DockerからcontainerdへのNTT移行

NTT Tech Conference 2022 (March 23rd)

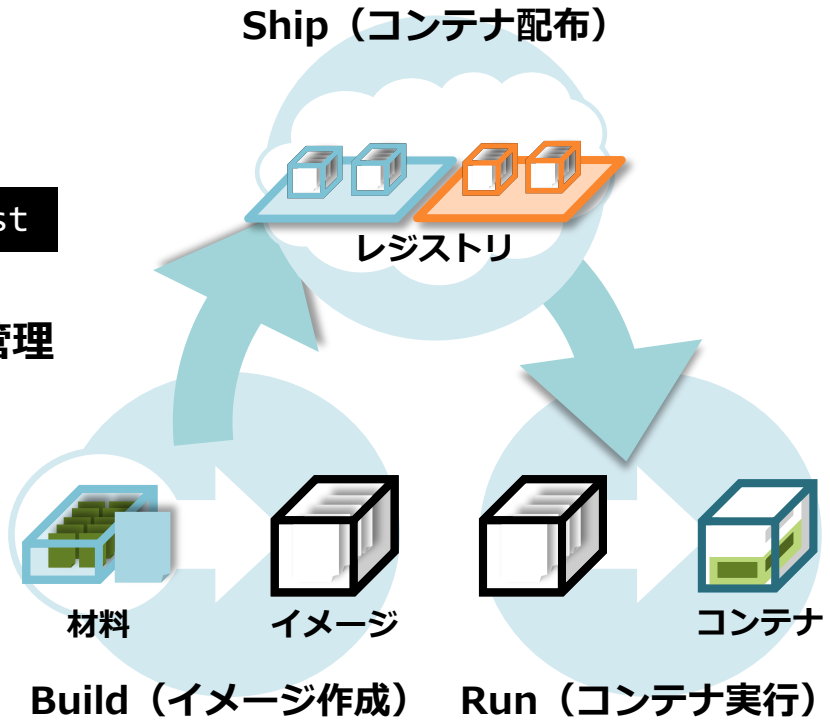
日本電信電話株式会社
ソフトウェアイノベーションセンタ
徳永　航平
須田　瑛大

Dockerの復習

- コンテナ開発に有用な機能を提供するツール

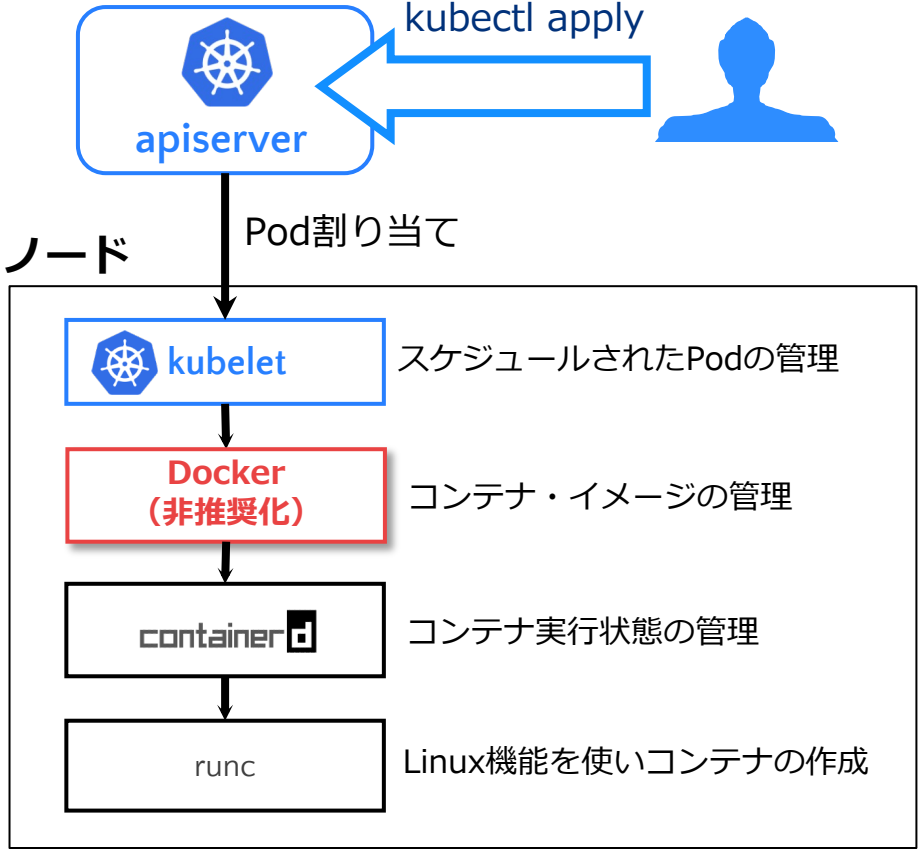
Build	`docker build -t foo /dockerfile-dir`
Ship	`docker run -it --rm alpine`
Run	`docker push ghcr.io/ktock/myalpine:latest`

- docker composeで複数コンテナをまとめて管理可能
- Mac,WindowsではDocker Desktopを利用可能



Dockerに関する最近の話題：Kubernetesでのdockershim非推奨化

- Kubernetes v1.23まではノード上でDockerをコンテナランタイムとして利用可能
- Kubernetes v1.20からkubeletからDocker利用は非推奨に
- Kubernetes v1.24ではDockerのサポート(dockershim)を削除予定
  - ○ 今後独立のツールとしてMirantis社によってメンテされる
  - ○ https://github.com/Mirantis/cri-dockerd



Dockerに関する最近の話題：Docker Desktop有償化

- Docker Desktop 有償化：*"professional use in larger businesses"*対象
  - https://www.docker.com/blog/updating-product-subscriptions/

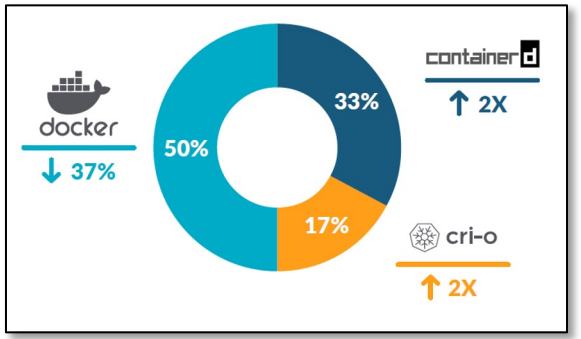
- GUI版のDocker Desktopは有償だが、Docker CLI + Docker EngineのVM内での利用は有償化されない
  - https://www.docker.com/pricing/faq
    - “Can I just install the Docker CLI instead of using Docker Desktop?”

containerdによるコンテナ管理

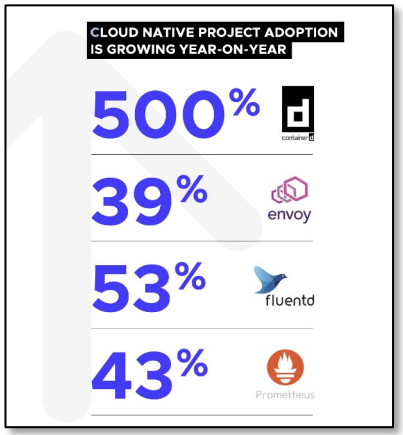
containerdによるコンテナ管理

https://github.com/containerd/containerd

- オープンソースなコンテナランタイム（CNCF graduatedプロジェクト）
- Docker/Moby、Kubernetesサービス（EKS、AKS、GKE）で使われるなど、採用が拡大している



引用元：https://sysdig.com/blog/sysdig-2021-container-security-usage-report/
(2021年1月)

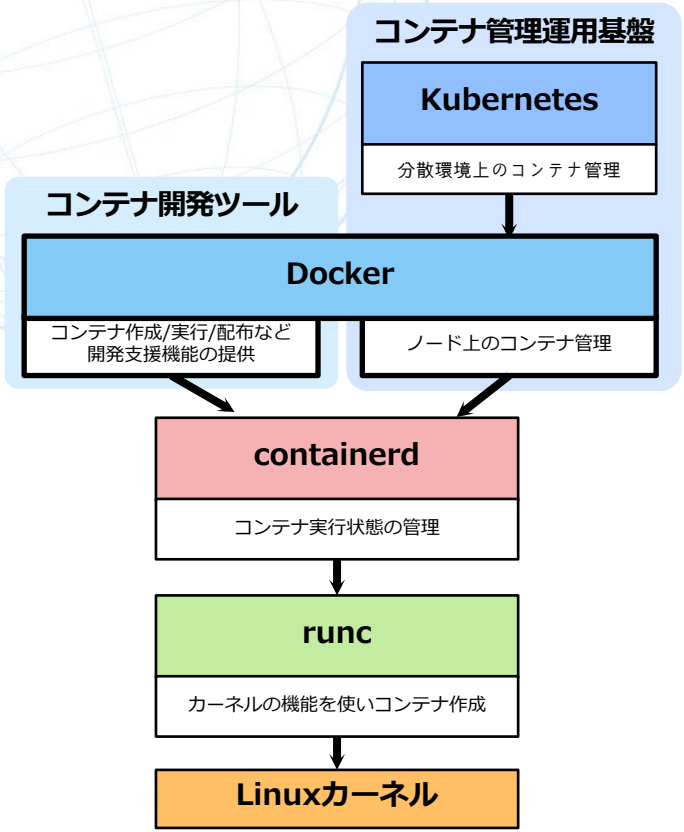


引用元：
https://www.cncf.io/reports/cncf-annual-survey-2021/
(2022年2月)

containerdとDocker、Kubernetesの関係

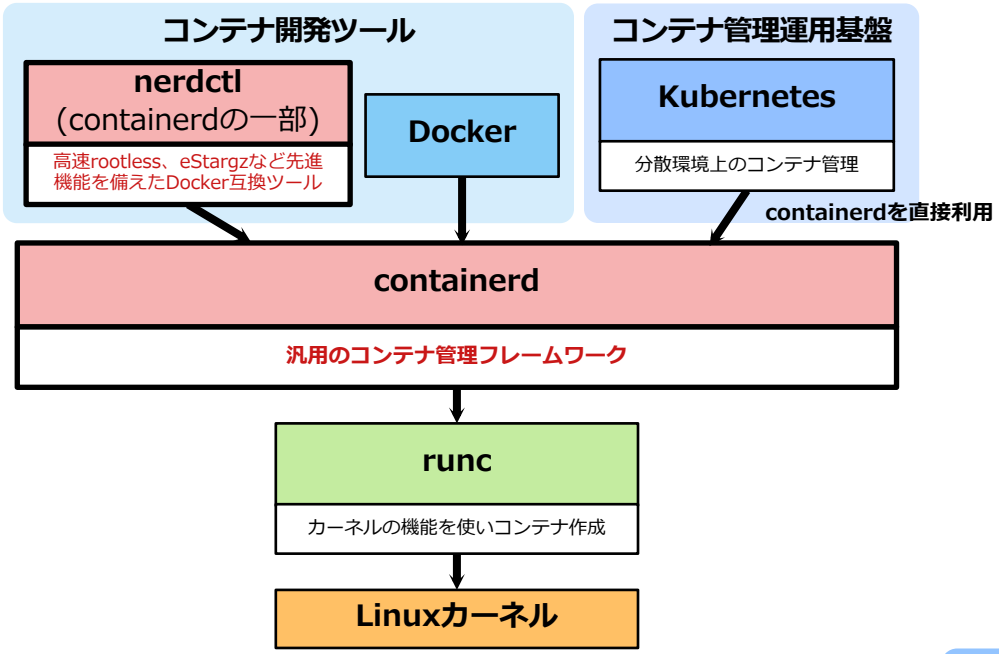
NTT

従来



近年

containerd だけで Docker の機能をほぼカバー

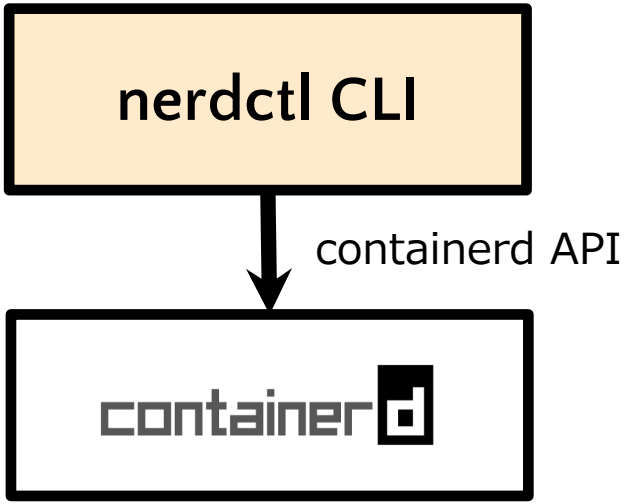


nerdctl：Docker互換なcontai*nerd*用CLI

https://github.com/containerd/nerdctl

- **Docker風のUI/UX（composeもサポート）**

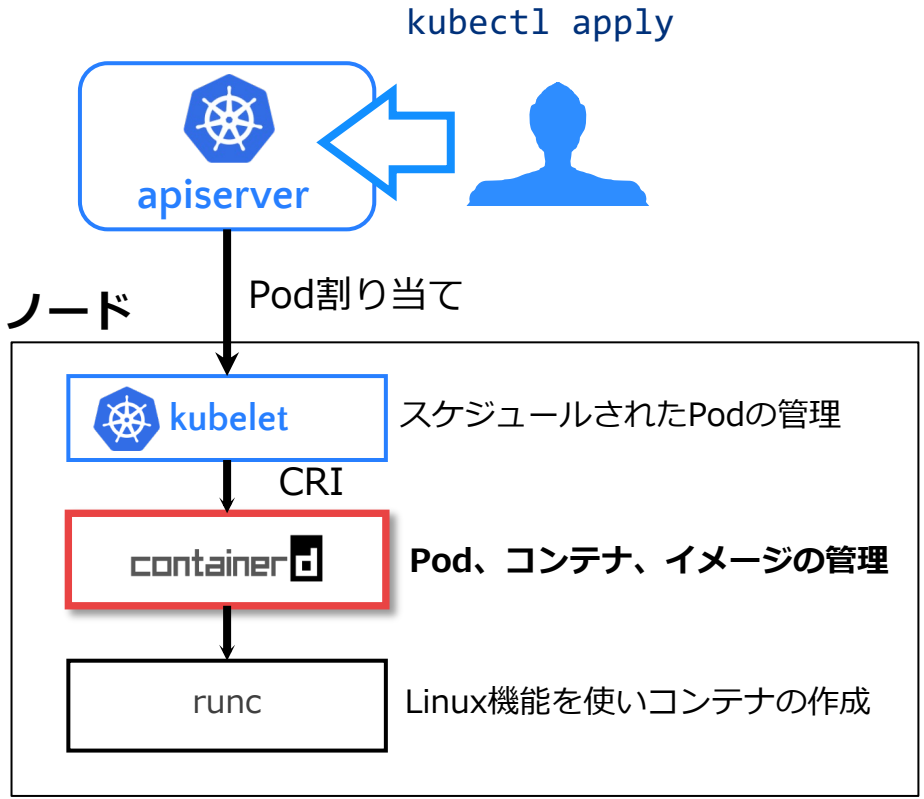
```
nerdctl build -t foo /dockerfile-dir
nerdctl run -it --rm alpine
nerdctl push ghcr.io/ktock/myalpine:latest
nerdctl compose up
```



- **先進機能**
  - 高速イメージpull（eStargz）
  - P2Pイメージ共有（IPFS）
  - 暗号化イメージ （OCIcrypt）
  - イメージへの署名/検証（cosign）
  - 高速rootless（bypass4netns）

containerdによるKubernetesでのコンテナ管理

- 各ノード上でコンテナランタイムとしてcontainerdを利用可能
- kubeletはContainer Runtime Interface（CRI）経由でcontainerdを直接利用
- EKS、AKS、GKEなどマネージドKubernetesサービスでも利用可能



GKEでのcontainerdへの移行

https://cloud.google.com/kubernetes-engine/docs/deprecations/docker-containerd
https://cloud.google.com/kubernetes-engine/docs/how-to/migrate-containerd

- Dockerベースのノードイメージは1.24以降でサポートされなくなる
- containerdベースのノードイメージ（Linuxノードでは1.19、Windowsでは1.21からデフォルト）に移行する必要がある

Dockerノードイメージ	Containerdノードイメージ
Dockerを含むContainer-Optimized OS (cos)	Containerdを含むContainier-Optimized OS (cos_containerd)
Dockerを含むUbuntu(ubuntu)	Containerdを含むUbuntu(ubuntu_containerd)
Dockerを含むWindows Server LTSC(windows_ltsc)	Containerdを含むWindows Server LTSC(windows_ltsc_containerd)
Dockerを含むWindows Server SAC(windows_sac)	Containerdを含むWindows Server SAC(windows_sac_containerd)

- 移行作業のためのスクリプト`find-nodepools-to-migrate.sh`が提供されている
  - クラスタを解析して、移行のための推奨コマンドを示してくれる
    https://github.com/GoogleCloudPlatform/k8s-node-tools/blob/HEAD/migrating-to-containerd/find-nodepools-to-migrate.sh
- 推奨されたコマンド（`gcloud container clusters upgrade`）を使ってcontainerdに移行可能

EKSでのcontainerdへの移行

https://docs.aws.amazon.com/ja_jp/eks/latest/userguide/dockershim-deprecation.html
https://docs.aws.amazon.com/ja_jp/eks/latest/userguide/eks-optimized-ami.html#containerd-bootstrap

- Amazon Linux AMIは1.21現在でDockerがデフォルトランタイム
- バージョン1.23以降、含まれるランタイムはcontainerdのみになる
- 1.21現在までのAMIでも、ブートストラップフラグ`--container-runtime containerd`でcontainerdノードを利用可能

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
  - name: my-nodegroup
    ami: eks-optimized-AMI-ID
    overrideBootstrapCommand: |
      #!/bin/bash
      set -eu -o pipefail
      /etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

- あるいは、ランタイムとしてcontainerdを利用しているBottlerocket AMIまたはEKS Fargateに移行するのも可
- WindowsノードでもKubernetes 1.20より新しいバージョンでcontainerdを選択可能
  - https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-windows-ami.html#containerd-bootstrap-windows

AKSでのcontainerdへの移行

https://docs.microsoft.com/en-us/azure/aks/cluster-configuration#container-runtime-configuration

- LinuxノードではKubernetes v1.19からcontainerdがランタイムとして使われている
- AKSのバージョンを`az aks upgrade`などで1.19以降にアップグレードすることでLinuxノードについてはcontainerdへ移行可能
- とはいえすでに1.19以前はEOLのため、サポートされているAKS全バージョンでLinuxノードではcontainerdが使われていることになる
- Windowsノードでは1.20からcontainerdを利用可能。1.22以前のクラスタではDockerがデフォルト。1.23からはcontainerdがデフォルトになる。
  - `az aks nodepool upgrade`などで移行可能：https://docs.microsoft.com/en-us/azure/aks/windows-container-cli#optional-using-containerd-with-windows-server-node-pools

kubeadmでのcontainerdへの移行

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#initializing-your-control-plane-node

- ノードにDockerが稼働していればcontainerdも既に稼働している
- kubeadm(v1.23現在)はDockerを優先的に使用するので、明示的にcontainerdのソケットパス`--cri-socket=/run/containerd/containerd.sock`を指定する必要がある

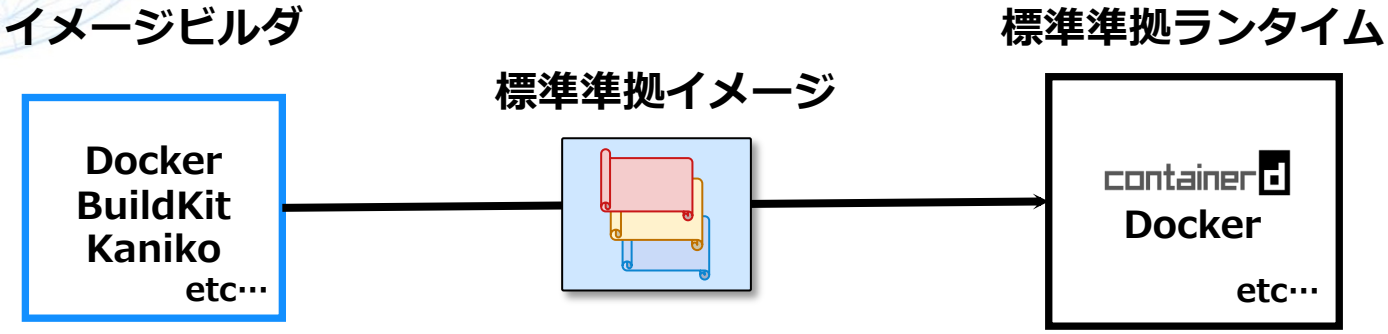
```
sudo kubeadm reset
sudo kubeadm join 192.168.56.120:6443 --cri-socket=/run/containerd/containerd.sock
--token <省略> --discovery-token-ca-cert-hash <省略>
```

- ここで、containerdのconfig（/etc/containerd/config.toml）に`disabled_plugins = [“cri”]`が指定されてないことを要確認

FAQ: イメージを作り直す必要はあるか？

A. No。既存のコンテナイメージがそのままcontainerdで使える。

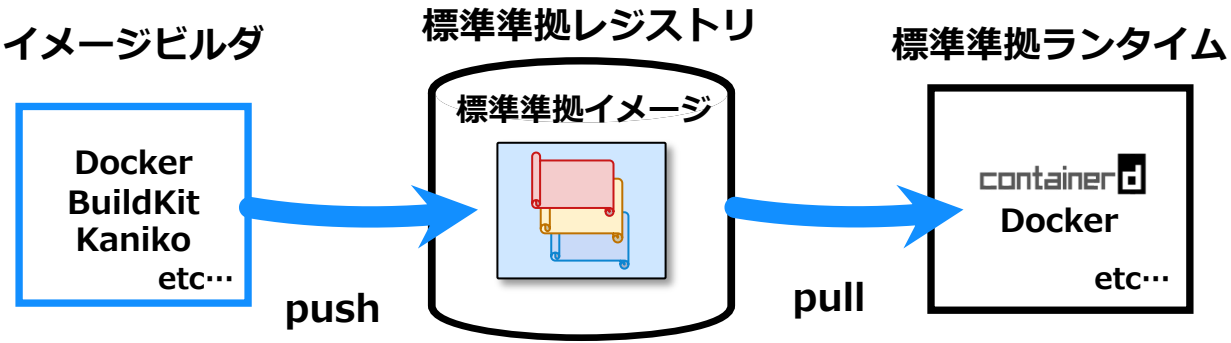
- イメージには標準仕様があるのでcontainerd移行後も引き続き実行可能



FAQ: レジストリはそのまま使えるか？

A. Yes。ただしレジストリミラー等の設定は移行が必要。

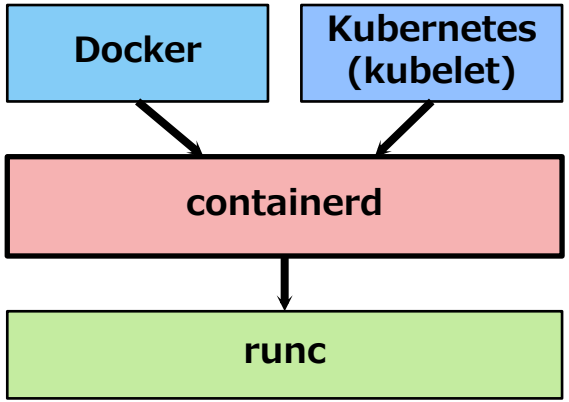
- レジストリには標準仕様があるのでcontainerd移行後も引き続き利用可能
- ただし、`/etc/docker/`にレジストリクライアント関連の設定（ミラーや証明書など）をしていた場合は、`/etc/containerd/`で設定しなおす必要がある
  - 設定方法(containerd >= 1.5)：
    https://github.com/containerd/containerd/blob/v1.6.1/docs/hosts.md



FAQ: ノードの作り直しは必要か？

A. クラウドの場合：Yes。
kubeadmの場合：厳密にはNo。運用上はYes。

- GKE/EKS/AKSでは前述の通りノードの更新が必要。
- kubeadmでは、厳密にはNo。運用上はYes。
  - Dockerが稼働していればcontainerdも稼働している
  - しかしDockerを維持する理由がないならばDocker無しでノードを作り直した方が良い



FAQ: ノードのデバッグに`docker`コマンドは使い続けられるか？

A. No。代わりにnerdctlが使える。

`--namespace=k8s.io`を指定するとKubernetesのコンテナをDocker同様のUIでデバッグ可能

```
$ sudo nerdctl --namespace=k8s.io ps
CONTAINER
ID    IMAGE                                    COMMAND                   CREATED          STATUS    PORTS
   NAMES
1fe5fe717d05    k8s.gcr.io/pause:3.5                          "/pause"                  9 minutes
ago    Up                  k8s://kube-system/etcd-ktock
226a45b850fa    k8s.gcr.io/kube-controller-manager:v1.23.4    "kube-controller-man…"    9 minutes
ago    Up                  k8s://kube-system/kube-controller-manager-ktock/kube-controller-manager
```

- 他にもノード上で利用可能なCLIはあるが、独特なUIに慣れが必要
  - **ctr**: containerdに付属するデバッグ専用CLI。containerd APIに肉薄した操作（e.g. snapshots、contents）や、containerd自体のデバッグに有用。
    - https://github.com/containerd/containerd/tree/main/cmd/ctr
  - **crictl**: Kubernetes SIG Nodeで開発されるCRI用CLI。YAMLファイルでPod,コンテナ,イメージを操作可能
    - https://github.com/kubernetes-sigs/cri-tools/blob/master/docs/crictl.md

FAQ: ノードのデバッグに`docker`コマンドは使い続けられるか？

A. No。代わりにnerdctlが使える。

```
$ mkdir -p /tmp/ctx && cat <<EOF > /tmp/ctx/Dockerfile
FROM alpine:3.15
CMD [ "sh", "-c", "while true ; do echo hello ; sleep 1 ; done" ]
EOF
$ sudo nerdctl --namespace k8s.io build -t foo /tmp/ctx
$ kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: foo
spec:
  containers:
    - name: foo
      image: foo
      imagePullPolicy: Never
EOF
$ kubectl logs foo
hello
```

`nerdctl build`を使ってノード上で
イメージをビルドして実行することも可能

FAQ: ログ管理に影響はあるか？

NTT

A. ログ基盤がDocker依存の設定をもつ場合は修正の必要あり。

- `/var/log/pods`のログフォーマットはDockerとCRIランタイムで異なる
- 非マネージドなログ基盤で、Docker用のパーサしか設定されていなかったり、Docker APIに依存してログ収集しているなど、Docker依存の設定がある場合は変更が必要

Dockerのログフォーマット

```
{"log":"Log line is here¥n","stream":"stdout","time":"2019-01-01T11:11:11.111111111Z"}
```

CRIランタイム(containerd含)のログフォーマット

```
2016-10-06T00:17:09.669794202Z stdout P log content 1
```

https://docs.docker.com/config/containers/logging/json-file/

https://github.com/kubernetes/kubernetes/blob/9a8defda15e4c34e9c198975968d9619f48a0786/pkg/kubelet/kuberuntime/logs/logs.go#L125-L169

FAQ: Pod内で`docker`コマンドは引き続き利用可能か？

A. Pod内でDockerデーモンを稼働させる場合：Yes
ノード上のDockerに依存する場合：クラウドでは移行が推奨されている。
kubeadm利用時はYes。

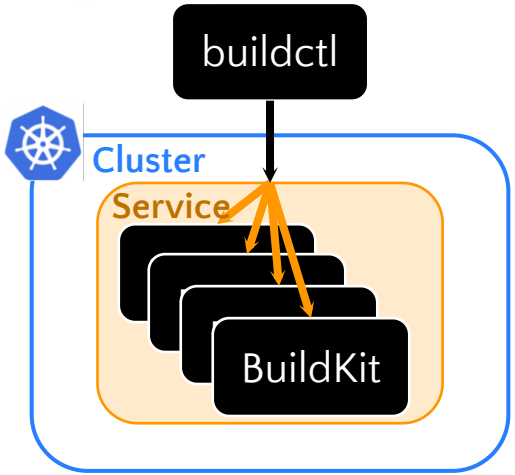
- **GKE**: Dockerに直接依存しないようワークロードをアップデートすべき。
  - https://cloud.google.com/kubernetes-engine/docs/deprecations/docker-containerd#impact_of_migrating
- **EKS**: Docker socketをマウントするアプリケーションでは変更が必要になる。
  - https://docs.aws.amazon.com/ja_jp/eks/latest/userguide/dockershim-deprecation.html
- **AKS**: Dockerは利用不可。
  - https://docs.microsoft.com/en-us/azure/aks/cluster-configuration#containerd-limitationsdifferences
- **kubeadm:** ノード上でCRIランタイムとしてcontainerdを使いつつ、Dockerも稼働させておけば引き続き可能。

FAQ: Pod内で`docker`コマンドは引き続き利用可能か？

Kubernetes上でのビルドにはBuildKitの利用も有用
https://github.com/moby/buildkit/tree/master/examples/kubernetes

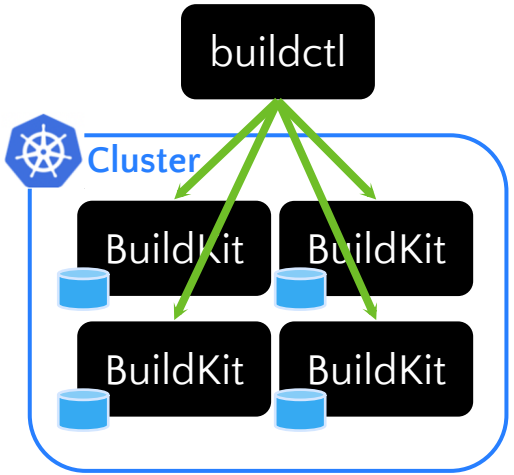
Deployment+Service

ランダムロードバランシング。レジストリを介したキャッシュ共有により高速なビルドが可能。



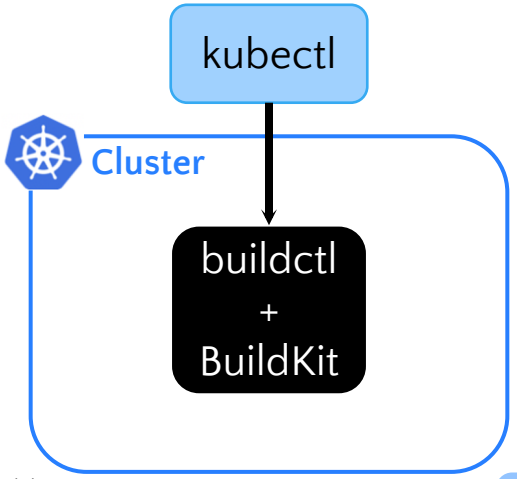
StatefulSet

クライアント側でのロードバランシング。consistent hashを使えば効率的にローカルキャッシュを活用可能。



Job

ワンショットなビルド実行。BuildKitデーモンの稼動が不要。ローカルキャッシュはビルドのたびに削除。



containerd・nerdctlの持つ様々な機能

containerd・nerdctlの持つ様々な機能

- nerdctl コマンドの使い方は、docker コマンドの使い方とだいたい同じ
- 単に docker コマンドの置き換えを目指しているわけではなくて、コンテナランタイム関連の新機能を実験、普及展開するためのプラットフォームとして開発している
  - ○ 高速イメージpull（eStargz）
  - ○ P2Pイメージ共有（IPFS）
  - ○ 暗号化イメージ （OCIcrypt）
  - ○ イメージへの署名/検証（cosign）
  - ○ 高速rootless（bypass4netns）

高速なイメージPull（eStargz）

https://github.com/containerd/nerdctl/blob/master/docs/stargz.md

✅nerdctl
✅Kubernetes

NTT

- **Lazy pulling:** イメージのpull完了を待たずにコンテナを起動可能
- **eStargz**: OCI仕様に後方互換のlazy pulling可能なイメージ形式
  - lazy pulling非対応なランタイムでも通常のイメージとして扱える

```
$ nerdctl --snapshotter=stargz run -it ghcr.io/stargz-containers/python:3.9-esgz
```

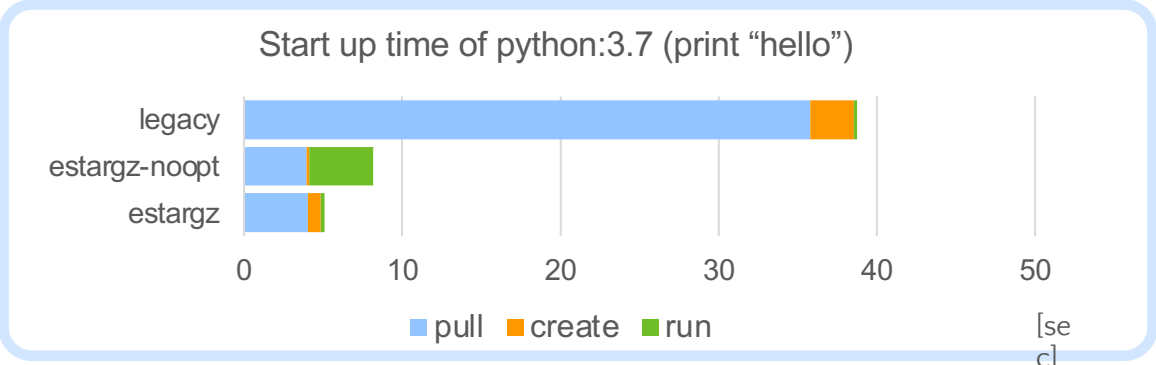


Figure from “Faster Container Image Distribution on a Variety of Tools with Lazy Pulling - Kohei Tokunaga & Tao Peng. KubeCon+CloudNativeCon North America 2021. https://sched.co/lV2a “

P2Pイメージ共有（IPFS）

Experimental

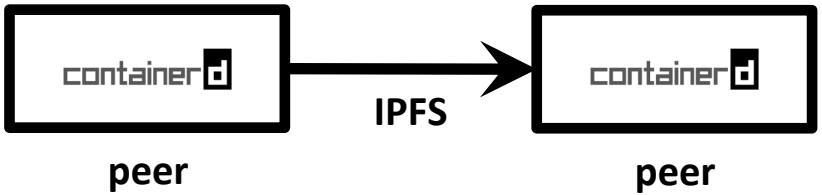
✅nerdctl
✅Kubernetes

※Kubernetesではnerdctl付属のOCI互換プロキシを用いる

https://github.com/containerd/nerdctl/blob/master/docs/ipfs.md

- P2Pデータ共有プロトコルのIPFSを使ってコンテナイメージをP2P共有可能
- nerdctl各コマンドで”**ipfs://CID**“の形式でIPFS上のイメージを利用可能
- eStargzのlazy pulling、OCICryptなども依然利用可能

```
nerdctl push ipfs://ubuntu:20.04
```

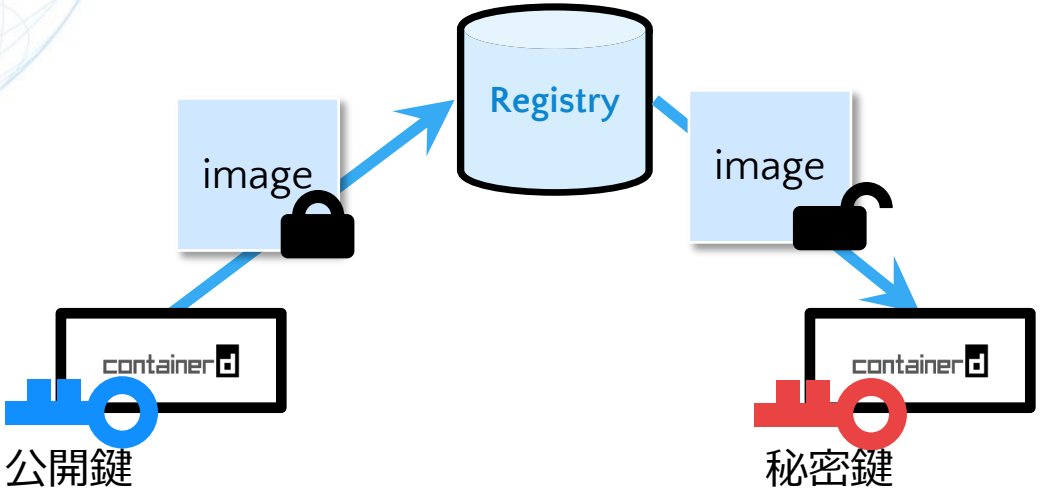



```
nerdctl run ipfs://bafkreicq4dg6nkef5ju422ptedcwfz6kcvpvvhuqeykfrwq5krazf3muze
```

イメージ暗号化・復号化（OCICrypt）

https://github.com/containerd/nerdctl/blob/master/docs/ocicrypt.md

- キーペアを用いて、イメージの暗号化が可能
- 暗号化したイメージはcontainerdでPullして復号、実行できる




```
nerdctl image encrypt
```

イメージの暗号化

```
nerdctl image decrypt
```

イメージの復号

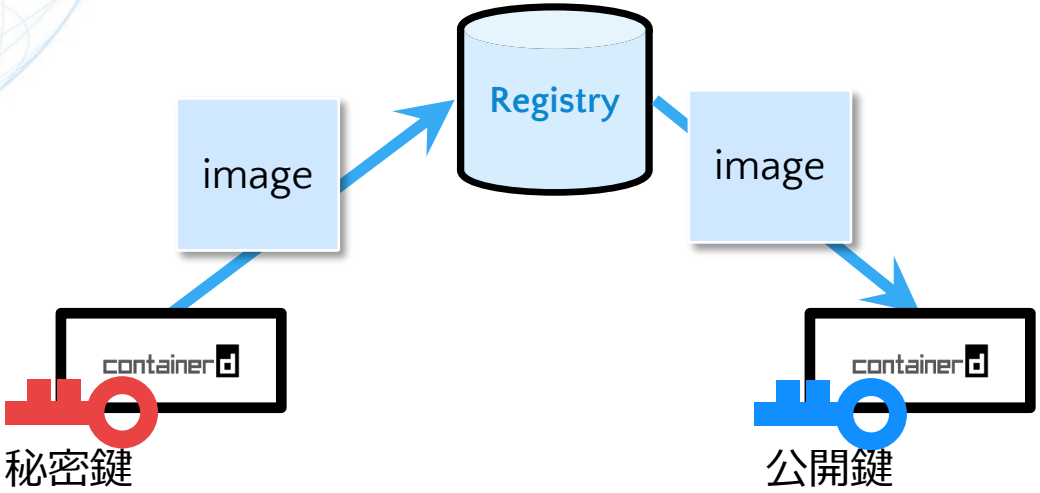
イメージ署名・検証（cosign）

https://github.com/containerd/nerdctl/blob/master/docs/cosign.md

✅nerdctl
✅Kubernetes

※Kubernetesではcosign付属のadmission controllerを使用

- 署名を用いて、イメージが改竄されていないかを検証できる
- Notaryに似ているが、最近はcosignが流行りつつある




```
nerdctl push --sign=cosign
```

イメージの署名

```
nerdctl pull --verify=cosign
```

イメージの検証

高速rootless（bypass4netns）

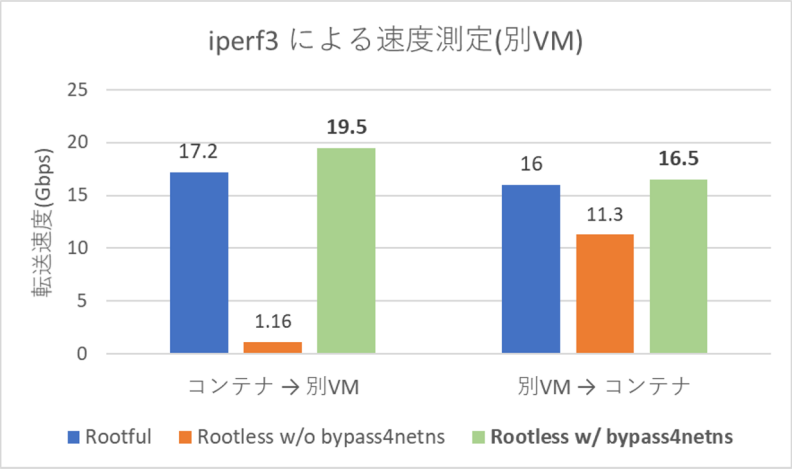
Experimental

nerdctl

NTT

https://github.com/containerd/nerdctl/blob/master/docs/rootless.md

- rootlessコンテナでもrootfulと同等以上のNW性能を得られる
- rootlessコンテナに従来存在したユーザ空間TCP/IPスタック（slirp4netns）のネットワークオーバヘッドを回避
  - SECCOMP_IOCTL_NOTIF_ADDFD で、bind(2) や connect(2) をホスト側にバイパスしている



引用元：”インターンレポート: RootlessコンテナのTCP/IP高速化”.
https://medium.com/nttlabs/accelerating-rootless-container-network-29d0e908dda4

その他

- **完全にread-onlyなmountが出来る: nerdctl run -v /mnt:/mnt:rro**
  - ○ docker run -v /mnt:/mnt:ro だと、/mnt/usb とかread-only にならない
- **nerdctl runの--netは複数指定できる (コンテナを複数ネットワークに同時接続できる)**
  - ○ DockerでもComposeでなら同じことが出来るが、docker runでは出来ない
- **FreeBSD コンテナ (jails) も少しだけ動く**
  - ○ ネットワークには未対応
- **CNIやOCIなどの標準への準拠度が高い**
  - ○ nerdctl はCNI対応、docker は未対応
  - ○ nerdctl save/nerdctl load はOCI Image SpecとDocker Image Specの両方対応、docker save/docker load はOCI Image Spec未対応

containerdに移行すると出来なくなること

NTT

- **Kubernetesノード上での docker コマンド使用**
  - ○ nerdctlを代わりに使用できる

```
nerdctl --namespace=k8s.io build -t foo:1.2.3 .
```

- **Docker REST API**
  - ○ Docker REST API に頼るアプリは動かない
  - ○ docker コマンドを直に exec するアプリは、nerdctl を exec するように書き換えるだけで動く

- **Docker Swarm**
  - ○ コンテナオーケストレーションにはKubernetesやNomadを利用できる

containerd・nerdctlへの移行方法

※Kubernetesの場合の移行方法は前述
(スライド10-13)

containerd・nerdctl への移行（linux）

https://github.com/containerd/nerdctl/releases

tarballにcontainerd、nerdctl、runc等のバイナリ一式が全て含まれている
(/usr/local などのパスに展開)

Rootless

```
$ containerd-rootless-setuptool.sh install
$ nerdctl run -d --name nginx -p 8080:80 nginx:alpine
```

Rootful

```
$ sudo systemctl enable --now containerd
$ sudo nerdctl run -d --name nginx -p 80:80 nginx:alpine
```

containerd・nerdctl への移行（macOS）

NTT

- Lima (macOS用のWSL2のようなもの)に containerd・nerdctl も付属

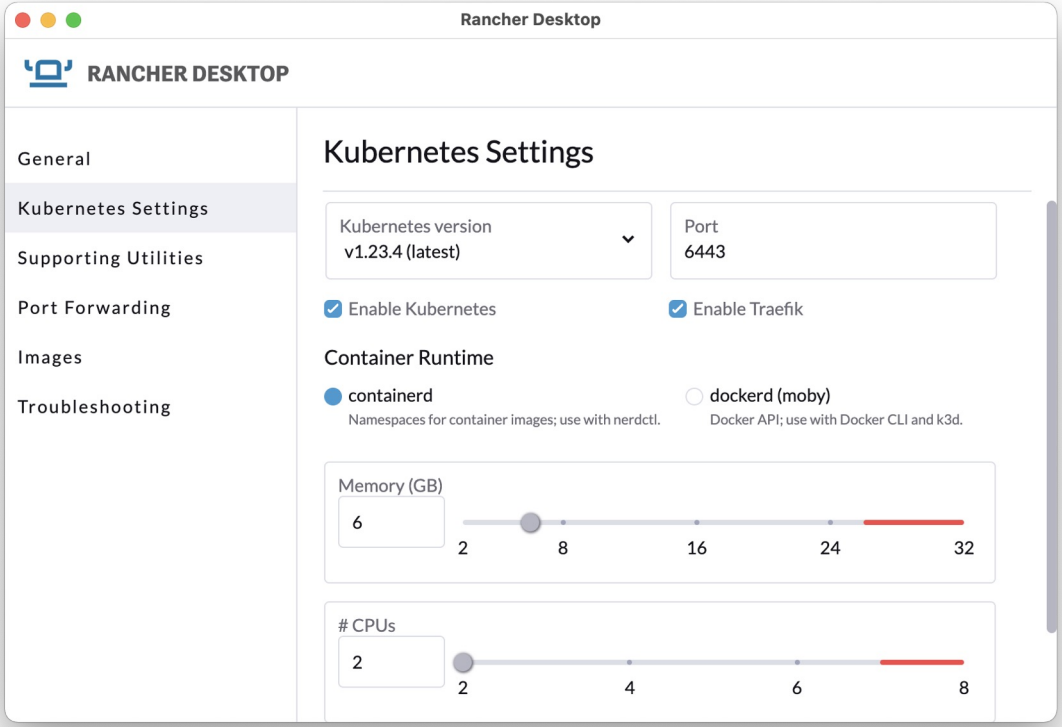
```
$ brew install lima
$ limactl start
$ lima nerdctl run …
```

- ホストのホームディレクトリが、ゲストからも見える
- ゲストのlocalhostに、ホストからもlocalhostとしてアクセスできる

https://github.com/lima-vm/lima

containerd・nerdctl への移行（macOS）

- Rancher Desktop にも containerd・nerdctl が付属
  - 内部的にはやはりLima
  - GUI付き
  - https://rancherdesktop.io/



containerd・nerdctl への移行（Windows）

- LinuxコンテナはWSL2上で動かせる
- Rancher Desktop for Windows にも containerd・nerdctl が付属
  - 内部的にはWSL2
- Windowsネイティブなコンテナにも対応 (nerdctlではexperimental)
  - https://github.com/containerd/nerdctl/issues/28

まとめ

- Docker から containerd への移行が進んでいる
- kubeadmの場合は、--cri-socket=/run/containerd/containerd.sock を渡すだけ
- 従来の Docker イメージはそのまま使える
- docker コマンドの代わりに nerdctl (contaiNERD CTL) が使える
  - ○ ただの docker コマンドの置き換えではなく、いろいろな新機能がついている (高速イメージPull、暗号化、...)
  - ○ macOS でもすぐ始められる (Windowsの場合はWSL2)

```
$ brew install lima
$ limactl start
$ lima nerdctl run …
```