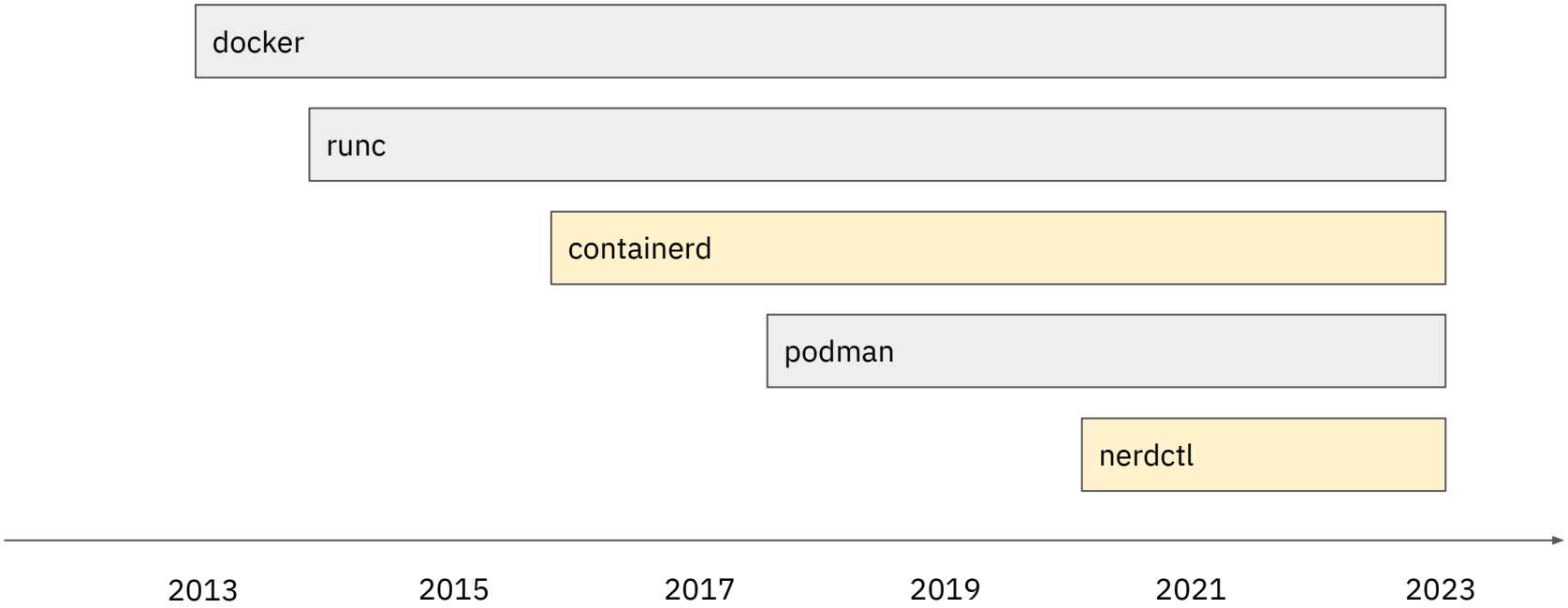
nerdctl and containerd as an alternative to Docker and Podman

Danish Prakash, SUSE

agenda

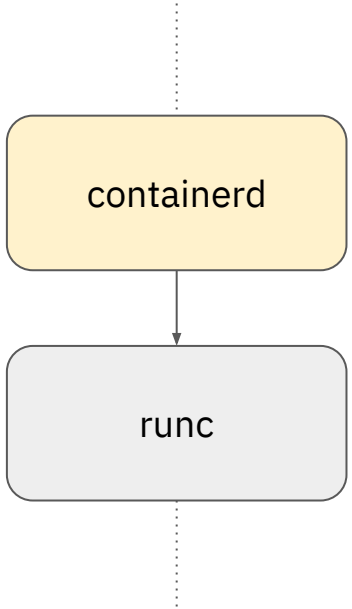
- container engine/runtime evolution
- introducing containerd and nerdctl
- comparison
- containerd/nerdctl features
- getting started w/ containerd/nerdctl
- wrap-up

container runtime/engine evolution
(so far)



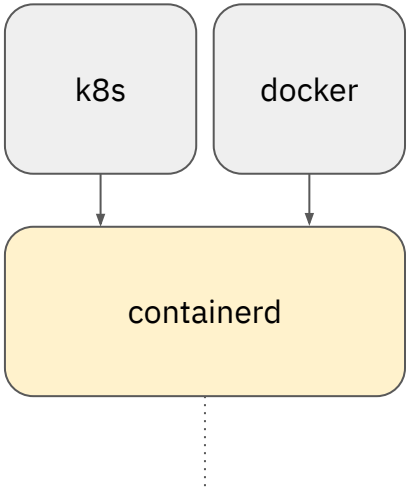
containerd

- High-level container runtime
- uses runc internally to manage container lifecycle
- Features:
  - Container image management
  - Network management
  - Container supervision



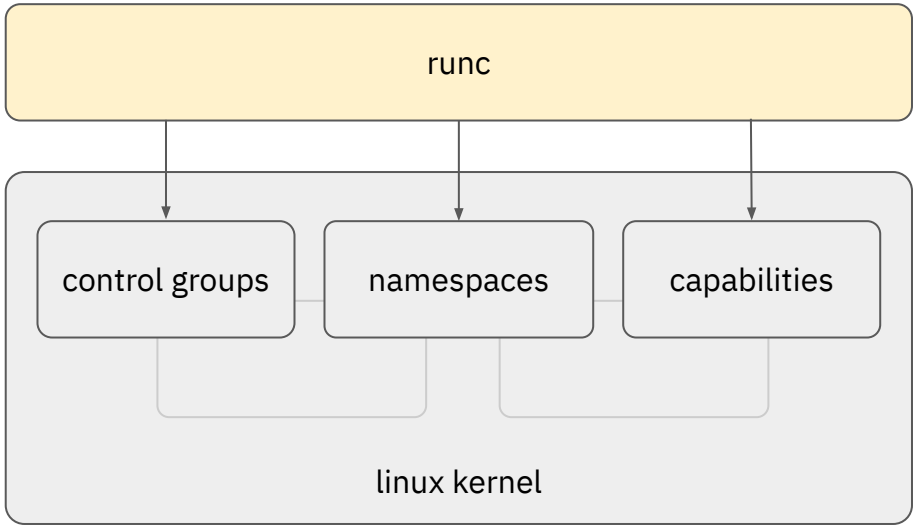
containerd (history)

- Developed initially by Docker and then open-sourced
- Initial goals:
  - Designed as a daemon to manage runc
  - Pave the way for advanced functionality
  - To be embedded into larger systems (docker, k8s)
  - Not to be used directly by developers or end-users.
  - Have a cleaner interface and API



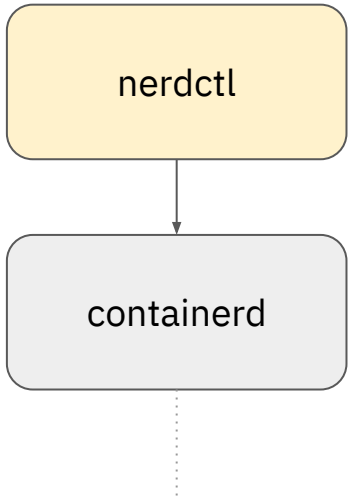
runc

- lightweight, portable low-level container runtime
- unified low-level component dealing with the Linux Kernel



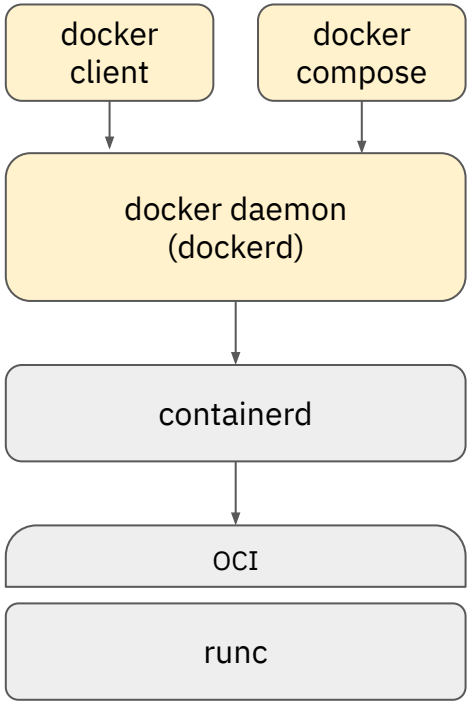
nerdctl

- Docker-compatible (!competing) CLI for containerd
- Designed to experiment and innovate with new features
- (experimental) Features:
  - Lazy image pulling
  - Interactive Dockerfile debugging
  - Registry-less container image distribution (IPFS)
  - Container image signing (cosign/notary)

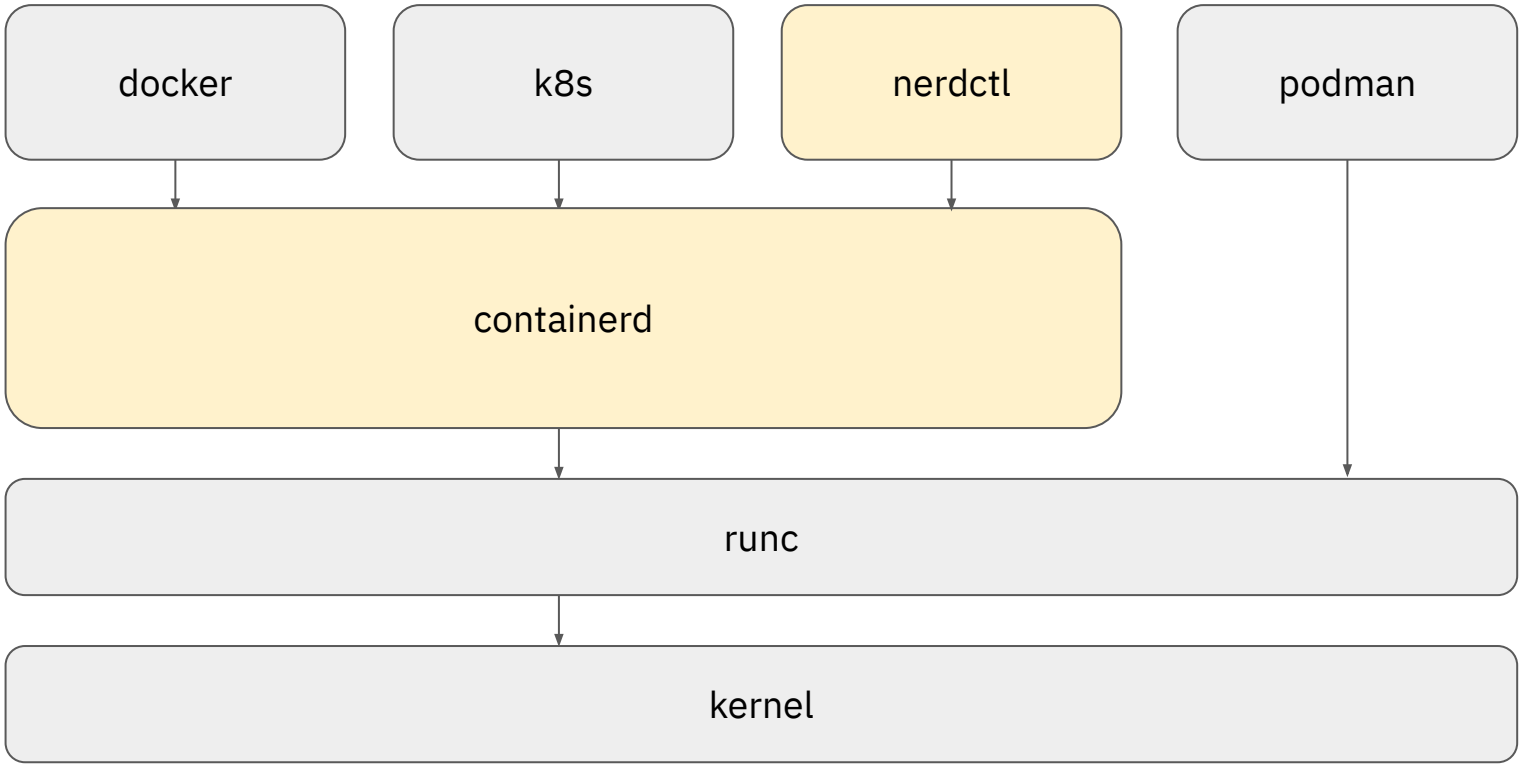


docker

- widespread container adoption
- container engine
- provides high-level CLI and API



how do they work together?



ctl, crictl, nerdctl ??

- ctl – debugging utility bundled along with containerd,
  not meant to be used by users
- crictl – cli utility for kubernetes kubelet CRI (container runtime interface)
- nerdctl – docker-compatible end-user cli for containerd

comparison

	rootless	container lifecycle	image management	network management
containerd /nerdctl	nerdctl >= 0.6.0	standard operations	standard operations along with image encryption	supports high-throughput rootless networking via bypass4nets
docker	>= 19.03	standard operations	standard operations	rootless networking via vpnkit or slirp4nets
podman	one of the initial goals	standard operations	standard operations along with image encryption	supports rootless networking using slirp4nets

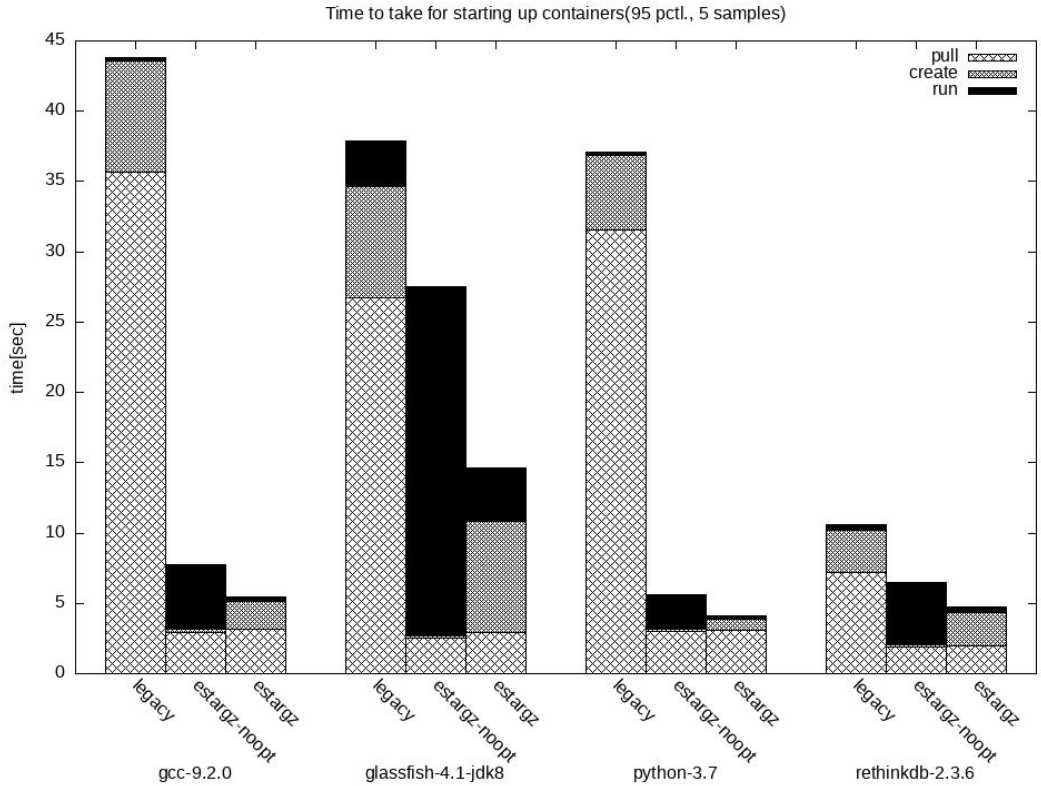
nerdctl features
+demo

lazy image pulling

- *“..pulling packages accounts for 76% of container start time, but only 6.4% of that data is read.”*
- avoid pulling the complete image
- running containers before image pull is completed

```
$ nerdctl run \
    --snapshotter=stargz \
    --rm ghcr.io/stargz-containers/python:3.7-esgz \
    python3 -c 'print("hello lazy-pulling")'
```

lazy image pulling



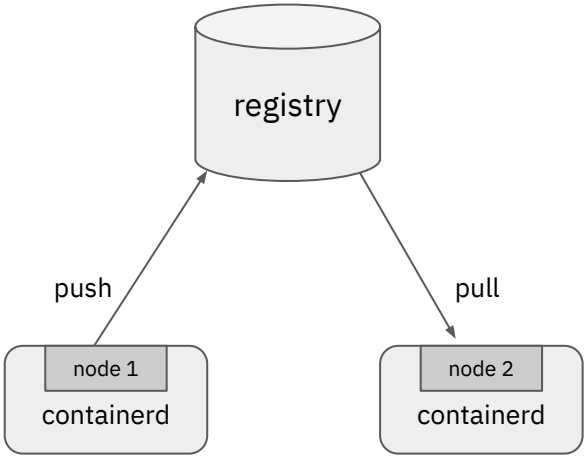
registry-less image distribution (IPFS)

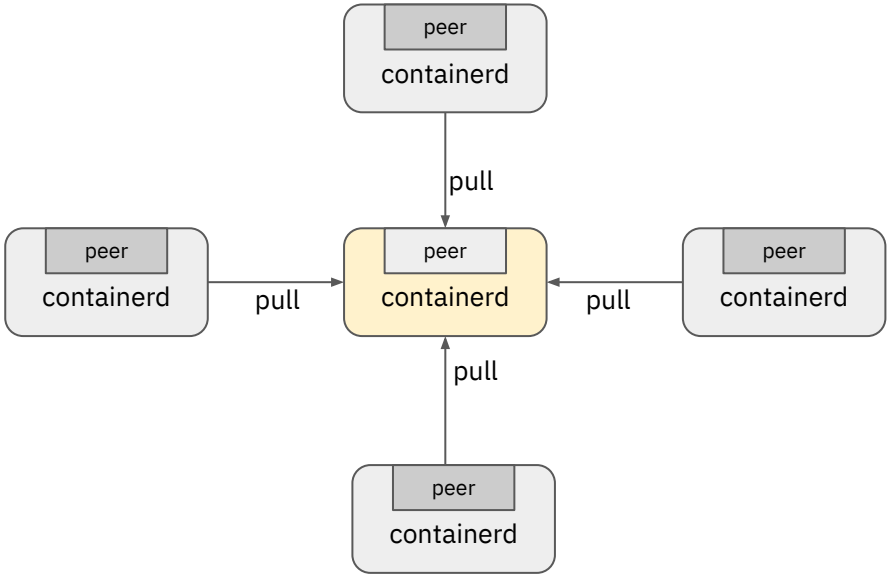
- eliminate the need for a centralized container registry
- E.g. node-to-node peer image sharing in a k8s cluster

```
$ nerdctl push ipfs://alpine:latest
INFO[0000] pushing image "alpine:latest" to IPFS
INFO[0000] ensuring image contents
bafkreiferwtutmt5ll6v25rlrojzmpqrxwflnspuvwobu3iu6rbq3xasqe
```

```
$ nerdctl run --rm -it ipfs://bafkreiferwtutmt5ll6v25rlrojzmpqrxwflnspuvwobu3iu6rbq3xasqe echo hello
...
...
hello
```

registry-less image distribution (IPFS)





dockerfile debugging

- supports interactive debugging of Dockerfiles (via `buildg`)
- ability to add breakpoints in Dockerfile for debugging

```
$ nerdctl builder debug .
...
...
Filename: "Dockerfile"
 =>   1| FROM alpine
      2| RUN apk add figlet
      3| RUN figlet "hello"
(buildg) break 2
```

running containers from encrypted images

- encrypt images `nerctl image encrypt …`
- ensure private keys present in `/etc/containerd/ocicrypt/keys`
- uses imgcrypt (OCIcrypt) for encryption/decryption

```
$ openssl genrsa -out mykey.pem
$ openssl rsa -in mykey.pem -pubout -out mypubkey.pem
$ nerdctl image encrypt \
    --recipient=jwe:mypubkey.pem \
    --platform=linux/amd64,linux/arm64 \
    test danishprakash/test:encrypted
```

rootless networking using bypass4nets

- slirp4nets is slower due to virtualized networking in usermode TCP/IP
- bypass4netns avoids this overhead by syscall exec in host ns

```
$ containerd-rootless-setuptool.sh install-bypass4netnsd
$ nerdctl run --label nerdctl/bypass4netns=true alpine echo hello
hello
```

getting started with containerd-nerdctl

```
# rootfull
$ sudo zypper install containerd nerdctl
$ sudo systemctl start containerd
$ sudo nerdctl run -it alpine echo hello
hello

# rootless
$ containerd-rootless-setuptool.sh install
$ nerdctl run -it alpine echo hello
hello
```

https://github.com/containerd/containerd
https://github.com/containerd/nerdctl
https://github.com/containerd/nerdctl/blob/main/extras/rootless/containerd-rootless-setuptool.sh

wrap-up

- try out containerd/nerdctl as viable container runtimes
- allows for more innovation in future in the containerization domain
- container runtime/engine space is continuously evolving

thank you!
questions?