4

See all from Dheeman Das

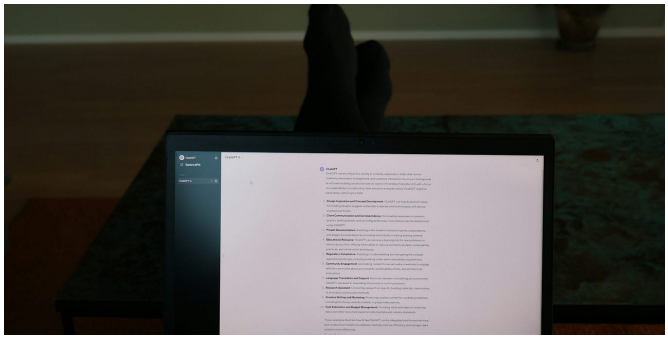
Recommended from Medium

In AWS in Plain En... by Sandesh I DevOps | AW...

How a "Minor" Kubernetes Config Change Took Down Our Entire...

- By Sandesh (5 + Years of DevOps | CI/CD | AWS | k8 | DevSecOps)

Jul 6 38 1

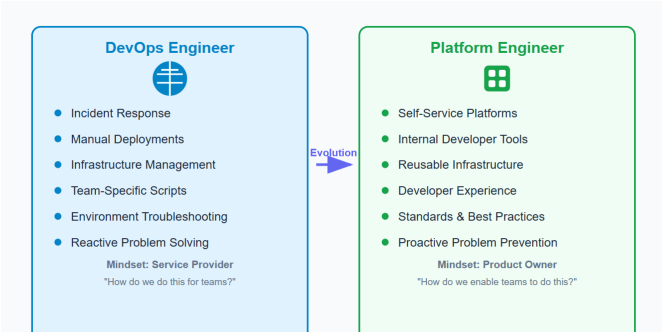


In FAUN—Developer Communit... by Quân Huỳ...

I Generated Production-Ready Kubernetes Configs in 30...

The 5-letter framework that turned my AI from a glorified search engine into a senio...

Jul 3 71



Sridhar

When Your Company Transforms DevOps into Platform...

What happens when your organization decides to rebuild how engineering teams...

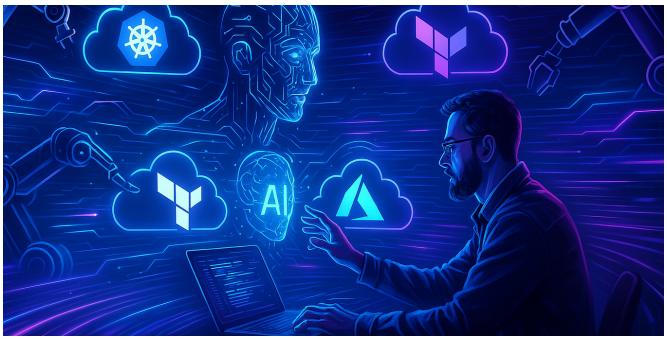
Jul 4 4 1

In CareerByteCode by Saraswathi Lakshman

DevOps Scenario-Based Interview Guide

Introduction

May 15 5



Zudonu Osomudeya

How AI Is Revolutionizing DevOps in 2025

Jun 16 354 6

Devops Diaries

DevOps Interview Questions

Are you preparing for a DevOps interview and feeling overwhelmed by the vast topic...

5d ago 7 1

See more recommendations