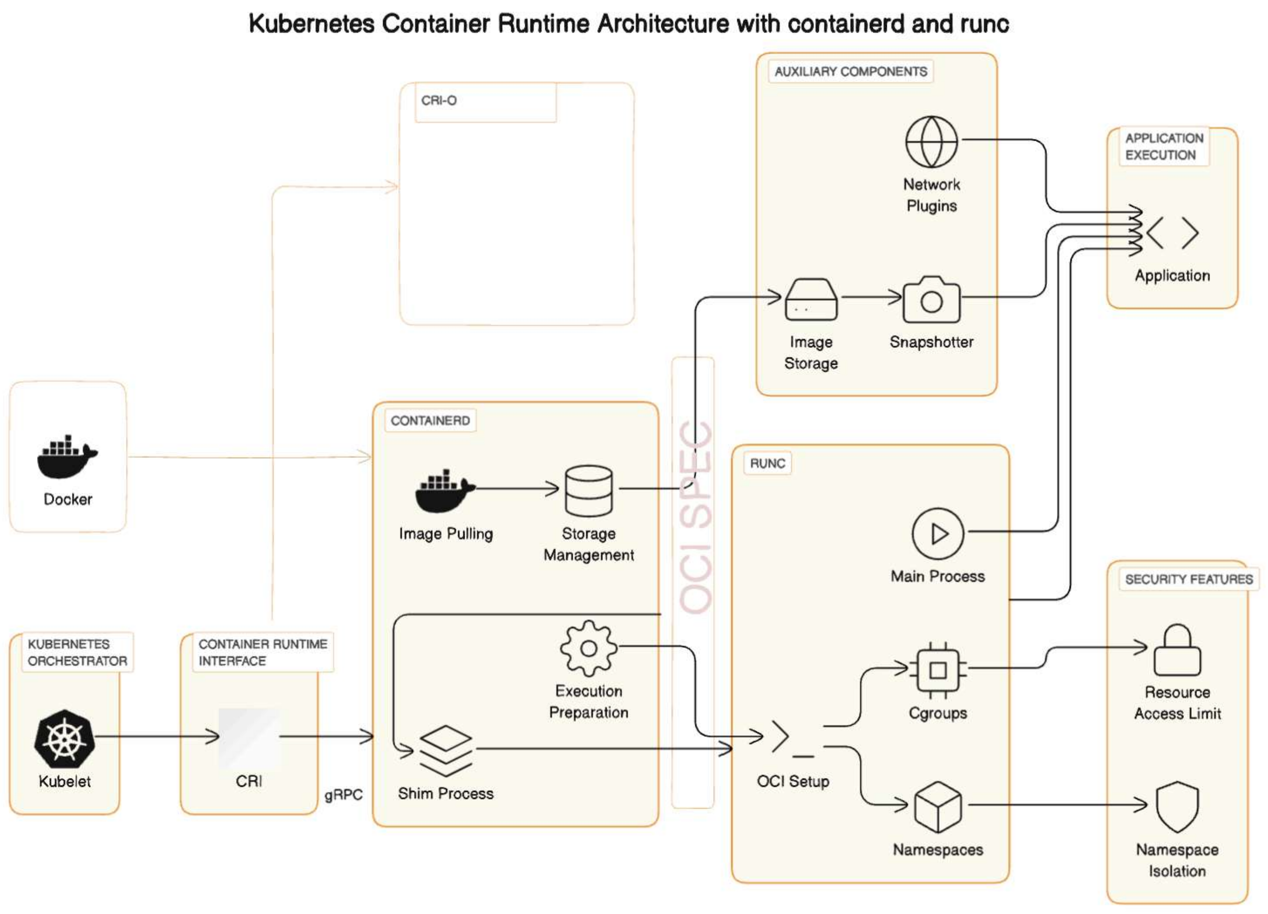
🧠 What is a Container Runtime?

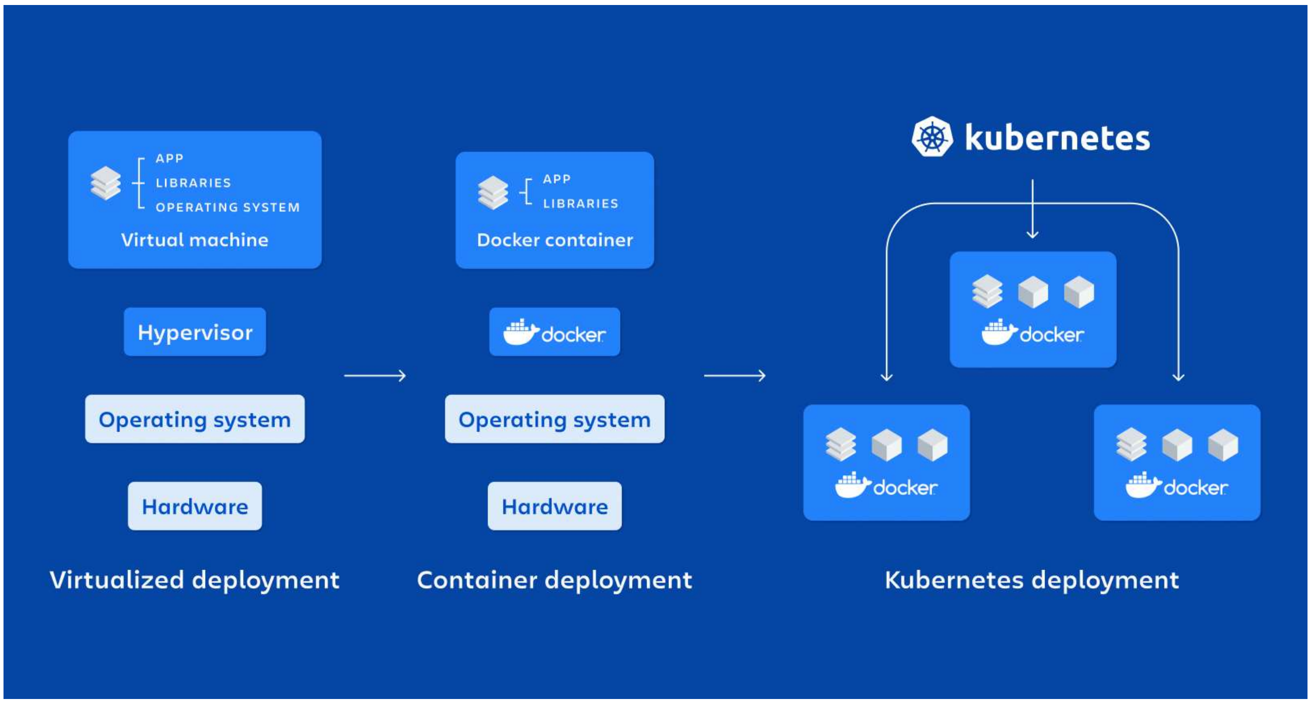
A **container runtime** is the component responsible for:

- Pulling container images
- Creating containers
- Running containers
- Stopping & deleting containers

👉 Kubernetes **does NOT run containers directly** 👉 It talks to the runtime using **CRI (Container Runtime Interface)**

🏗️ Container Runtime Architecture





```
kubectl
   |
kube-apiserver
   |
kubelet
   | (CRI)
┌──────────────┐
│ containerd   │  ← Runtime
└──────────────┘
       |
       |
     runc
       |
   Linux Kernel
```

🔁 Container Runtime Landscape

Runtime	Type	CRI	Production
Docker	High-level	❌ (removed)	❌
containerd	Low-level	✅	⭐⭐⭐⭐⭐
CRI-O	Low-level	✅	⭐⭐⭐⭐
runc	OCI runtime	❌	internal

Runtime	Type	CRI	Production
Kata Containers	VM-based	✅	niche

🐳 1. Docker (❌ Deprecated in Kubernetes)

❗ Important Truth

- Docker **was removed from Kubernetes v1.24+**
- Kubernetes **never used Docker directly**
- It used **dockershim** (now removed)

```
Kubernetes ❌ Docker
Kubernetes ✅ containerd / CRI-O
```

✔ Docker still useful for **local development**

📦 2. containerd (🔥 DEFAULT & RECOMMENDED)

Why containerd?

- CNCF project
- Lightweight
- High performance
- Used by **EKS, GKE, AKS, kubeadm**

🔧 Install containerd (Ubuntu – Production Way)

Step 1: Install

```
sudo apt update
sudo apt install -y containerd
```

Step 2: Generate config

```
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml
```

Step 3: Enable Systemd Cgroup (MANDATORY)

Edit:

```
sudo vi /etc/containerd/config.toml
```

Set:

```
SystemdCgroup = true
```

Step 4: Restart

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

🔍 Verify containerd

```
crictl info
crictl ps
```

📜 Kubernetes Uses containerd Like This

```
apiVersion: v1
kind: Pod
metadata:
  name: runtime-test
spec:
```

```yaml
  containers:
  - name: nginx
    image: nginx
```

⬆️ Kubernetes pulls & runs image via **containerd + runc**

🔴 3. CRI-O (RedHat / OpenShift)

Why CRI-O?

- Built **only for Kubernetes**
- No extra features
- Secure by default

Used by:

- OpenShift
- Some hardened clusters

🔧 Install CRI-O (Ubuntu Example)

```bash
sudo apt install -y cri-o cri-o-runc
sudo systemctl enable crio
sudo systemctl start crio
```

Verify:

```bash
crictl info
```

📜 Pod YAML (Same as containerd)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: crio-test
```

```
spec:
  containers:
  - name: busybox
    image: busybox
    command: ["sleep","3600"]
```

⚙️ 4. runc (Low-Level Runtime)

What is runc?

- Executes containers
- Implements **OCI spec**
- Used internally by:
  - containerd
  - CRI-O
  - Docker

You **never configure runc directly** in Kubernetes.

🧠 Runtime Relationship (VERY IMPORTANT)

```
Docker
 └─ containerd
     └─ runc

Kubernetes
 └─ containerd / CRI-O
     └─ runc
```

🧊 5. Kata Containers (Extra Isolation)

What makes Kata special?

- Each container runs in a **lightweight VM**
- Strong isolation
- Slight performance overhead

Use cases:

- Multi-tenant clusters
- Untrusted workloads

📜 RuntimeClass Example (Kata)

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: kata
handler: kata-runtime
```

Pod using Kata:

```
spec:
  runtimeClassName: kata
```

🔍 How to Check Runtime in Your Cluster

```
kubectl get nodes -o wide
```

Look for:

```
CONTAINER-RUNTIME
containerd://1.7.x
```

Or:

```
kubectl describe node | grep -i runtime
```

⚠️ Common Runtime Issues (On-Call SRE)

Issue	Cause
Pods stuck in ContainerCreating	Runtime down
CrashLoopBackOff	Image / runtime error
kubelet not starting	Cgroup mismatch
ImagePullBackOff	containerd registry issue

Fix:

```
sudo systemctl restart containerd
journalctl -u containerd
```

🧭 Runtime Selection Guide

Environment	Runtime
kubeadm	containerd
EKS	containerd
GKE	containerd
OpenShift	CRI-O
Multi-tenant	Kata
Local Dev	Docker

🧠 What is containerd?

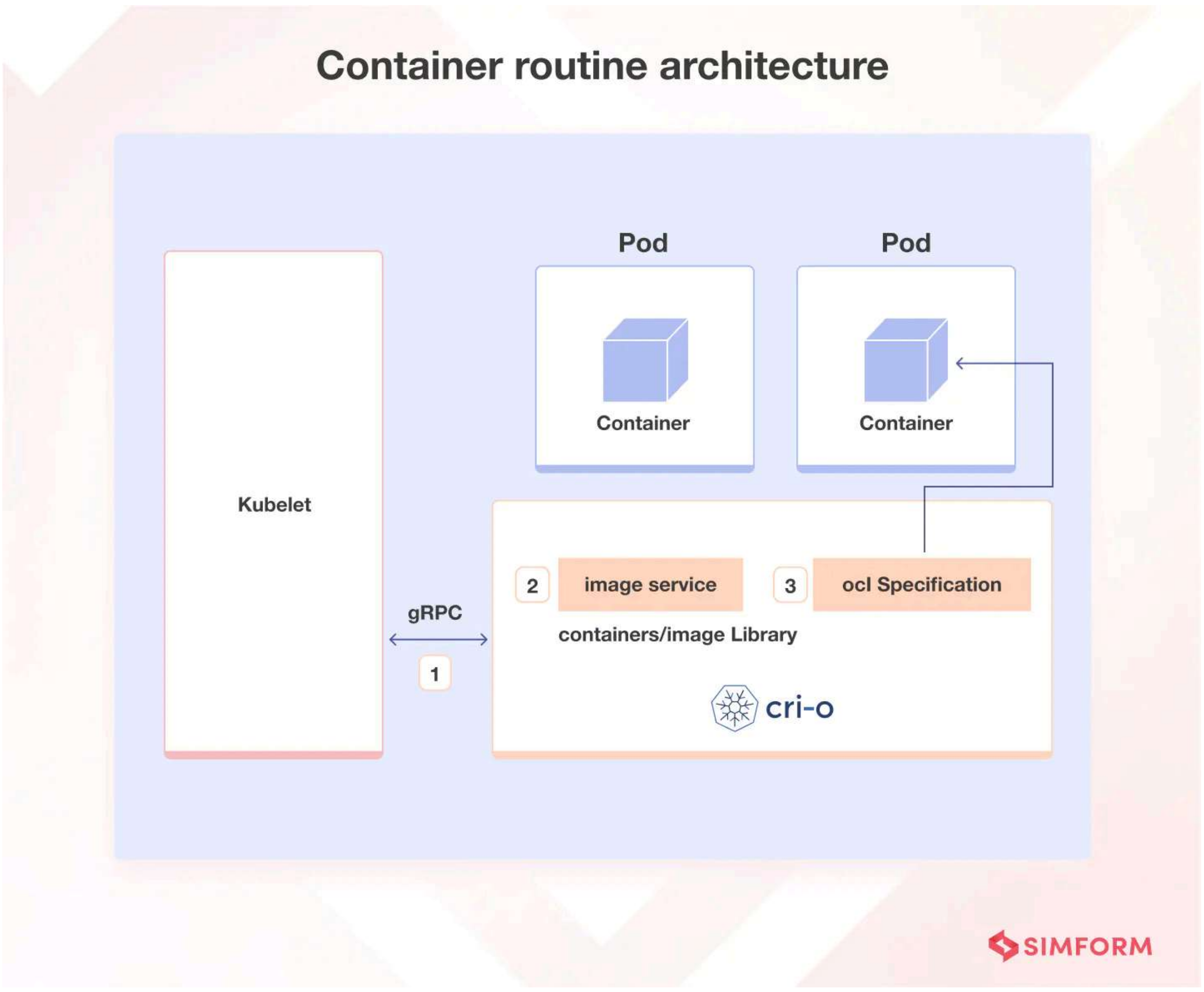
`containerd` is a **low-level container runtime** responsible for:

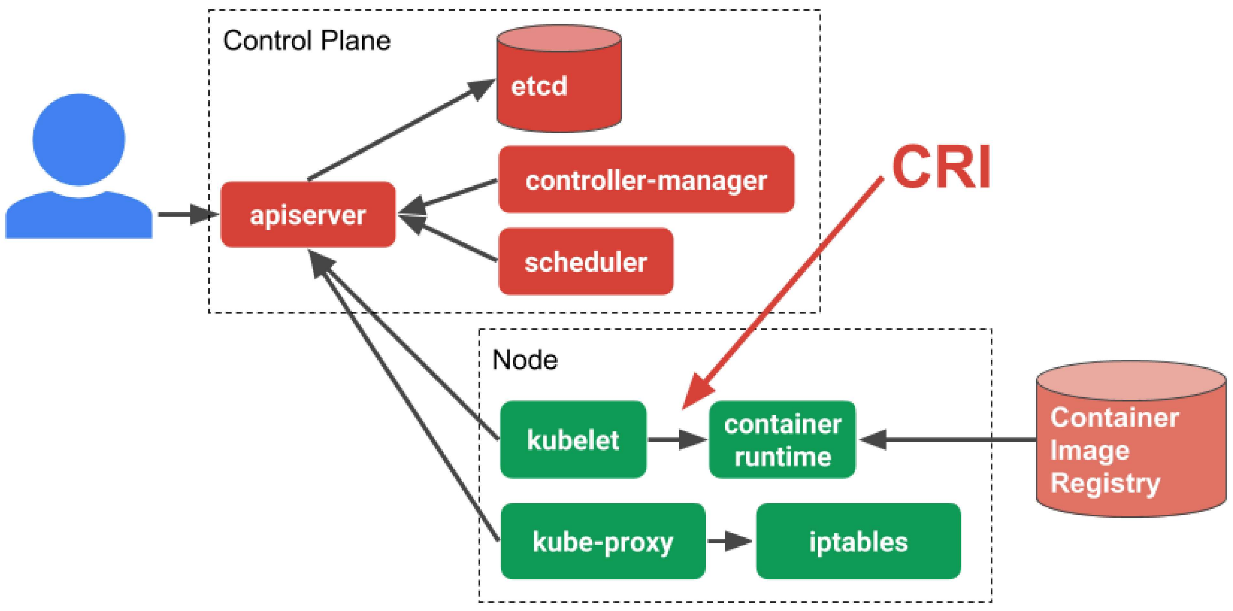
- Pulling container images

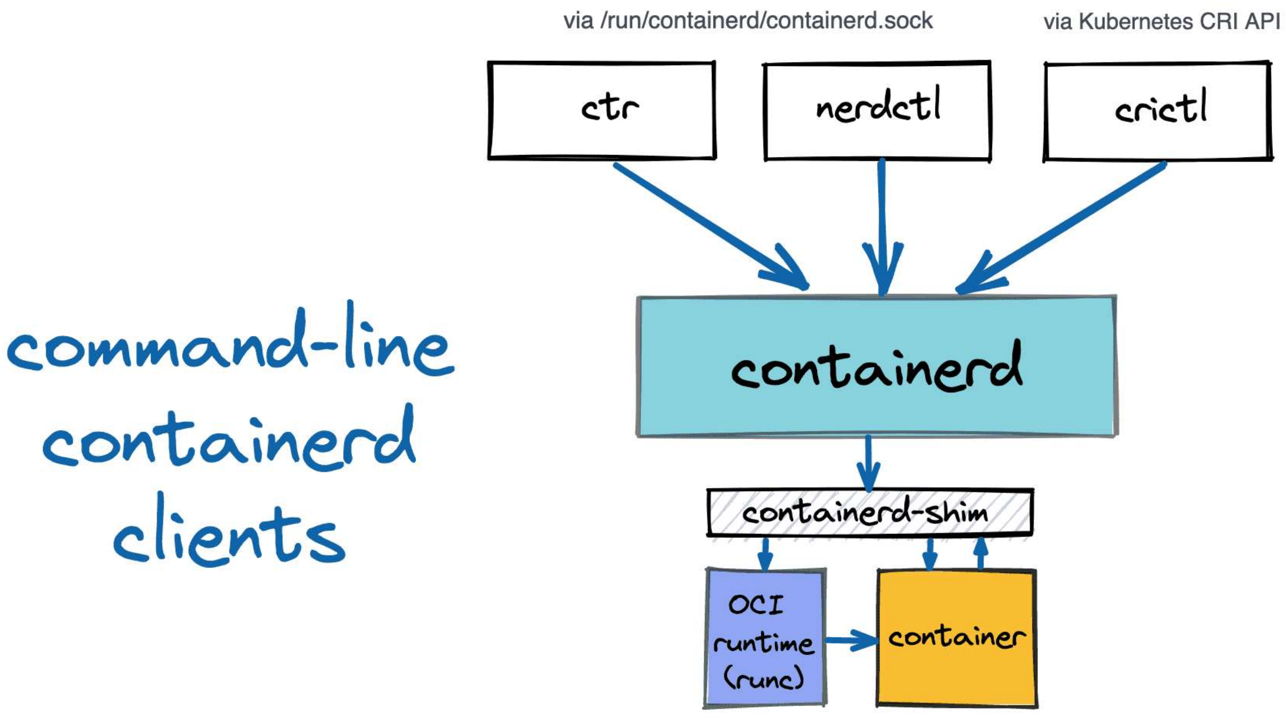
- Creating containers
- Running containers
- Managing container lifecycle

👉 Kubernetes talks to containerd using **CRI (Container Runtime Interface)** 👉 containerd internally uses **runc** to start containers

🏗 Container Runtime Architecture







```
kubectl
   │
kube-apiserver
   │
kubelet
   │  (CRI)
┌────────────┐
│ containerd │
├────────────┤
│ ctr / nerdctl│  ← CLI tools
└──────┬─────┘
```

```
         |
       runc
         |
    Linux Kernel
```

📦 Install containerd (Ubuntu)

Step 1: Install containerd package

```
sudo apt install -y containerd.io
```

Step 2: Create containerd config directory

```
sudo mkdir -p /etc/containerd
```

Step 3: Generate default configuration

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

This creates the **official default runtime configuration**.

Step 4: Restart & enable containerd

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Step 5: Verify service

```
systemctl status containerd
```

📥 Pull Image Using `ctr`

```
ctr image pull docker.io/library/alpine:latest
```

🔎 Notes

- `ctr` is **containerd's native CLI**
- Mostly for **debugging**
- Not user-friendly
- **Not recommended** for daily use

🧪 Run Container Using `nerdctl`

```
nerdctl run -it --rm docker.io/library/alpine:latest
```

Why nerdctl?

- Docker-like syntax
- Works directly with containerd
- Supports:
    - volumes
    - networks
    - compose

Tool	Ease	Use Case
ctr	❌ Hard	Debugging
nerdctl	✅ Easy	Daily usage
docker	✅ Easy	Dev only

🔍 CRI Tool – `crictl`

```
crictl
```

What is `crictl`?

- CLI for **CRI-compatible runtimes**
- Used by Kubernetes admins
- Talks directly to containerd / CRI-O

Common commands:

```
crictl info
crictl images
crictl ps
crictl pods
```

🧠 Tool Comparison (Very Important)

Tool	Talks To	Used By
ctr	containerd	Runtime debugging
nerdctl	containerd	Humans
crictl	CRI	Kubernetes admins
kubectl	kube-apiserver	Users

⚠️ Production Best Practices

✔ Use **containerd**, not Docker ✔ Use **nerdctl** for local testing ✔ Use **crictl** for Kubernetes debugging ✔ Never use `ctr` in automation ✔ Always enable **systemd cgroups** in `/etc/containerd/config.toml`

🔧 Mandatory Production Setting (IMPORTANT)

Edit:

```
sudo vi /etc/containerd/config.toml
```

Set:

```
SystemdCgroup = true
```

Restart:

```
sudo systemctl restart containerd
```

Without this → **kubelet will fail**

🧭 How Kubernetes Uses containerd

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: runtime-test
spec:
  containers:
  - name: alpine
    image: alpine
    command: ["sleep","3600"]
```

Flow:

```
kubectl → kubelet → CRI → containerd → runc → kernel
```

🚨 Common On-Call Issues

Issue	Cause
ImagePullBackOff	Registry / auth issue
ContainerCreating	Runtime down
kubelet crash	Cgroup mismatch
Pods stuck	containerd not running

Fix:

```
sudo systemctl restart containerd
journalctl -u containerd
```

🎯 Key Takeaways (Interview Gold)

- Kubernetes **does NOT use Docker**
- containerd is **production standard**
- runc actually **creates containers**
- CRI is the **runtime interface**
- nerdctl = Docker for containerd