nerdctl Docker-like CLI for containerd

START HERE

`nerdctl run -d -p` *`<host>:<ctr> <image>`*
Start a container

`nerdctl build -t` *`<name> .`*
Build an image

`nerdctl ps`
List running containers

`nerdctl compose up -d`
Start a Compose stack

`nerdctl logs -f` *`<container>`*
Follow output

`nerdctl --namespace k8s.io ps`
See Kubernetes containers

`nerdctl exec -it` *`<container>`* **`sh`**
Shell in a container

`nerdctl system prune -a`
Clean up unused data

nerdctl CLI → gRPC → **containerd** images, namespaces → shim → **runc** OCI runtime → starts → **Container** CNI network

CONCEPTS

nerdctl is a Docker-compatible CLI that talks straight to containerd. Most docker commands and flags work unchanged; flags that are not implemented yet are rejected.

Containers and images live in a containerd namespace (default: default). Kubernetes uses k8s.io, so pass --namespace k8s.io to see its containers.

nerdctl build needs a running buildkitd. Rootless use needs a separate rootless containerd setup (see the rootless docs).

RUN

`nerdctl run` *`<image>`*
Create and start a container

`nerdctl run -d` *`<image>`*
Run in the background

`nerdctl run -it` *`<image>`* **`sh`**
Interactive shell

`nerdctl run --rm` *`<image>`*
Delete the container on exit

`nerdctl run --name` *`<name> <image>`*
Give the container a name

`nerdctl run -p` *`<host>:<ctr> <image>`*
Publish a port
Format is [HOST_IP:]HOST_PORT:CONTAINER_PORT[/PROTO].

`nerdctl run -v` *`<name>:<path> <image>`*
Mount a named volume

`nerdctl run -v` *`<dir>:<path>:ro <image>`*
Bind-mount read-only

`nerdctl run -e KEY=VALUE` *`<image>`*
Set an environment variable

`nerdctl run --env-file` *`<file> <image>`*
Load variables from a file

`nerdctl run --net` *`<network> <image>`*
Join a network
May be given more than once to join several networks.

`nerdctl run --restart unless-stopped` *`<image>`*
Restart after crashes and reboots

`nerdctl create` *`<image>`*
Create without starting

CONTAINERS

`nerdctl ps`
List running containers

`nerdctl ps -a`
List all, including stopped

`nerdctl stop` *`<container>`*
Stop gracefully

`nerdctl start` *`<container>`*
Start a stopped container

`nerdctl restart` *`<container>`*
Stop and start again

`nerdctl rm` *`<container>`*
Remove a stopped container

`nerdctl rm -f` *`<container>`*
Force-remove a running one

`nerdctl ps -q`
Print only container IDs

`nerdctl ps -f status=exited`
Filter the list

`nerdctl stats --no-stream`
One-shot CPU and memory usage

INSIDE A CONTAINER

`nerdctl exec -it` *`<container>`* **`sh`**
Open a shell in a running container

`nerdctl exec` *`<container> <cmd>`*
Run one command

`nerdctl logs` *`<container>`*
Show output
Only containers started with run -d are supported.

`nerdctl logs -f` *`<container>`*
Follow output live

`nerdctl logs -n` *`<lines> <container>`*
Only the last lines

`nerdctl logs --since` *`<duration> <container>`*
Only recent output, e.g. 10m

`nerdctl cp` *`<container>:<path> <dest>`*
Copy out of a container
Use only with trusted containers.

`nerdctl cp` *`<src> <container>:<path>`*
Copy into a container

`nerdctl inspect` *`<object>`*
Low-level details as JSON

IMAGES

`nerdctl pull` *`<image>`*
Download an image

IMAGES (cont.)

`nerdctl images`
List local images

`nerdctl build -t` *`<name> .`*
Build from a Dockerfile here

`nerdctl build -f` *`<file>`* **`-t`** *`<name> .`*
Use another Dockerfile

`nerdctl build --build-arg KEY=VALUE .`
Pass a build variable

`nerdctl build --no-cache .`
Ignore the build cache

`nerdctl tag` *`<image> <repo>:<tag>`*
Add a name to an image

`nerdctl push` *`<repo>:<tag>`*
Upload to a registry

`nerdctl login` *`<registry>`*
Log in to a registry
Use --password-stdin in scripts.

`nerdctl rmi` *`<image>`*
Remove an image

`nerdctl save -o` *`<file> <image>`*
Export an image to a tar file

`nerdctl load -i` *`<file>`*
Import an image from a tar file

`nerdctl pull --platform` *`<os/arch> <image>`*
Pull another platform

VOLUMES AND NETWORKS

`nerdctl volume create` *`<name>`*
Create a named volume

`nerdctl volume ls`
List volumes

`nerdctl volume rm` *`<name>`*
Remove a volume

`nerdctl volume inspect` *`<name>`*
Show volume details

`nerdctl volume prune`
Remove unused volumes

`nerdctl network create` *`<name>`*
Create a bridge network

`nerdctl network ls`
List networks

COMPOSE

`nerdctl compose up -d`
Create and start the stack

`nerdctl compose down`
Stop and remove containers and networks

COMPOSE (cont.)

`nerdctl compose ps`
List the stack's containers

`nerdctl compose logs -f` *`<service>`*
Follow service logs

`nerdctl compose exec` *`<service>`* **`sh`**
Shell in a service

`nerdctl compose up --build`
Rebuild images, then start

`nerdctl compose build`
Build service images

`nerdctl compose pull`
Pull service images

`nerdctl compose stop`
Stop without removing

`nerdctl compose down -v`
Also remove volumes
Deletes named volumes declared in the file: data is lost.

`nerdctl compose config`
Show the resolved file

NAMESPACES AND DAEMON

`nerdctl --namespace` *`<ns> <cmd>`*
Run against a containerd namespace
Containers and images are per namespace; k8s.io holds Kubernetes workloads.

`nerdctl namespace ls`
List namespaces

`nerdctl --address` *`<socket> <cmd>`*
Use another containerd socket

`nerdctl info`
Show system information

`nerdctl version`
Client and server versions

`nerdctl system df`
Show disk usage

`nerdctl system prune`
Remove stopped containers and dangling data

RECIPES

Inspect Kubernetes containers

```
nerdctl --namespace k8s.io ps
nerdctl --namespace k8s.io images
nerdctl --namespace k8s.io logs <container>
```

Throwaway shell with your code

```
nerdctl run --rm -it -v "$PWD":/work -w /work \
  <image> sh
```