crictl CRI container debugging

START HERE

`crictl pods`
List pod sandboxes

`crictl ps -a`
List all containers

`crictl logs <container>`
Show container logs

`crictl exec -it <container> sh`
Shell into container

`crictl inspect <container>`
Container details

`crictl images`
List images

`crictl stats`
CPU and memory use

`crictl info`
Runtime status

crictl CRI client → gRPC socket → **Runtime** containerd / CRI-O → runs → **Pod sandbox** shared network → holds → **Containers** app + sidecars

CONCEPTS

crictl talks to the node's CRI runtime directly, so it sees kubelet-managed pods that docker or nerdctl do not list. Run it on the node, usually as root.

Without an endpoint crictl probes several default sockets and is slow; set one in /etc/crictl.yaml.

Removing an image by tag deletes the image and all its tags. Use nerdctl, ctr or podman to untag one.

CONNECT

`crictl --runtime-endpoint <socket> ps`
Use a specific runtime socket

`crictl config --set runtime-endpoint=<socket>`
Store the endpoint in the config

`crictl config --list`
Show client config values

`crictl --timeout <seconds> ps`
Raise the connection timeout

`crictl --debug ps`
Print debug output

`CONTAINER_RUNTIME_ENDPOINT`
Env var for the runtime socket

CONNECT (cont.)

`sudo -E crictl ps`
Keep env vars under sudo

`crictl --config <file> ps`
Use another config file

`crictl version`
Client and runtime versions

PODS

`crictl pods`
List pod sandboxes

`crictl pods --name <name>`
Filter pods by name

`crictl pods --label <key>=<value>`
Filter pods by label

`crictl pods --namespace <ns>`
Filter pods by namespace

`crictl pods --state Ready`
Only ready pods

`crictl pods -q`
Print only pod IDs

`crictl inspectp <pod>`
Pod sandbox details

`crictl stopp <pod>`
Stop a pod sandbox

`crictl rmp <pod>`
Remove a stopped pod

`crictl runp <pod.json>`
Create and start a pod sandbox

PODS (cont.)

`crictl runp --runtime <handler> <pod.json>`
Pod with a runtime handler

`crictl statsp`
Pod resource usage

CONTAINERS

`crictl ps`
List running containers

`crictl ps -a`
Include exited containers

`crictl ps --name <name>`
Filter containers by name

`crictl ps --pod <pod>`
Containers of one pod

`crictl ps --state exited`
Filter by state

`crictl ps -q`
Print only container IDs

`crictl inspect <container>`
Container details as JSON

`crictl stop <container>`
Stop a container

`crictl rm <container>`
Remove a container

`crictl start <container>`
Start a created container

`crictl create <pod> <ctr.json> <pod.json>`
Create a container in a pod

CONTAINERS (cont.)

`crictl run <ctr.json> <pod.json>`
Create and start in one step

`crictl update --memory <bytes> <container>`
Change resource limits

`crictl checkpoint --export=<file.tar> <container>`
Checkpoint to an archive

LOGS AND EXEC

`crictl logs <container>`
Show container logs

`crictl logs -f <container>`
Follow the log

`crictl logs --tail=<n> <container>`
Last N lines

`crictl logs --since=<duration> <container>`
Logs from the last period

`crictl logs --timestamps <container>`
Prefix lines with time

`crictl exec -it <container> sh`
Interactive shell

`crictl exec <container> <cmd>`
Run one command

`crictl attach <container>`
Attach to the main process

LOGS AND EXEC (cont.)

`crictl port-forward <pod> <local>:<remote>`
Forward a port from a pod

IMAGES

`crictl images`
List images

`crictl images <repo>`
Filter by repository

`crictl images --digests`
Show image digests

`crictl images -q`
Print only image IDs

`crictl pull <image>`
Pull an image

`crictl rmi <image>`
Remove an image

`crictl rmi --prune`
Remove unused images

`crictl inspecti <image>`
Image details

`crictl imagefsinfo`
Image filesystem usage

NODE AND METRICS

`crictl info`
Runtime status and config

`crictl stats`
Container CPU and memory

`crictl stats -a`
Include stopped containers

NODE AND METRICS (cont.)

`crictl events`
Stream runtime events

`crictl runtime-config`
Show runtime cgroup driver

`crictl ps -o json`
Machine-readable output

`crictl help <command>`
Help for one subcommand

RECIPES

Configure containerd once

```
sudo tee /etc/crictl.yaml
<<'EOF'
runtime-endpoint: unix:///run/
containerd/containerd.sock
image-endpoint: unix:///run/
containerd/containerd.sock
timeout: 10
EOF
sudo crictl info
```

Debug a crashing pod on the node

```
crictl pods --name <pod>
crictl ps -a --pod <pod-id>
crictl logs --tail=50
<container-id>
crictl inspect <container-id>
```

Find a container's exit code

```
crictl ps -a --name <name>
crictl inspect <container-id> |
jq '.status.exitCode'
```

Clean up a node

```
crictl ps -a --state exited -q
| xargs -r crictl rm
crictl rmi --prune
```