My Kubernetes Troubleshooting Checklist

What Are Runtime Errors in Kubernetes?

When an application that is containerized crashes after launching successfully, runtime problems arise. Misconfigurations, memory leaks, missing dependencies, and poor programming can all cause these issues.

Common Causes:

- Application crashes due to unhandled exceptions (e.g., bad database queries).
- Missing dependencies in the container image
- Insufficient permissions (trying to write files in read-only locations)
- Memory exhaustion causing Out-Of-Memory (OOMKilled)
- Misconfigured startup settings
- Network/plugin issues
- OS-level security settings

How to overcome :

- Check application logs using `docker logs <container_id>` or `kubectl logs <pod_name>`
- Ensure all required dependencies are installed inside the container
- Debug memory issues with `docker stats <container_id>` or `kubectl top pod`

Common Kubernetes Runtime Errors (Explained in Detail):

✓ <u>CrashLoopBackOff</u>

Meaning:

The container starts → crashes → Kubernetes restarts it → it crashes again → Kubernetes restarts it and so on… K8s backs off from trying for a while.

$$\begin{tikzcd}
\text{Container starts} \arrow[r, dash] \arrow[d, dash] & \text{crashes} \arrow[d, dash] \\
\text{it crashes again} \arrow[r, dash] & \text{Kubernetes restarts it}
\end{tikzcd}$$

Common Causes:

- Application exits immediately (e.g., wrong env var)
- Misconfigured command/entrypoint
- Missing file/config
- Failing liveness/readiness probe

<u>How to Fix:</u>

Check logs:

```
kubectl logs <pod_name>
```

Inspect pod details:

```
kubectl describe pod <pod_name>
```

Verify configs:

`kubectl get configmap` and `kubectl get secrets`

Inspect events:

```
kubectl get events --sort-by=.metadata.creationTimestamp
```

Note : we can check for Exit code, Stack trace, Probe failures

Application errors: The app inside the container might be misconfigured or broken.

To Fix :

Check logs using:

```
kubectl logs <pod-name> --previous
```

Missing dependencies: The container may require files, services, or configurations that are unavailable.

To Fix:

Ensure dependencies are correctly installed and accessible.

Insufficient resources: The pod may not have enough CPU/memory to run.

To Fix :

Adjust resource requests/limits using:

```
yaml
resources:
  requests:
    memory: "512Mi"
    cpu: "250m"
  limits:
    memory: "1Gi"
    cpu: "500m"
```

- ✓ <u>**RunContainerError**</u>

Meaning:
Kubernetes failed to start the container after pulling the image. The container didn't even get to start properly due to bad images or permission issues.

Common Causes:

- Invalid command or script inside Dockerfile
- Wrong ENTRYPOINT or CMD
- Volume mounts pointing to non-existent paths
- User permissions problem (like trying to run as root when it’s not allowed)
- Invalid container image or **missing image** in registry

<u>How to correct:</u>

Verify the image exists

```
kubectl get pods -o wide
```

Check entry points using

```
docker inspect <image>
```

Try running the container manually to test

```
docker run <image>
```

<u>Corrupt container image</u>: The image may be broken or missing.

To Fix:

Try pulling the image manually

```
docker pull <image-name>:<tag>
```

<u>Incorrect entrypoint command</u>: The container startup command might be invalid.

To Fix:

Check entrypoint script inside the Dockerfile

dockerfile

```
ENTRYPOINT ["python", "app.py"]
```

Permissions issue: The container may lack proper access to resources.

To Fix:

Review security policies and fix RBAC permissions.

- ✓ **KillContainerError**

Meaning:
Kubernetes attempted to stop a running container and failed — possibly during pod shutdown or eviction.

Common Causes:

- ➢ Application didn't respond to SIGTERM
- ➢ Stuck in cleanup or shutdown hooks
- ➢ Kubelet or Docker runtime timeout
- ➢ Container exceeding **memory limits** (spec.containers[].resources.limits)
- ➢ Memory leaks in the application
- ➢ Unoptimized resource usage

Ways to Correct:

- ➢ Review logs for shutdown routines
- ➢ Ensure app handles graceful termination
- ➢ Set appropriate terminationGracePeriodSeconds in pod spec
- ➢ Check pod details `kubectl describe pod <pod_name>`
- ➢ Monitor resource usage `kubectl top pod`
- ➢ Adjust memory limits in YAML (resources.requests.memory and resources.limits.memory)

- ✓ **<u>VerifyNonRootError</u>**

Meaning:
Container tried to run as root when the pod security settings **require non-root**.

Common Causes:

Pod securityContext.runAsNonRoot: true is set

Image uses USER root

No user defined in the Dockerfile

<u>How to Adjust:</u>

<u>Pod security policies require non-root execution</u>.
To Fix :
Check the security policy using:

```
yaml
securityContext:
  runAsNonRoot: true
  runAsUser: 1000
```

<u>The container image is built to run as root</u>.
To Fix :
Modify the Dockerfile to specify a non-root user:

```
dockerfile
USER 1000
```

- ✓ **RunInitContainerError**

Meaning:
An init container (runs before your main container) failed.

Common Causes:

- Wrong command/script in init container
- Network or volume dependency issue
- Pull image failure (if different registry)
- Missing required dependencies (e.g., database not ready)
- Timeout in init script execution
- Incorrect commands in init container

<u>Possible Fixes:</u>

<u>Check Script inside the init container failed.</u>

To Fix Debug the logs using:

```
kubectl logs <pod-name> -c <init-container-name>
```

<u>Missing dependencies in the init container.</u>

To Fix Ensure that all required configurations or tools are installed.

Check init container logs

```
kubectl logs <pod_name> -c <init-container-name>
```

- Ensure dependencies exist before starting the pod

<u>Verify entry points in YAML (initContainers.command)</u>

- ✓ **CreatePodSandboxError**

Meaning:
Kubelet failed to create the pod's **sandbox** (a runtime environment), before even starting containers.

Common Causes and Fixes:

- CNI plugin misconfiguration
- Docker runtime failure
- Node-level issue (out of resources or disk)
- Container runtime failure**:** The node may be experiencing Docker or containerd failures.

  To Fix

  Restart the runtime service:

  ```
  sudo systemctl restart docker
  sudo systemctl restart containerd
  ```
- Insufficient node resources: The Kubernetes node might be running out of CPU/memory.

  To Fix Use

  ```
  kubectl top node
  ```