# P2P Container Image Distribution on IPFS With containerd and nerdctl

FOSDEM 2022 (February 6)

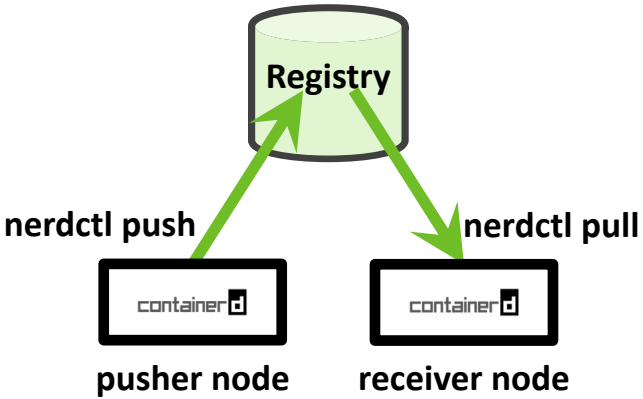
**Kohei Tokunaga, NTT Corporation**

# Summary

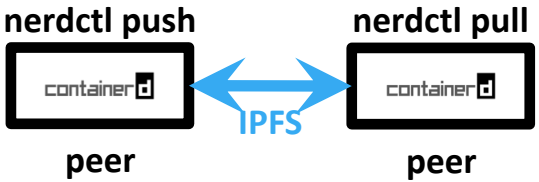
- **nerdctl** experimentally supports P2P image distribution on IPFS
  - simple UI/UX for P2P
  - allows IPFS-agnostic tools to get images from IPFS (e.g. BuildKit, Kubernetes)
  - fast image distribution from bandwidth-limited seeder

- Combination with existing OCI image distribution techniques
  - lazy pulling of eStargz
  - distributing encrypted image by OCIcrypt

**Registry-based image distribution**



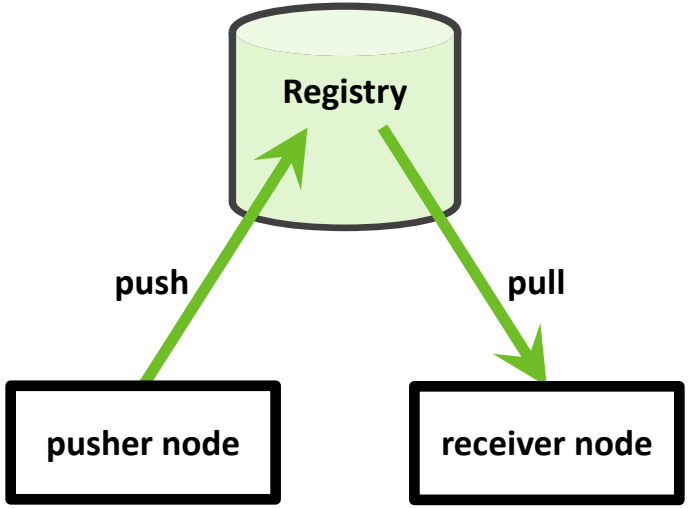

**IPFS-based image distribution**



# Problems in image distribution

- Pulling is time-consuming
  - Pulling packages accounts for 76% of container start time [Harter et al. 2016]
  - Can be slower under limited bandwidth between registry and node

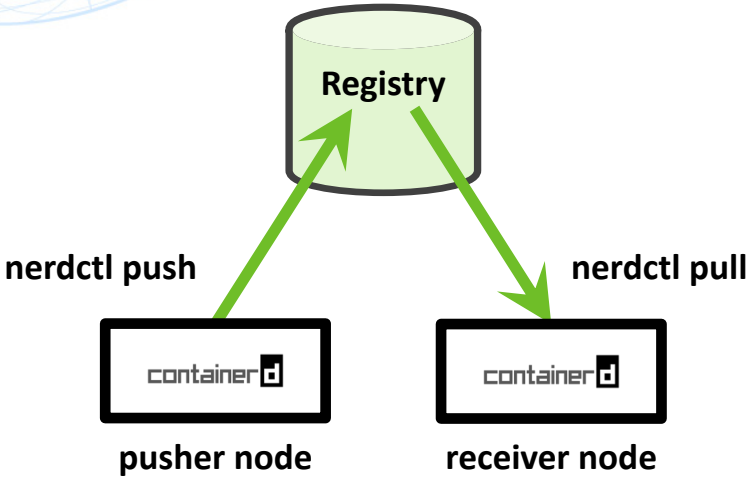
- Images can't be shared if no access to the registry (e.g., registry outage, rate limited, no access to the internet, …)



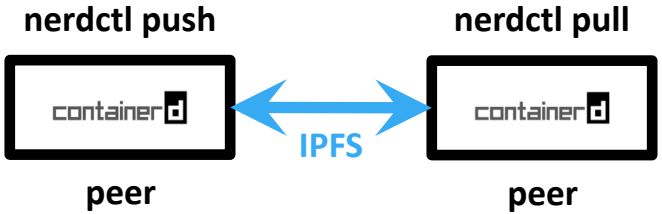
# IPFS-based P2P image distribution with nerdctl

- nerdctl CLI (>= v0.14) of containerd experimentally supports image distribution on IPFS
  - Images are shared in P2P manner without relying on the centralized registry
  - Simple UI/UX for P2P image distribution
  - Fast image distribution from bandwidth-limited seeder (discussed later)

**Registry-based image distribution**




**IPFS-based image distribution**



# nerdctl: Docker-compatible CLI of containerd

https://github.com/containerd/nerdctl

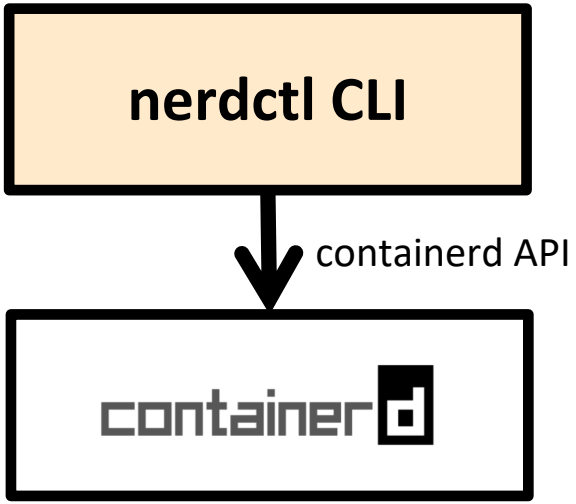
- **Has same UI/UX as Docker**

```
nerdctl run -it --rm alpine
```

```
nerdctl push ghcr.io/ktock/myalpine:latest
```

```
nerdctl build -t foo /dockerfile-dir
```

- **Supports cutting-edge features**
  - rootless
  - lazy-pulling (eStargz)
  - encrypted images (OCIcrypt)
  - P2P image distribution (IPFS)
  - container image signing and verifying (cosign)

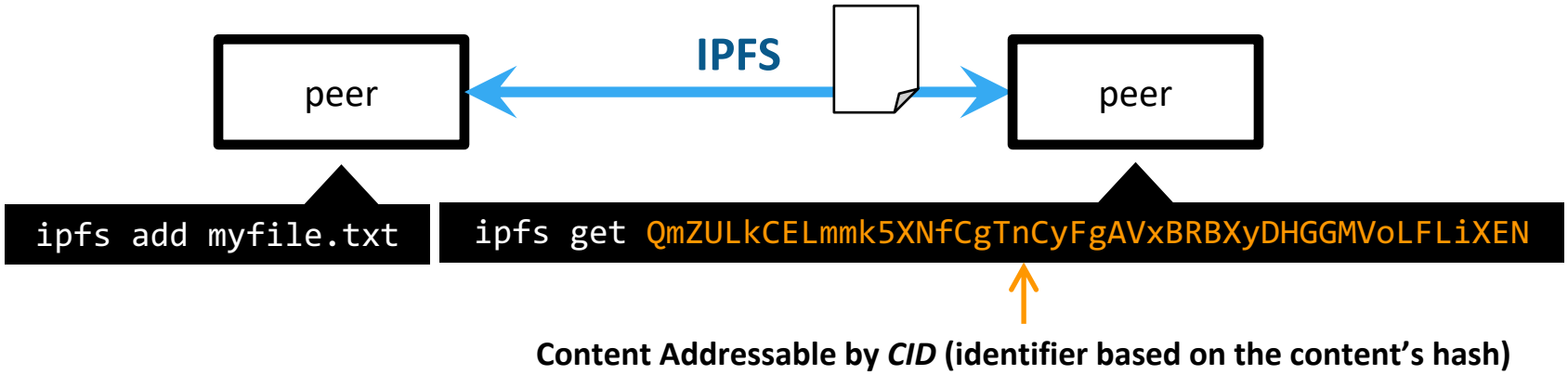



- **Adopted by lima and Rancher Desktop**
  - container management tool for desktop
  - https://medium.com/nttlabs/containerd-and-lima-39e0b64d2a59

# IPFS

https://ipfs.io

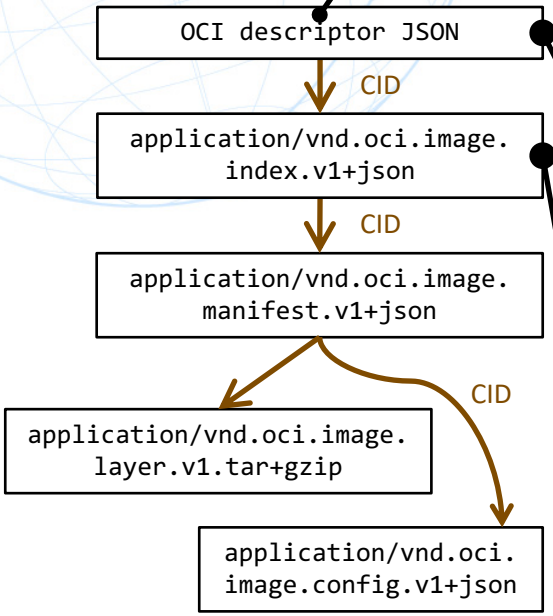
- P2P and content addressable data sharing protocol
- No central server is needed
- Content addressable by CID



# Configuration of OCI image for IPFS

- Constructing DAG by CIDs
- Image is referenced by CID of the topmost “OCI descriptor” JSON

**ipfs:// bafkreicq4dg6nkef5ju422ptedcwfz6kcvpvvhuqeykfrwq5krazf3muze**

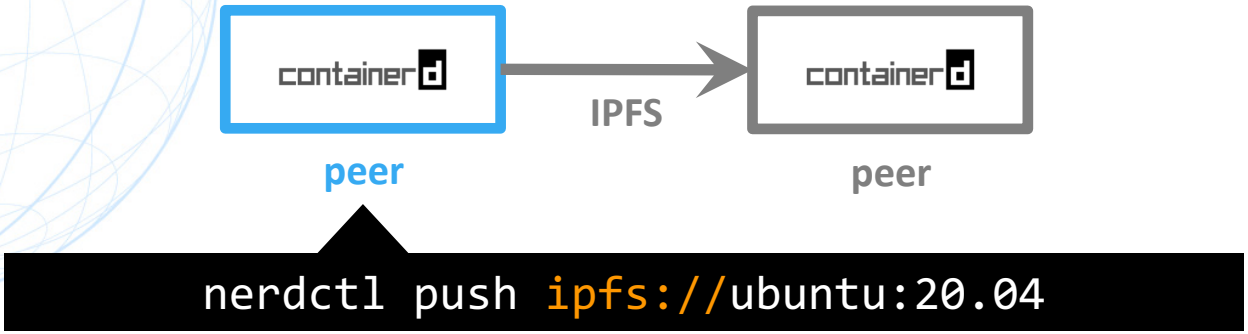



```
{
  "mediaType": "application/vnd.oci.image.index.v1+json",
  "digest": "sha256:28bfa1fc6d491d3bee91bab451cab29c747e72917efacb0adc4e73faffe1f51c",
  "size": 313,
  "urls": [
    "ipfs://bafkreibix6q7y3kjdu565en2wri4vmu4or7hfel67lfqvxcoop5p7ypvdq"
  ]
}
```

```
{
  “schemaVersion”: 2,
  “manifests”: [
    {
      “mediaType”: “application/vnd.oci.image.manifest.v1+json”,
      “digest”: “sha256:f6eed19a2880f1000be1d46fb5d114d094a59e350f9d025580f7297c8d9527d5”,
      “size”: 506,
      “urls”: [
        “ipfs://bafkreihw53izukea6eaaxyoun625cfgqsssz4niptubflahxff6i3fjh2u”
      ],
      . . .
```

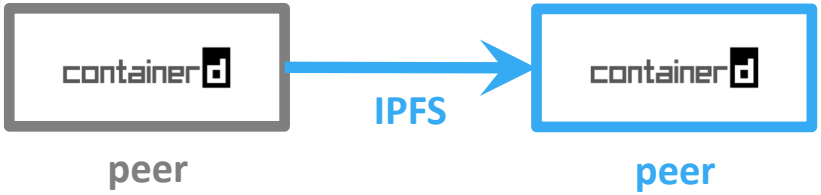
Each item in OCI image supports arbitrary URLs as the data source
→ we store CID (formed as IPFS URL)

# Adding an image to IPFS




- nerdctl supports **`ipfs://`** prefix for an arbitrary image name
- nerdctl pushes the image to IPFS instead of registry
  - Automatically configures the OCI image for IPFS (see previous slide)
- The image is distributed on IPFS in a p2p manner without registry

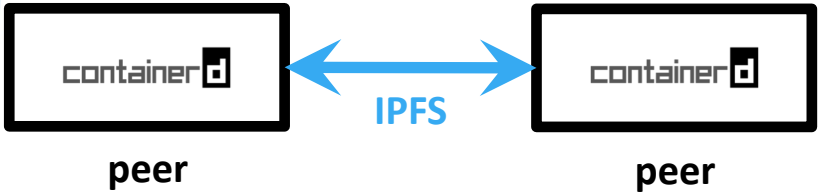
# Pulling an image from IPFS




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

- **`ipfs://CID`** references an image on IPFS
- nerdctl gets the image from IPFS instead of the registry
- The image needs to be configured for IPFS
  - “`nerdctl push ipfs://`” automatically does this

# Building image based on images on IPFS




Dockerfile

```
FROM localhost:5050/ipfs/bafkreicq4dg6nkef5ju422ptedcwfz6kcvpvvhuqeykfrwq5krazf3muze
RUN echo hello > /hello
```

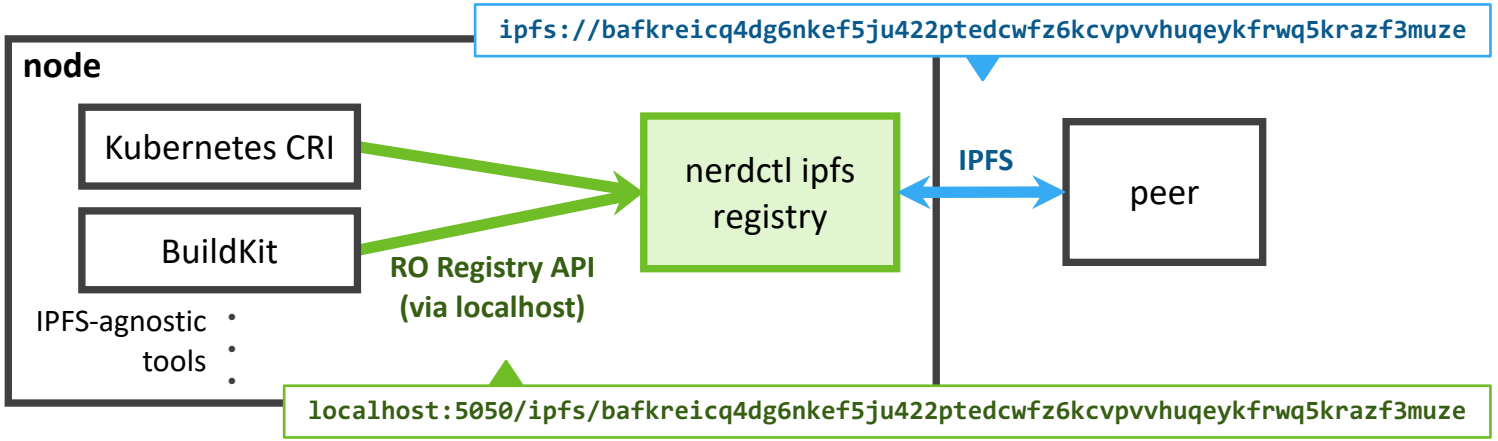
- **`localhost:5050/ipfs/CID`** references an image on IPFS
  - Dockerfile should support "`ipfs://CID`" image reference in the future
- Base image is acquired from IPFS
- The result image can also be pushed to IPFS using "`nerdctl push ipfs://`"

# IPFS-based image distribution for IPFS-agnostic tools

NTT

`nerdctl ipfs registry` **subcommand**

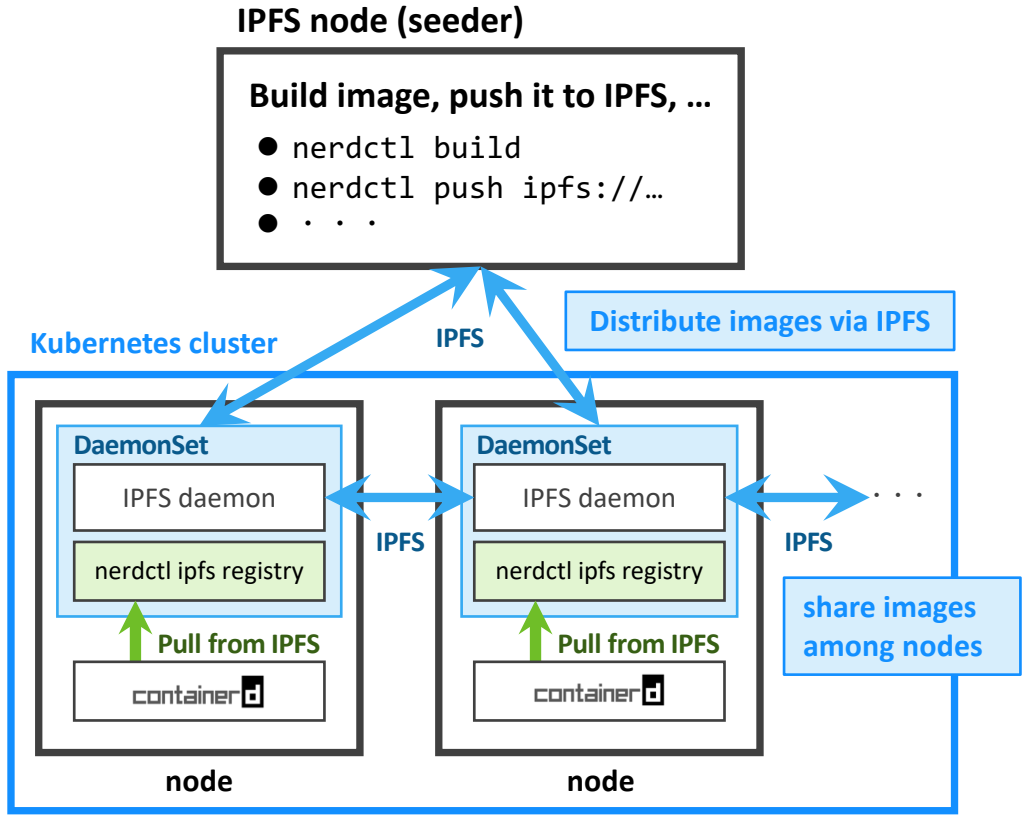
- Provides a read-only localhost registry backed by IPFS
  - image is accessible via **`localhost:5050/ipfs/CID`**
- IPFS-agnostic tools (e.g. Kubernetes) can pull images from IPFS
  - `nerdctl build` (backed by BuildKit) uses this functionality



# Example: node-to-node image sharing on Kubernetes

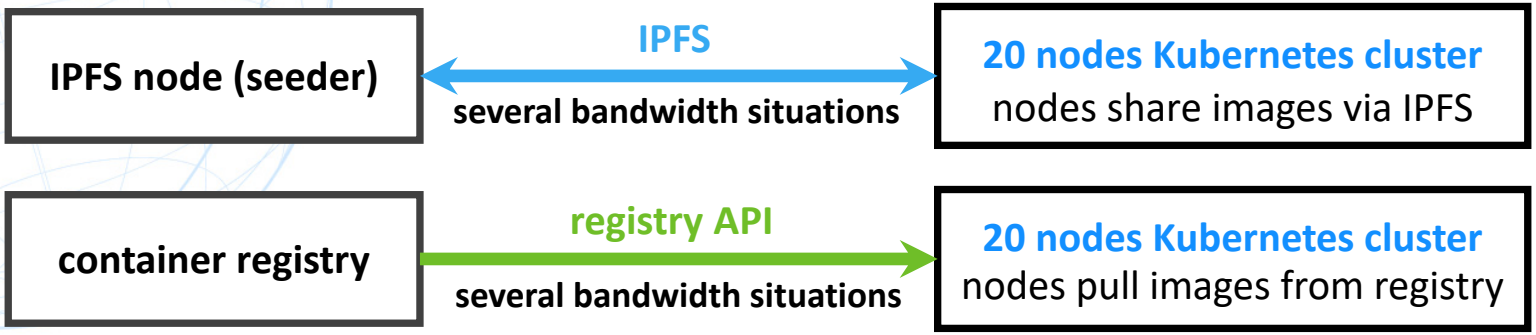
(WIP) https://github.com/containerd/nerdctl/pull/678

- “`nerdctl ipfs registry`” can be used for node-to-node image sharing
- In the future, Kubernetes should support “`ipfs://CID`” image reference
- Example configuration: running ipfs daemon as DaemonSet on each node



# Image distribution latency

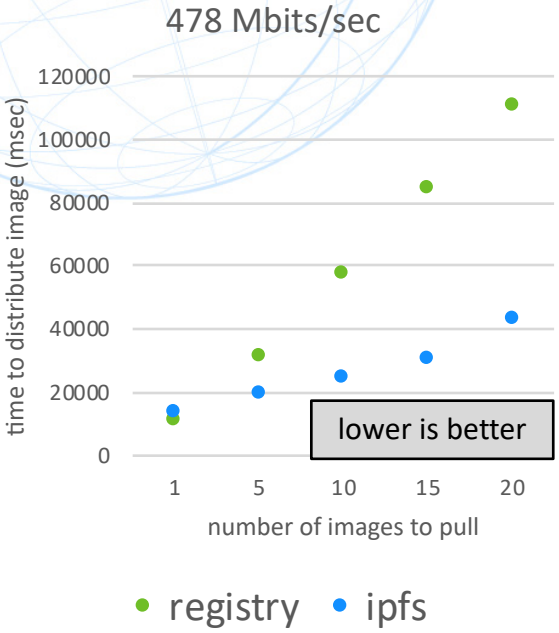
Measured time to take to distribute images under several bandwidth situations

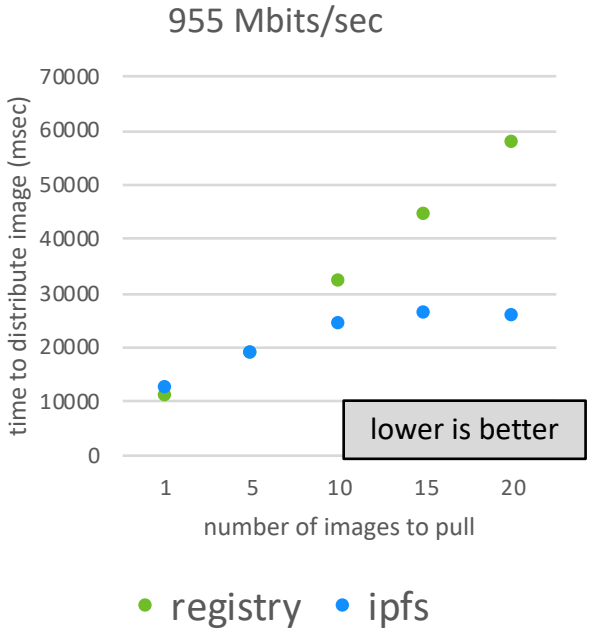


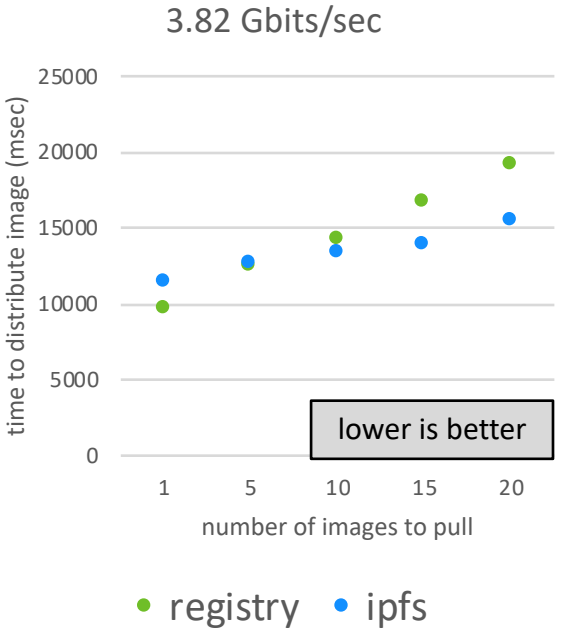

- GKE v1.21.5-gke.1302 (20 nodes)
  - instance: e2-standard-8 (asia-northeast1-a)
  - OS: ubuntu_containerd (upgraded containerd to v1.5.8 manually)
- private seeder/registry (1 node)
  - instance: e2-standard-8 (asia-northeast1-a)
  - OS: Ubuntu 20.04
- image: ghcr.io/stargz-containers/jenkins:2.60.3-org (726.4 MiB)
- Measured the worst time to take for pull with configuring bandwidth using linux tc
- commit: https://github.com/containerd/nerdctl/commit/3b5ed0df186d05d986b9cdb7c47773f29febed29
  - ipfs v0.11.0 (k8s nodes), ipfs v0.10.0 (seeder), nerdctl bb682bc
- benchmarking script: https://github.com/ktock/stargz-snapshotter/tree/nerdctl-ipfs-registry-kubernetes-benchmark/script/nerdctl-ipfs-registry-kubernetes-benchmark

# Image distribution latency

- On lower bandwidth with many images, IPFS distributes images faster than registry
- On higher bandwidth or with small number of images, IPFS can be slower than registry
  - will work on further investigation and mitigation



# Combination with existing image distribution techniques

# Lazy pulling: eStargz

https://github.com/containerd/stargz-snapshotter

- **Lazy pulling:** Starting up containers without waiting for the pull completion
  - Each chunk/file in the image is downloaded on-demand
- **eStargz**: OCI-compatible image format for lazy pulling with prefetch support
  - Can be lazily pulled from standard registries
- **Stargz Snapshotter**: Plugin of containerd for enabling lazy pulling

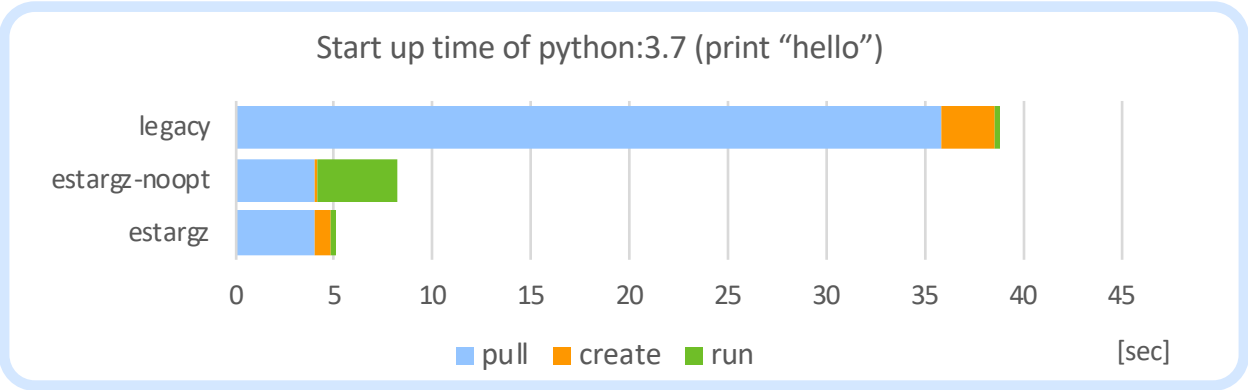
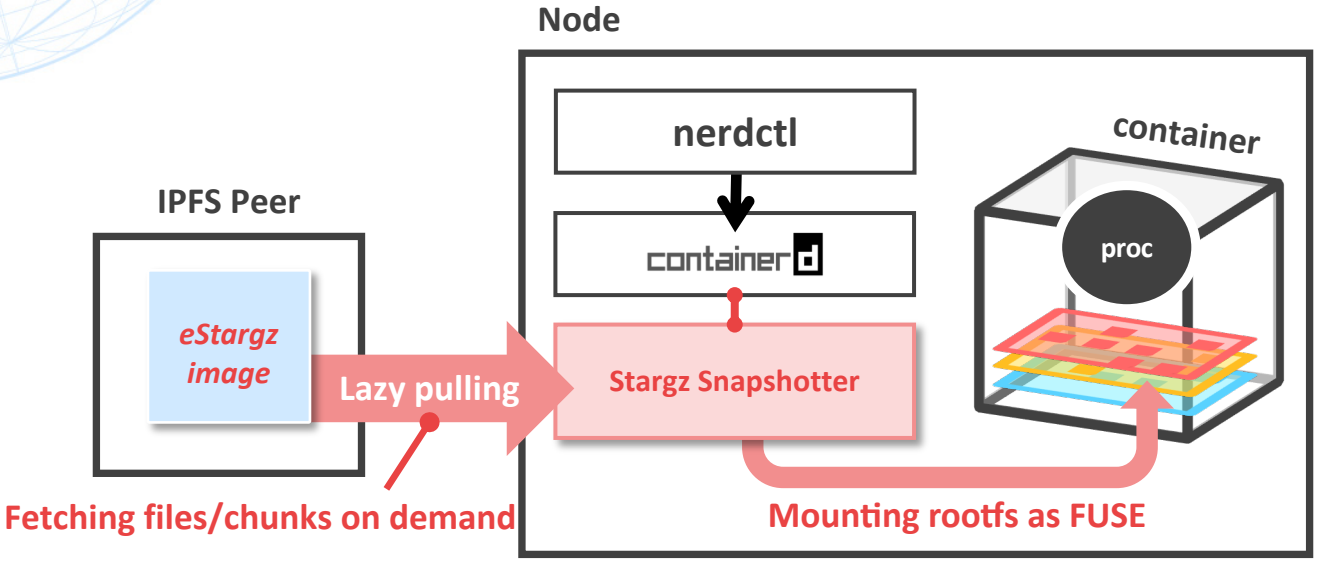



Figure from “Faster Container Image Distribution on a Variety of Tools with Lazy Pulling - Kohei Tokunaga & Tao Peng. KubeCon+CloudNativeCon North America 2021. https://sched.co/lV2a “

# Lazy pulling (eStargz) on IPFS

https://github.com/containerd/stargz-snapshotter

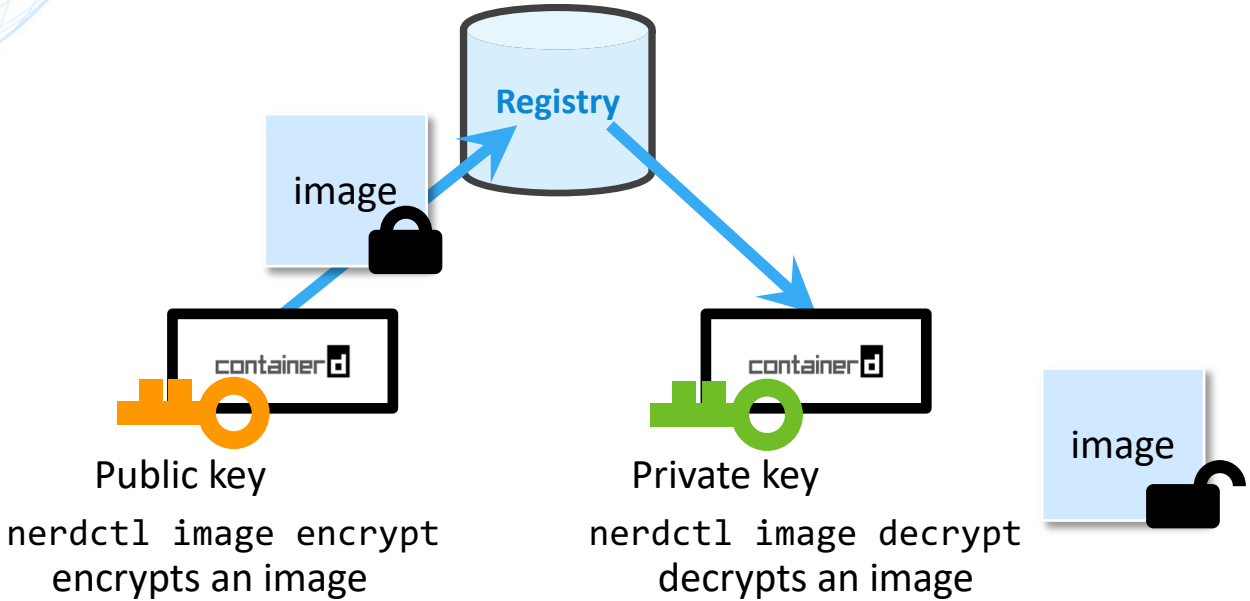
- eStargz can be stored to IPFS
- Stargz Snapshotter supports lazy pulling of eStargz from IPFS
  - mounts eStargz image from IPFS to container's rootfs
- Chunks are fetched lazily thus hopefully faster cold-start



# Image encryption: OCIcrypt

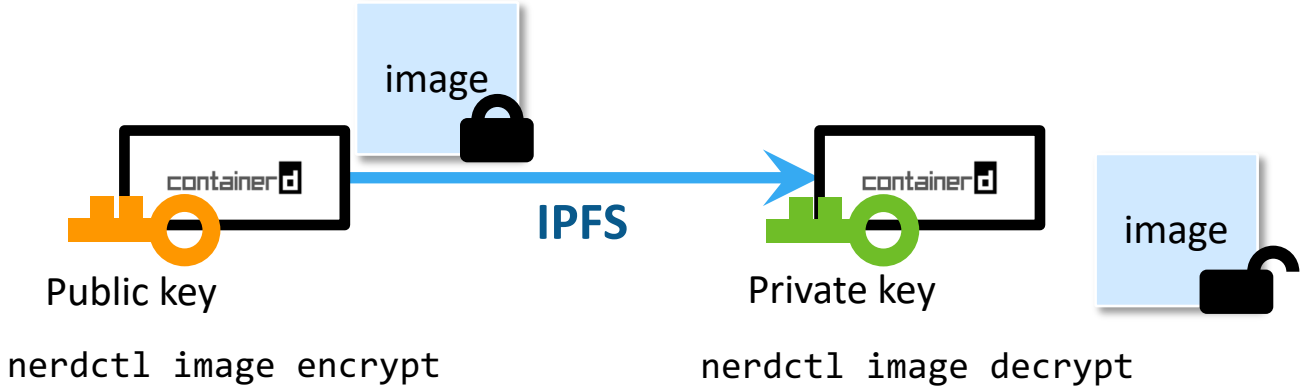
https://github.com/containerd/imgcrypt

- nerdctl supports encryption/decryption of image layers with key pair
- OCIcrypt (imgcrypt plugin for containerd) is used



# Image encryption (OCICrypt) on IPFS

- Encrypted image can be pushed to IPFS
- Configuration of the image for IPFS (`urls` field) is done by "`nerdctl push ipfs://`"



# Future works

- Performance improvements
  - Especially on high bandwidth environment with small number of images
  - Pulling hangs when the searching image isn't found on IPFS
- Enabling "`ipfs://CID`" reference on a various tools (e.g. BuildKit, Kubernetes, …)
- CID reproducibility
  - "`nerdctl push ipfs://`" will produce different CIDs for different configurations of IPFS
    - e.g. different chunk size
- Higher availability of data
  - "pinning services" can be used maybe
- …

# Related projects about image distribution on IPFS

- **ipcs**: https://github.com/hinshun/ipcs
  - Proposed by Edgar Lee (Netflix)
  - containerd content store plugin backed by IPFS
  - Focuses on content deduplication but incompatible to OCI image

- **ipdr**: https://github.com/ipdr/ipdr
  - Proposed by Miguel Mota
  - Docker registry backed by IPFS
  - No native integration with runtime (requires a dedicated CLI)
  - Lazy pulling unsupported

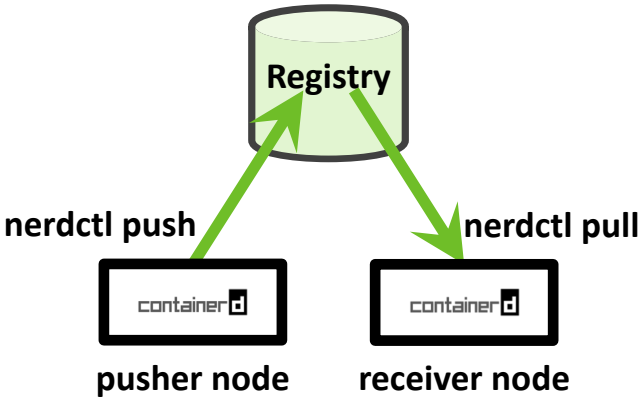
- **EdgePier[1]**
  - Proposed by Soeren Becker, et al.
  - Integrated ipdr (mentioned in the above) with Kubernetes
  - Fast image distribution under bandwidth-restricted environment
  - No OSS implementation

[1] Soeren Becker, et al."EdgePier: P2P-based Container Image Distribution in Edge Computing Environments". IEEE International Performance Computing and Communications Conference 2021

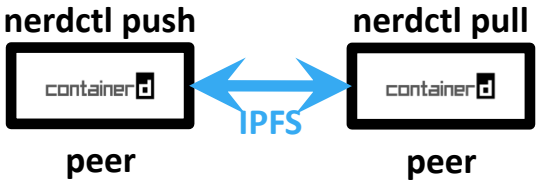
# Summary

- **nerdctl** experimentally supports P2P image distribution on IPFS
  - simple UI/UX for P2P
  - allows IPFS-agnostic tools to get images from IPFS (e.g. BuildKit, Kubernetes)
  - fast image distribution from bandwidth-limited seeder
- Combination with existing OCI image distribution techniques
  - lazy pulling of eStargz, image encryption by OCIcrypt

**Registry-based image distribution**




**IPFS-based image distribution**




**Thanks to Akihiro Suda (NTT) for the discussion!**