crictl CRI container debugging

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.

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

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

`CONTAINER_RUNTIME_ENDPOINT`
Env var for the runtime socket

PODS

`crictl pods`
List pod sandboxes

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

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

`crictl inspectp <pod>`
Pod sandbox details

CONTAINERS

`crictl ps`
List running containers

`crictl ps -a`
Include exited containers

CONTAINERS (cont.)

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

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

`crictl inspect <container>`
Container details as JSON

`crictl stop <container>`
Stop a container

`crictl rm <container>`
Remove a container

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 exec -it <container> sh`
Interactive shell

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

IMAGES

`crictl images`
List images

`crictl pull <image>`
Pull an image

`crictl rmi <image>`
Remove an image

NODE AND METRICS

`crictl info`
Runtime status and config

`crictl stats`
Container CPU and memory

`crictl ps -o json`
Machine-readable output