**Monitoring Guide**

Talentlink

Technical monitoring commands

Alizada,Erfan E.

[Year]

**Check for all namespaces:  
  
A screenshot of a computer

AI-generated content may be incorrect.**

* **default** → where things go if you don’t specify a namespace. Often empty in real deployments.
* **cert-manager** → runs the certificate manager (responsible for TLS certs, renewals).
* **ingress-nginx** → your ingress controller (handles routing incoming traffic).
* **kube-system** → core Kubernetes components (scheduler, kube-dns, controller-manager, etc.).
* **kube-node-lease** → used internally by Kubernetes for node heartbeats.
* **kube-public** → mostly empty, but readable by anyone (even unauthenticated users).

**View events in all namespaces**



if you see **“No resources found”**, it means your cluster currently has no recorded events (or the event history has expired). In our case there is currently no event to record.

If you want to focus on something specific, like cert-manager (for TLS) or ingress-nginx (for routing), you can filter:

A black background with white text

AI-generated content may be incorrect.

**Events are short-lived**

By default, Kubernetes events are **ephemeral**:

* They are stored in etcd but usually only kept for **1 hour** (sometimes 24h depending on cluster config).
* If nothing has happened recently, kubectl get events can show nothing.

**Force some events to appear for testing**

You can easily trigger fresh events:  
This should produce scheduling events like Scheduled pod, Pulling image, Started container, etc.

A computer screen shot of a program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

**Live event watching (good for debugging)**

Instead of listing once, you can stream events in real time:



A screen shot of a computer

AI-generated content may be incorrect.

You can have a more readable version with stern (if you install it first):

A computer screen shot of a code

AI-generated content may be incorrect.

Use stern for **streaming pod logs** (colorized, multiple pods at once). Example:



A screen shot of a computer code

AI-generated content may be incorrect.

**Key parts of each section**

* **Pod name** → nginx-ingress-ingress-nginx-controller-958495fb5-2bctk (the ingress controller handling the request).
* **Timestamp** → [17/Sep/2025:11:41:08 +0000].
* **Client IP** → 10.0.10.69 (the source of the request, maybe your load balancer node).
* **Request** → GET /flutter\_bootstrap.js HTTP/2.0.
* **HTTP status** → 200 (OK).
* **Bytes sent** → 9590 (size of response).
* **Referrer** → "https://talentlink-erfan.nl/".
* **User-Agent** → "Mozilla/5.0 ... HeadlessChrome/125..." (the client/browser).
* **Upstream service** → [default-frontend-service-80] (the backend Kubernetes Service it routed to).
* **Upstream pod IP** → 10.0.10.141:80 (the pod that served the request).
* **Response time** → 0.001 seconds.
* **Final status** → 200 (again confirmed at the end).

**What does this explain:**

* Requests are successfully flowing **through my Ingress** (ingress-nginx) to my **frontend service** (default-frontend-service).
* Static assets (main.dart.js, fonts, icons, service worker, etc.) are being served with **HTTP 200**, so the site loads fine.
* Response times are **fast** (0.000–0.087 sec) — looks healthy.
* Some 308 entries appear — that’s an **NGINX redirect** (likely HTTP → HTTPS or / → /index.html). That’s normal unless it loops.

Other Stern commands for monitoring that can be used.

A screenshot of a computer program

AI-generated content may be incorrect.

**Checking the Certificate status**

**A black background with white text

AI-generated content may be incorrect.**

**Checking orders**

**A black background with white text

AI-generated content may be incorrect.**

**Learning Outcome relation:**

**Learning Outcome 3: Applying Enterprise Technologies (Strongest)**

* **Why:** The document shows how you use Kubernetes monitoring tools (kubectl get events, stern, live logs) and cert-manager to ensure your system is running as expected.
* Monitoring is a **non-functional requirement** (observability, reliability), which is a key part of enterprise architecture.
* By including namespace overviews, event lifecycles, and certificate checks, you demonstrate applied knowledge of **enterprise-level monitoring practices**.

**Learning Outcome 4: Analyzing and Implementing Security (Strong)**

* **Why:** TLS certificate monitoring (cert-manager, certificate requests, orders) ensures secure communication.
* The document covers how certificates are issued, renewed, and debugged, which directly contributes to **system security and compliance**.

**Learning Outcome 2: Designing and Implementing Reliable Systems (Moderate)**

* **Why:** Monitoring logs and events helps ensure reliability by detecting failures early (image pull issues, pod crashes, ingress errors).
* The guide shows how you set up observability so your application **stays healthy under production conditions**.

**Learning Outcome 1: Applying Software Development Practices (Weakest, but still relevant)**

* **Why:** While the guide is more operations-focused, it indirectly supports development by showing how logs and events can be used to debug code issues.
* Developers benefit from this monitoring setup when diagnosing application-level errors.