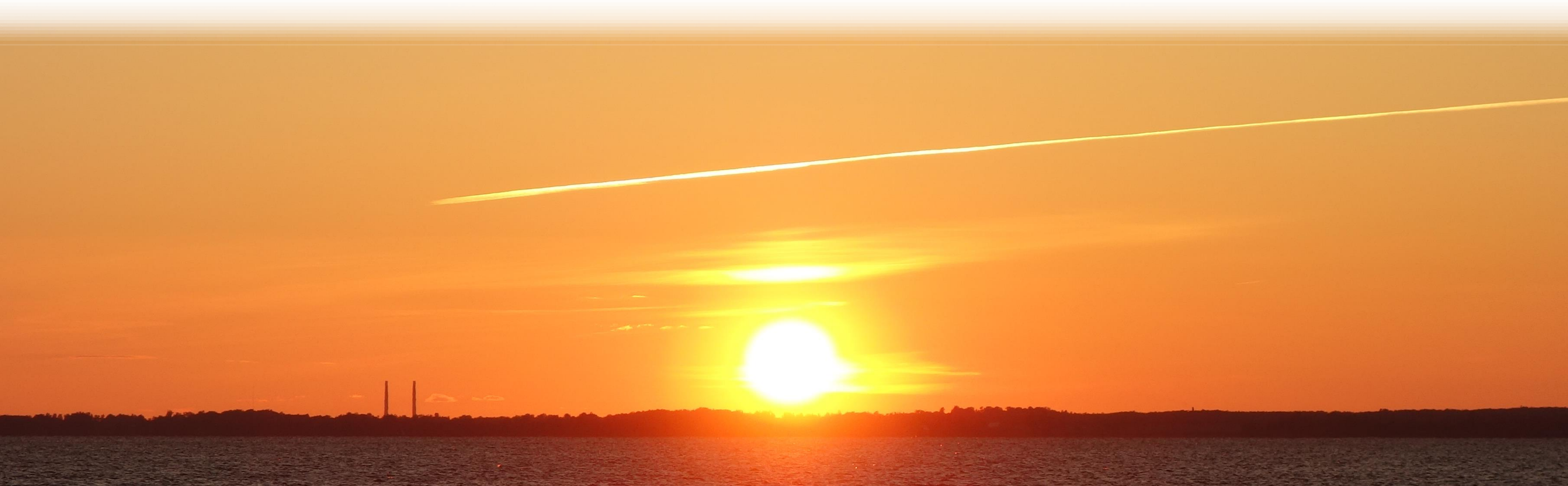
Starlight: Fast Container Provisioning on the Edge and over the WAN

Jun Lin Chen, Daniyal Liaqat, **Moshe Gabel**, Eyal de Lara

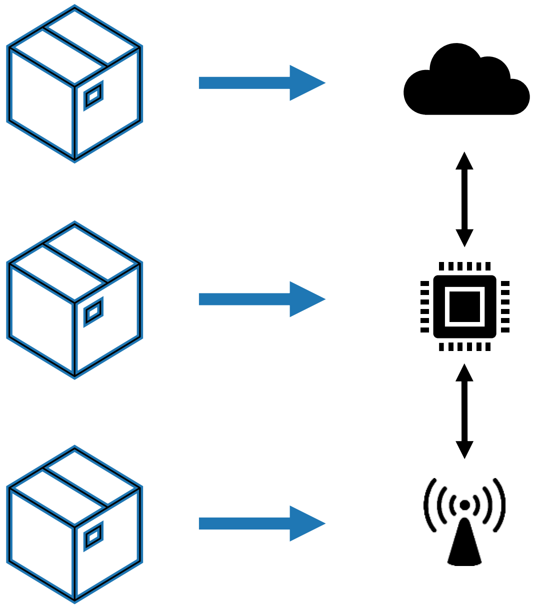
UNIVERSITY OF TORONTO



Container Provisioning

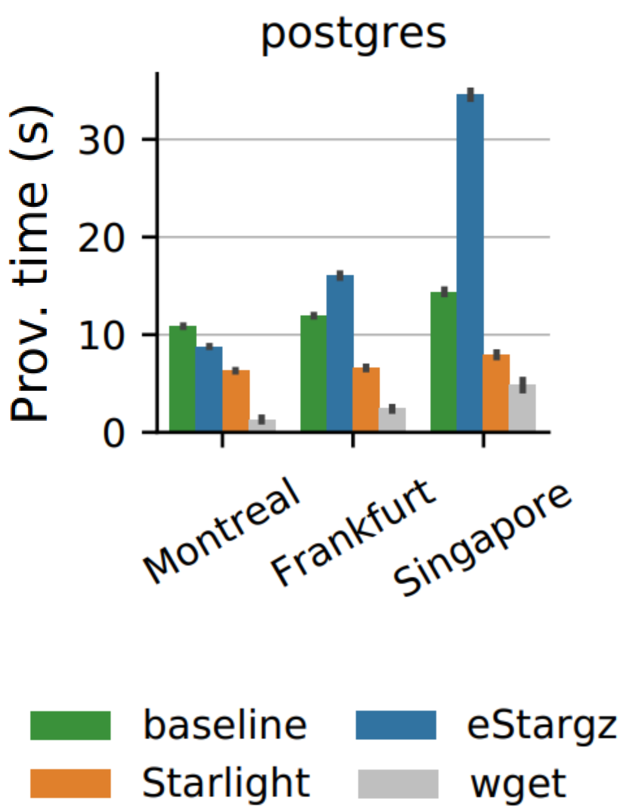
- De-facto standard approach for packaging and deploying in cloud
  - Standardized
  - Lightweight
  - Easy to develop and deploy
- Increasingly used outside cloud
  - WAN
  - Mobile
  - Edge

"the edge"



Starlight Contributions

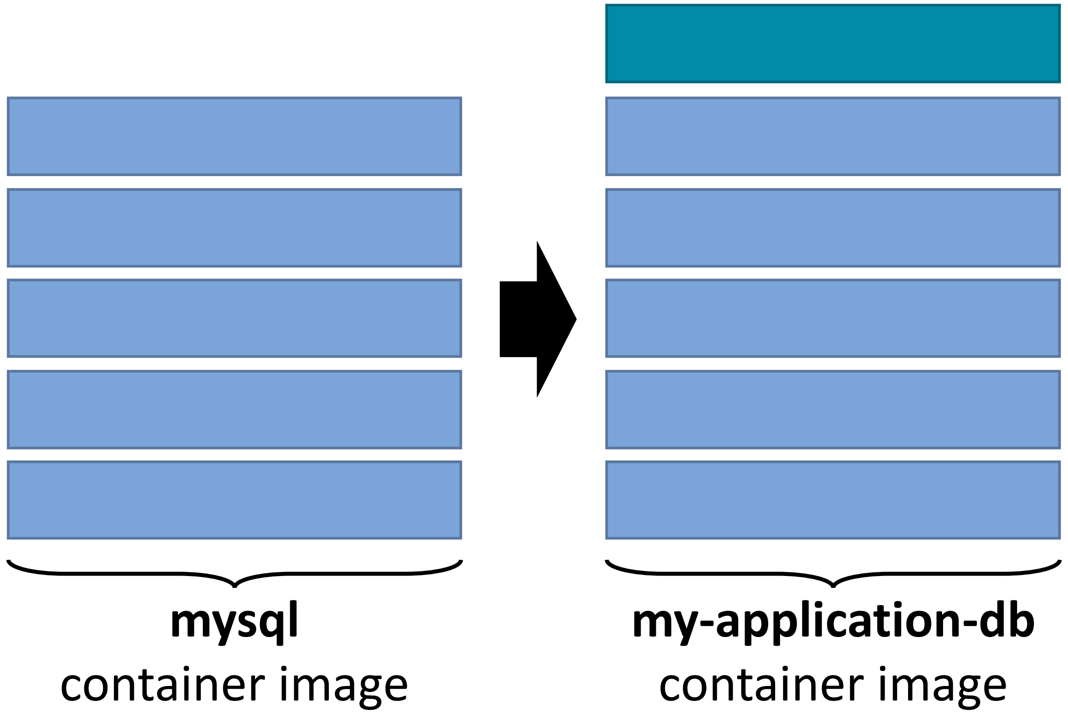
- Container provisioning slow outside datacenter.
- State-of-the-art optimizations make it worse!
- Root cause: design decisions from cloud.
- Starlight: accelerator for container provisioning
  - ✓ x3 faster, even in cloud
  - ✓ Backwards compatible with existing containers, tools, registries, standards.
  - ✓ Practically no overhead
  - ✓ Open source



Let’s expand on that…

What are Containers?

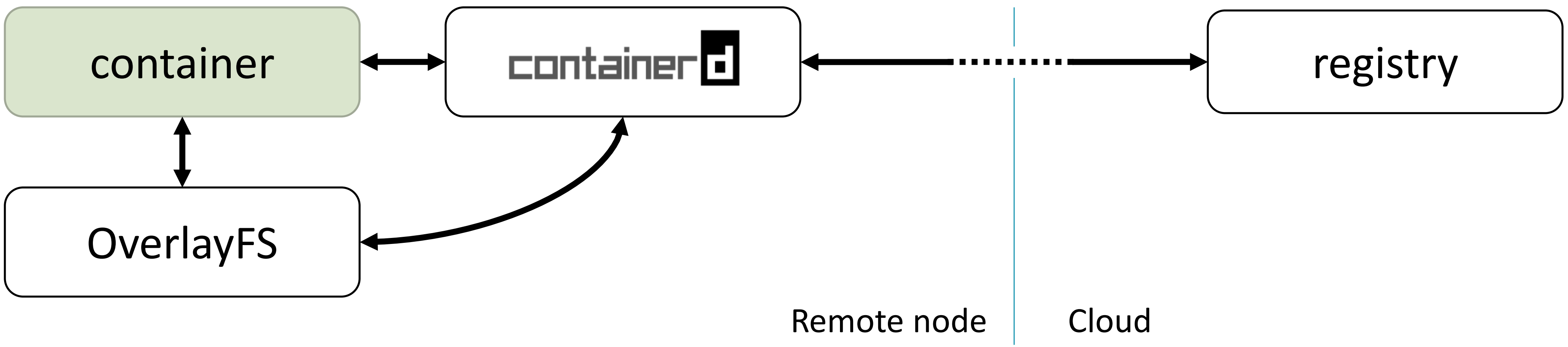
- *Container* = isolated processes
  - Filesystem, resources
- Container *image* = stack of layers
  - Filesystem is union of layers.
- Easy to develop and package:
  - Start with existing container…
  - …add new layer on top.



Deploying Containers on a Node

► Standard 3-phase process for deployment:

1. PULL: get compressed layers from registry (container DB)
2. CREATE: decompress contents, create mount points.
3. START: mount the filesystem and start.

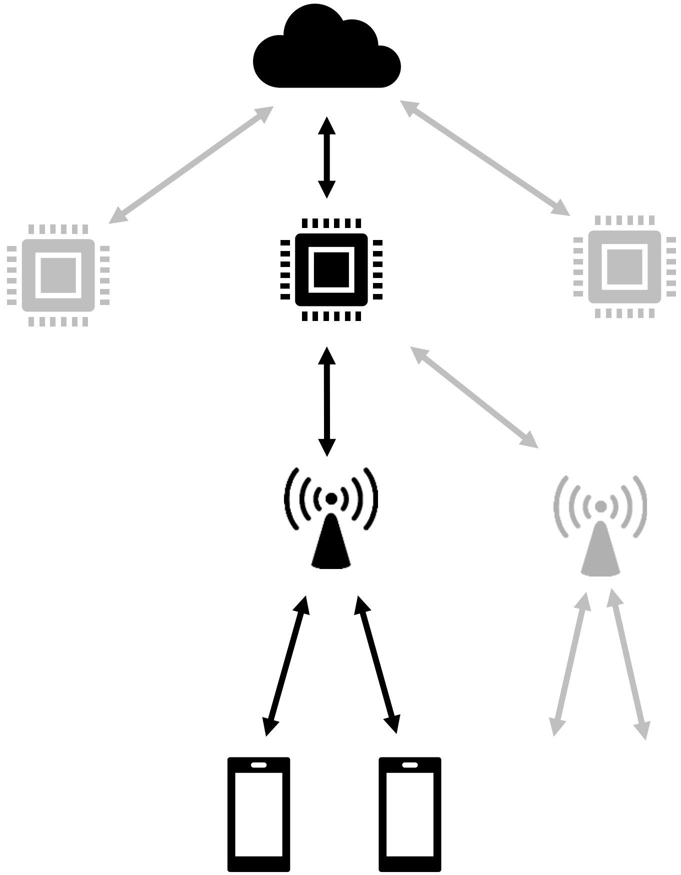


Fast Container Provisioning

- Containers-as-a-Service
  - Amazon ECS, Azure Container Instances…
- Function-as-a-Service
  - FaaSNet [Wang et al., ATC’21]
- Security and software updates
  - Log4j
- User mobility
  - [Tiwari et al., HotMobile’19]

Edge Challenges

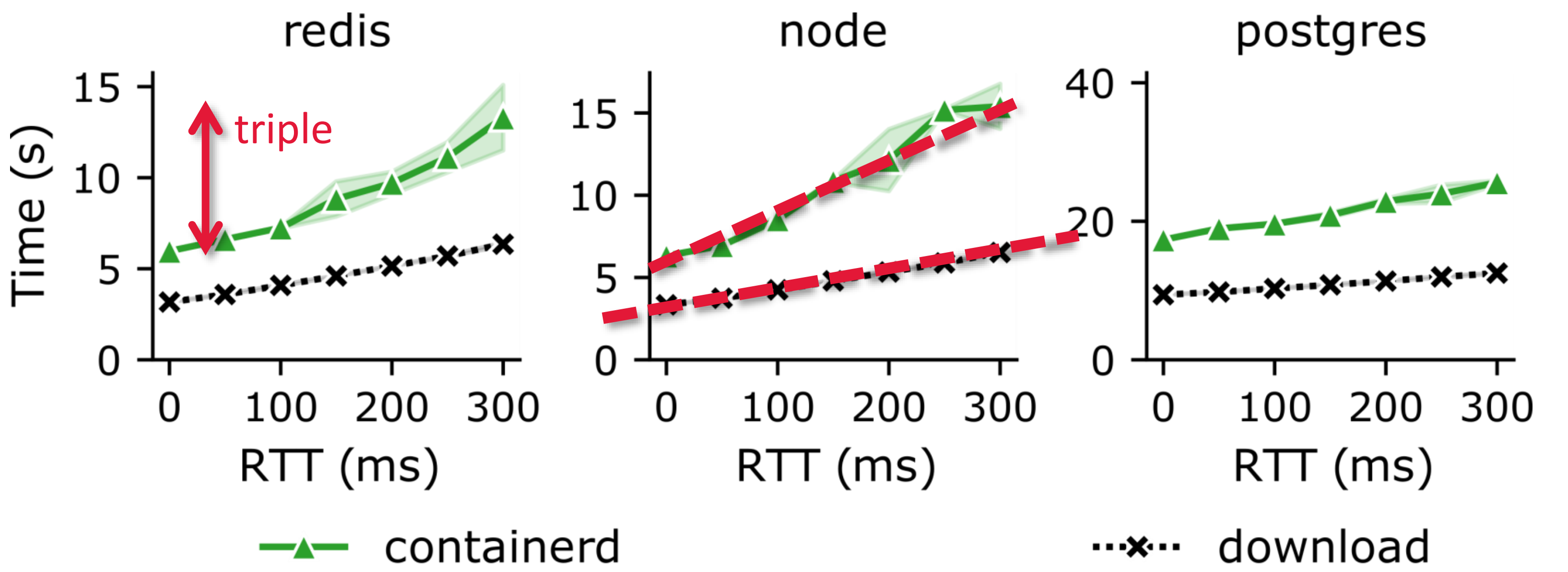
- ▶ High latency, low bandwidth links
  - → **long downloads**
- ▶ Limited edge resources
  - → **no local registry/cache**
  - → **aggressive repurposing**
- ▶ User mobility
  - → **frequent reconfiguration**



Containers on the Edge

- ► Deploy containers on the edge, measure provisioning time.
  - ➢ **containerd prov. time** = download + decompress + start + ready for work
  - ➢ **download**: just download.

lower is better



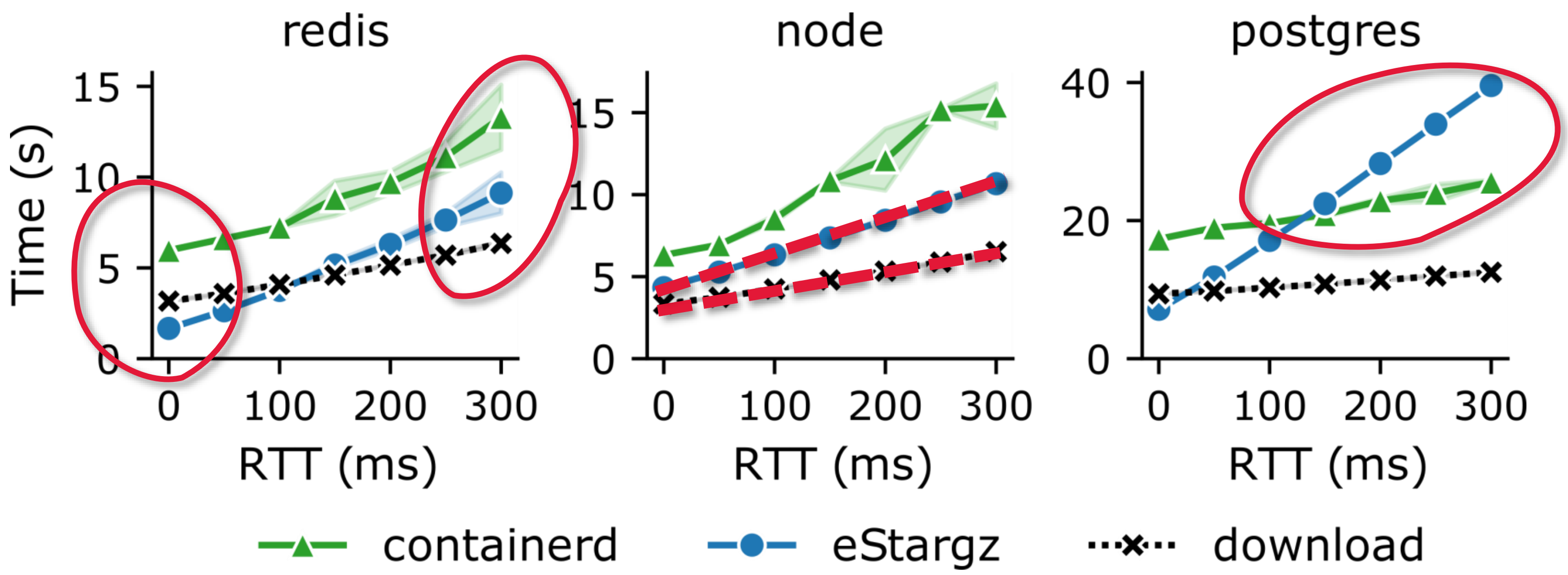
prov time **triples** when moving to edge

Prov time grows at faster rate than download times.

UNIVERSITY OF TORONTO

Containers on the Edge

- ▶ 60%—99% files not needed during startup
  Harter, et al. Slacker: Fast distribution with lazy docker containers, FAST 16
- ▶ **eStargz**: state-of-the art, start containers early, download on-demand



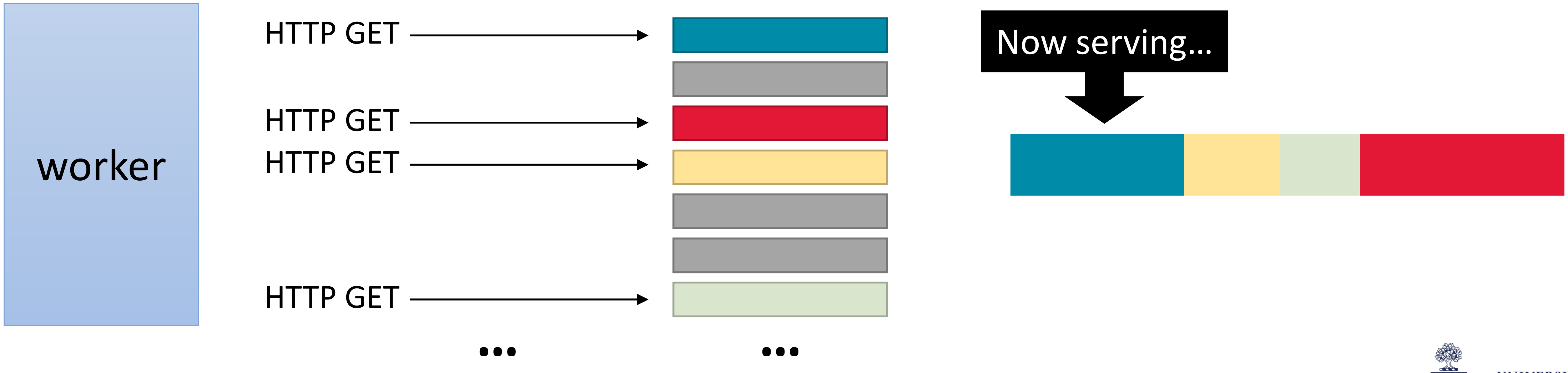
Fast in cloud, but slow on edge.

Scales badly with RTT

Can be slower than containerd!

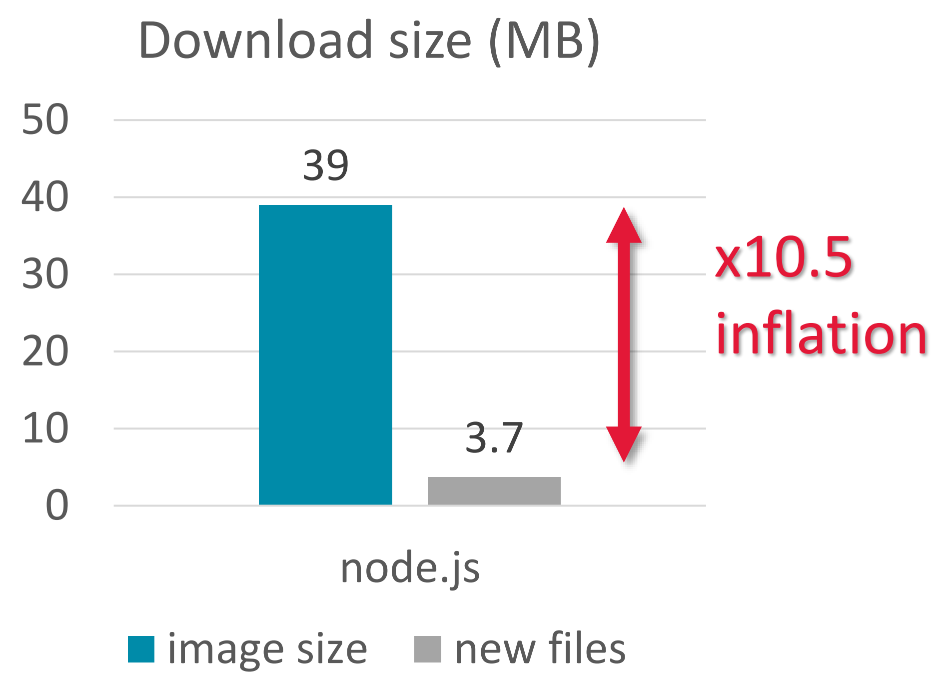
Why Slow?

- **Pull-based protocol**:
  - Worker retrieves only layers it needs
  - Multiple long HTTP requests → many roundtrips, queuing
  - On-demand file requests makes this **worse**!



Why Slow?

- ▶ **Layer-based structure**:
  - ➢ Metadata stored per-layer
    → extra roundtrips
  - ➢ Cross-layer file duplication
    → inflates downloads
  - ➢ Docker Hub study:
    99.4% of files are duplicated
    Zhao et al, CLUSTER '19



Starlight

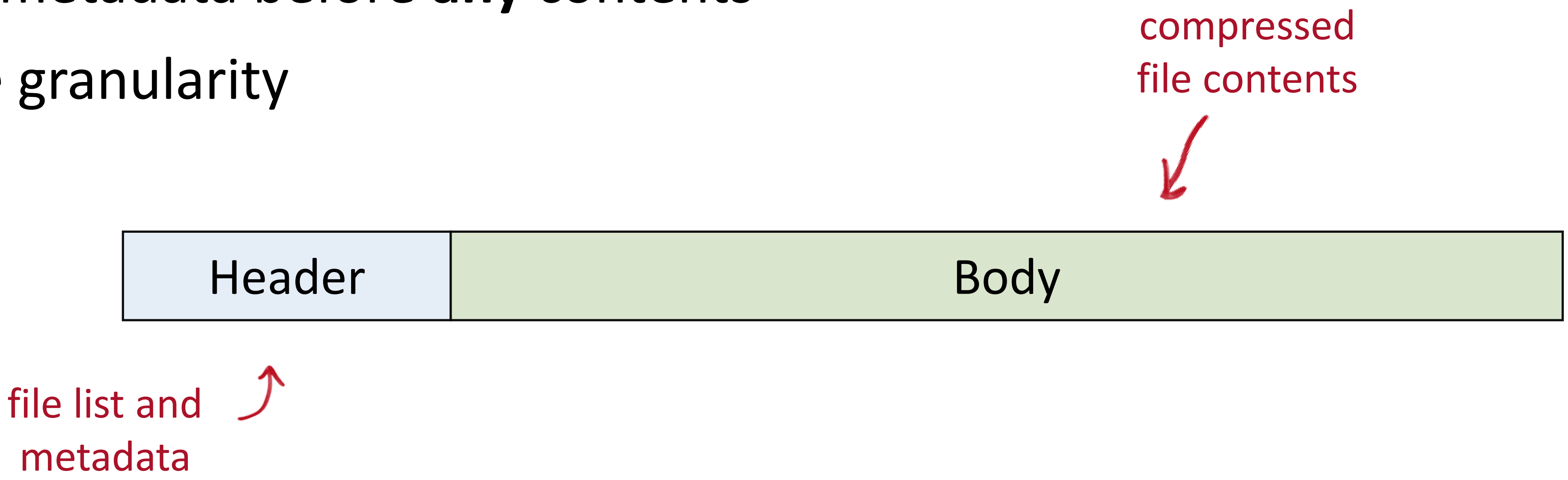
- ► Piecemeal approaches won't fix core design.
  - ➢ …and we want to be backwards compatible.
- ► Must rethink deployment pipeline as a whole.

- ► So that's what we did with Starlight!

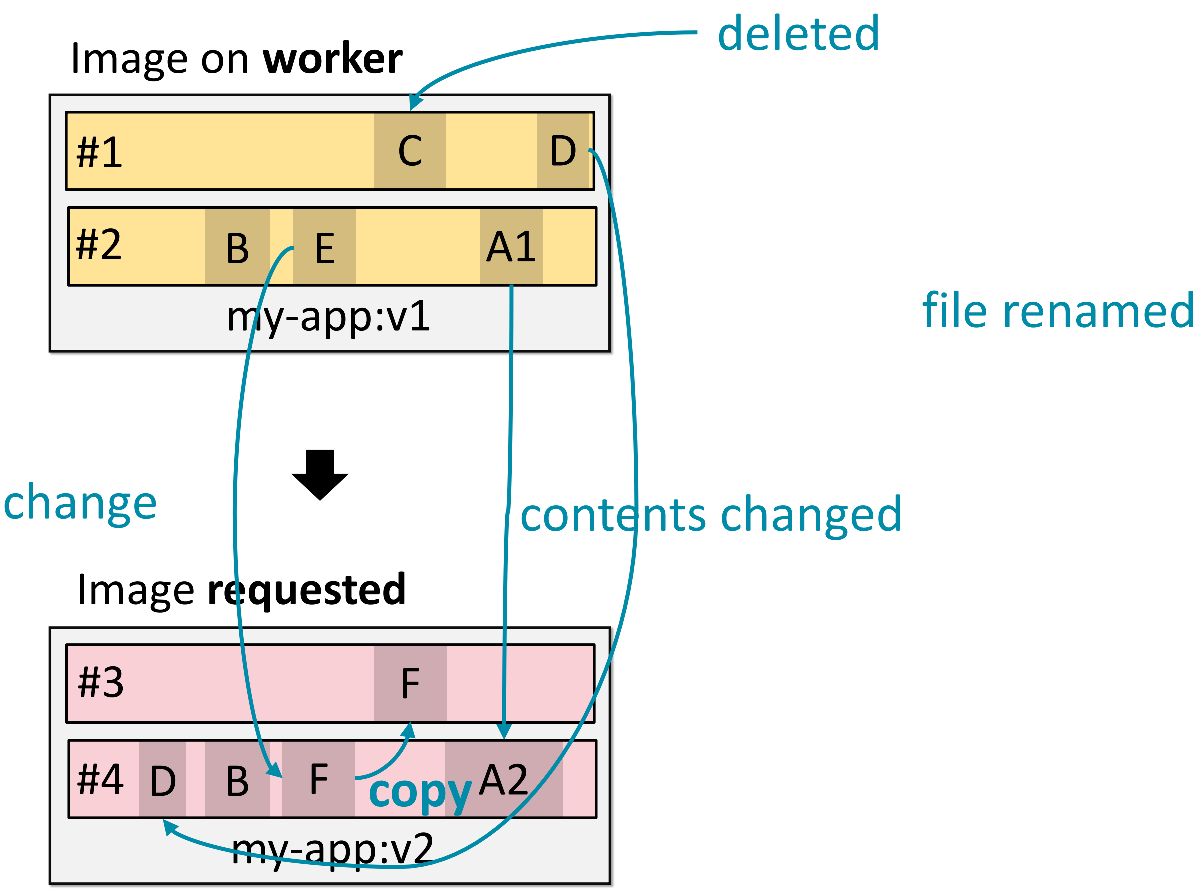
1. Designed new **worker-cloud protocol** (push-based, file-granularity).
2. Implemented components to support it.

Design of Delta Bundle Protocol

- ▶ Push-based: single request, no roundtrips
- ▶ Only send what worker needs
- ▶ **All** metadata before **any** contents
- ▶ File granularity

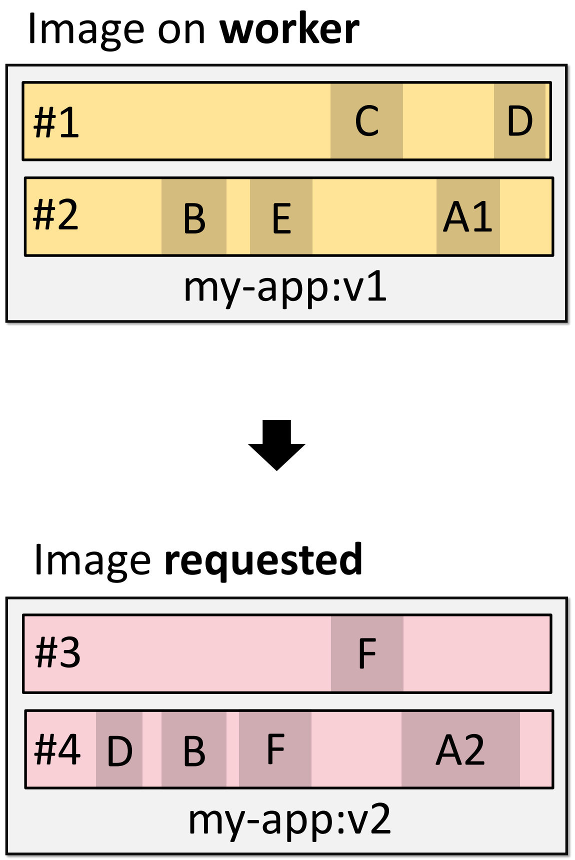


Delta Bundle Structure

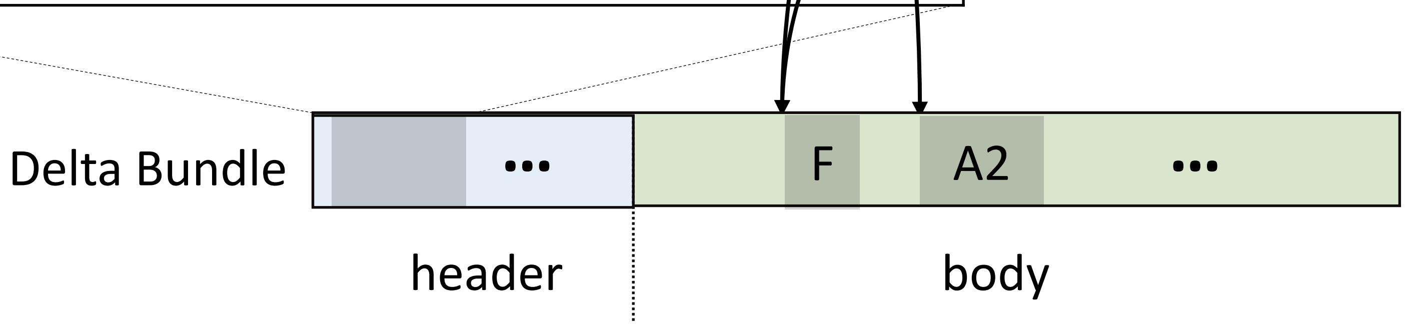


UNIVERSITY OF TORONTO

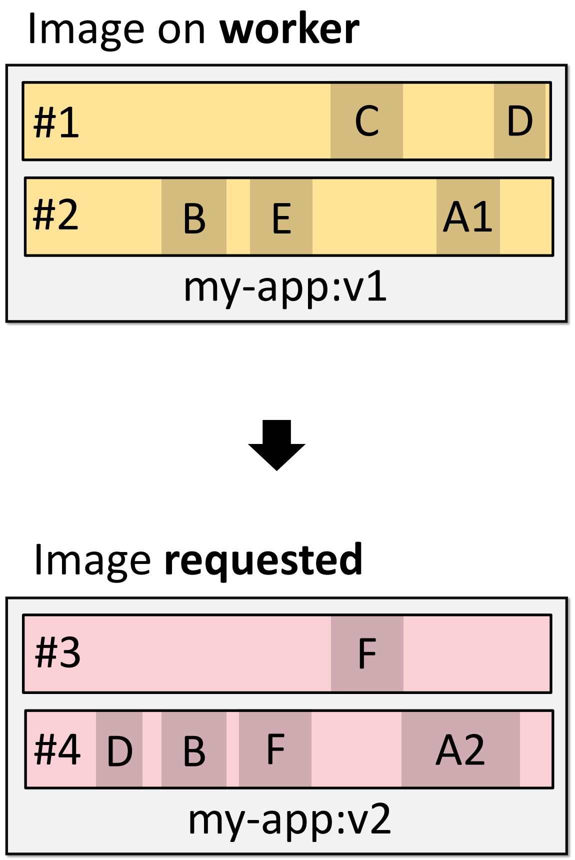
Delta Bundle Structure



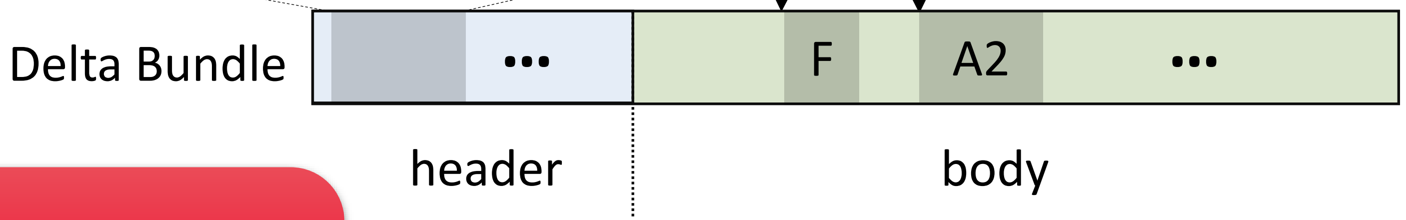
file	hash	attributes	…	source	offset
etc/my.cnf	B	-r--------		#2 , #4	
etc/ssh		drwxr-xr-x		#2 , #4	
etc/sshd_config	A2	-rw-r--r--		#4	
etc/sshd_banner	F	-rw-r--r--		#3	
bin/tar	D	-rw-r--r--		#1 , #4	
root/test.txt	F	-rw-r--r--		#3	
		...			



Delta Bundle Structure

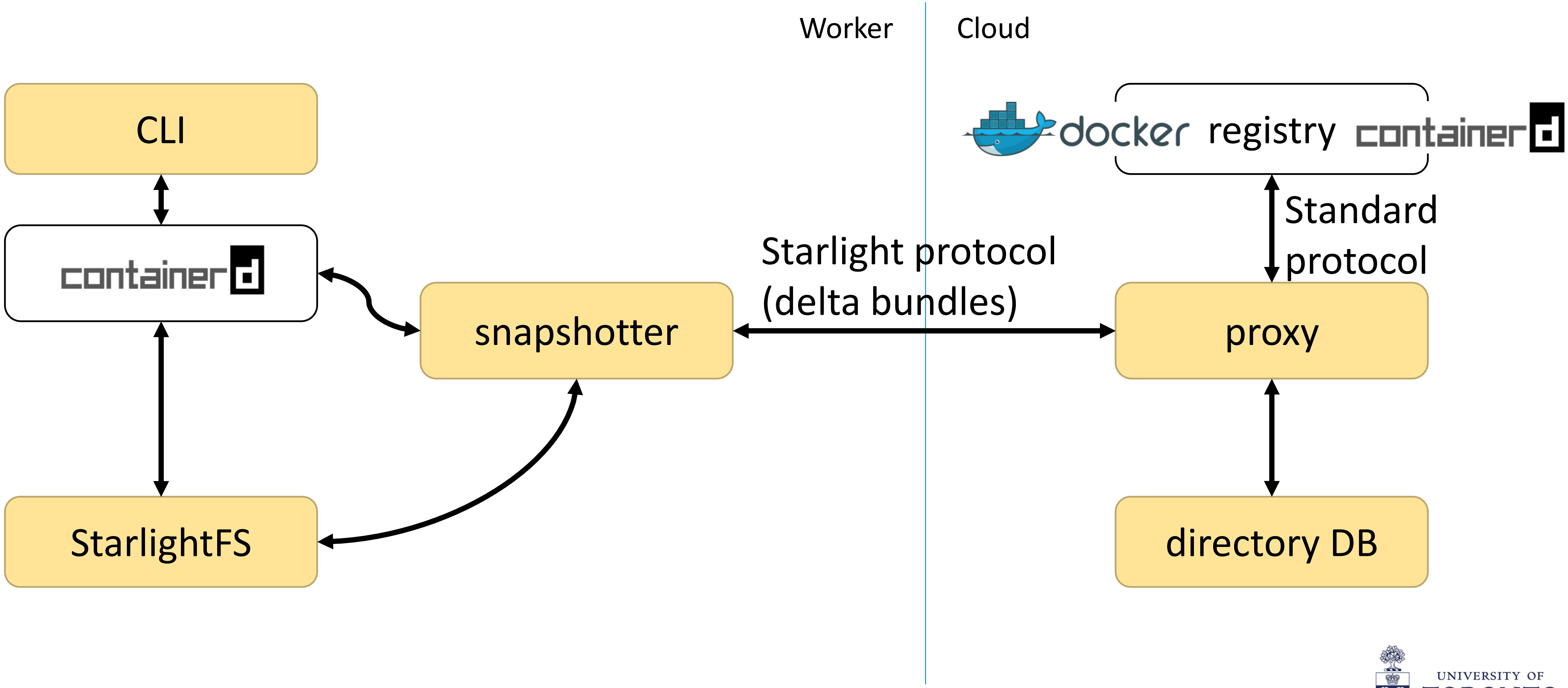


file	hash	attributes	…	source	offset
etc/my.cnf	**B**	-r--------		#2 , #4	
etc/ssh		drwxr-xr-x		#2 , #4	
etc/sshd_config	**A2**	-rw-r--r--		#4	
etc/sshd_banner	**F**	-rw-r--r--		#3	
bin/tar	**D**	-rw-r--r--		#1 , #4	
root/test.txt	**F**	-rw-r--r--		#3	
...					

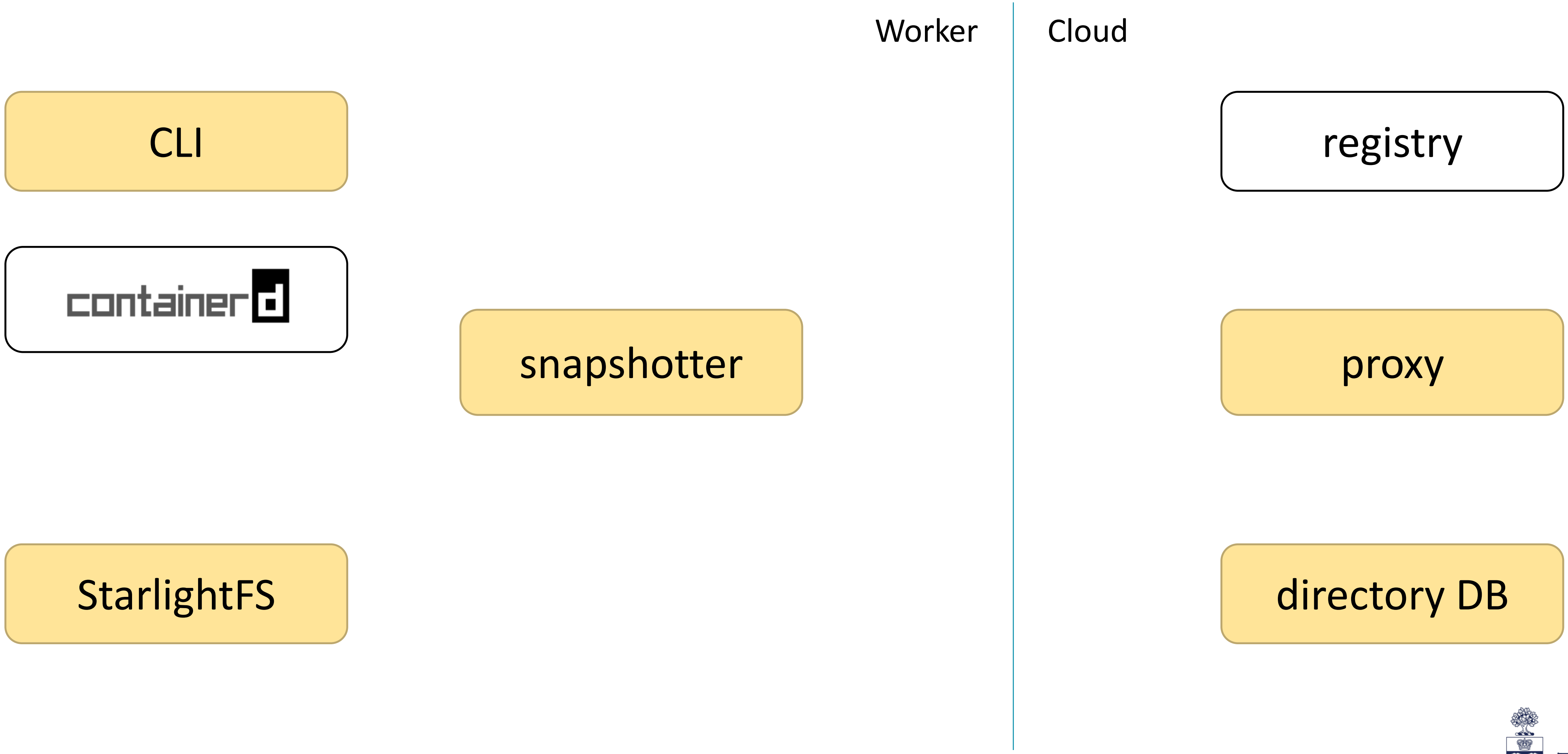


- ✓ Metadata in front
- ✓ Only new contents
- ✓ No duplication

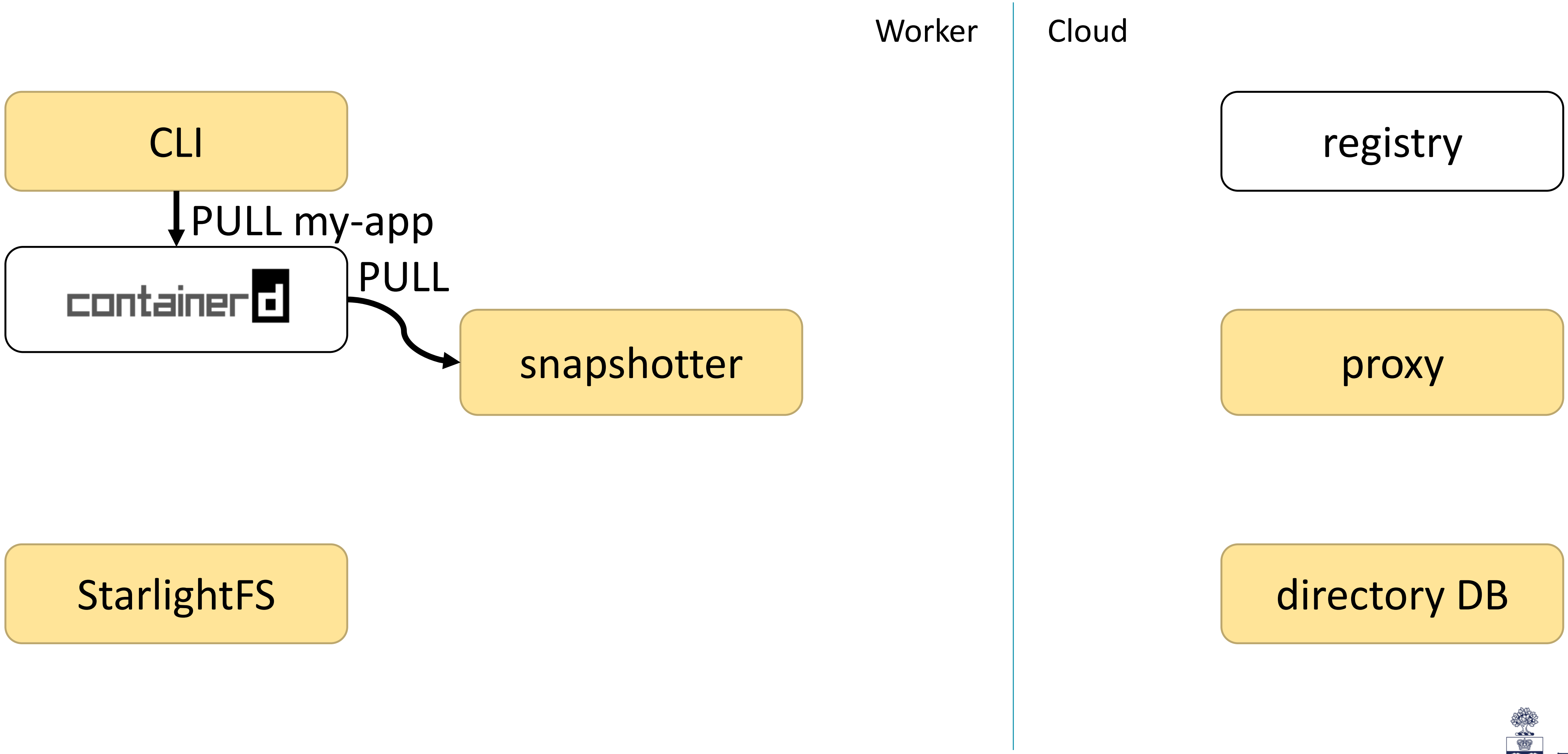
Starlight Architecture



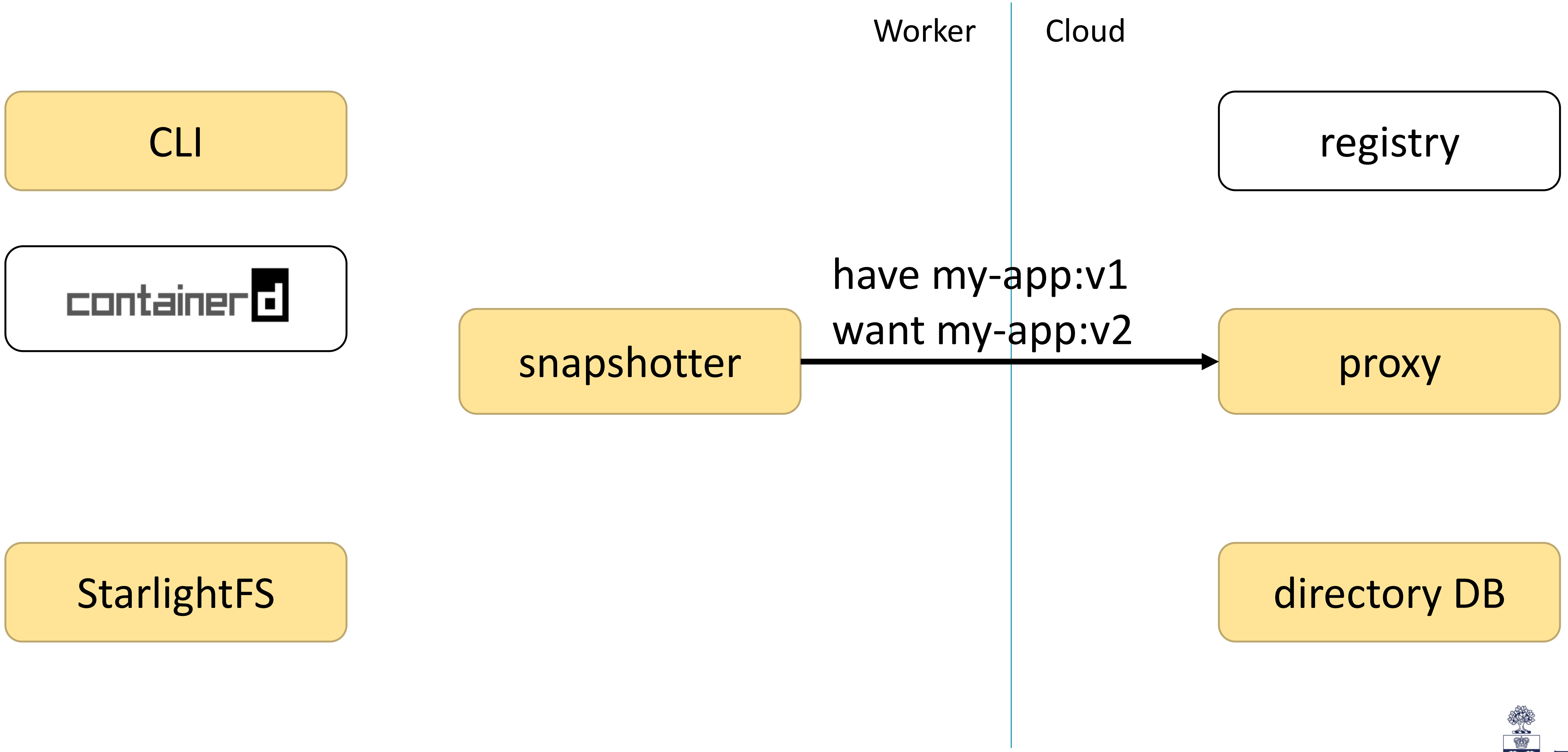
Starlight Operation



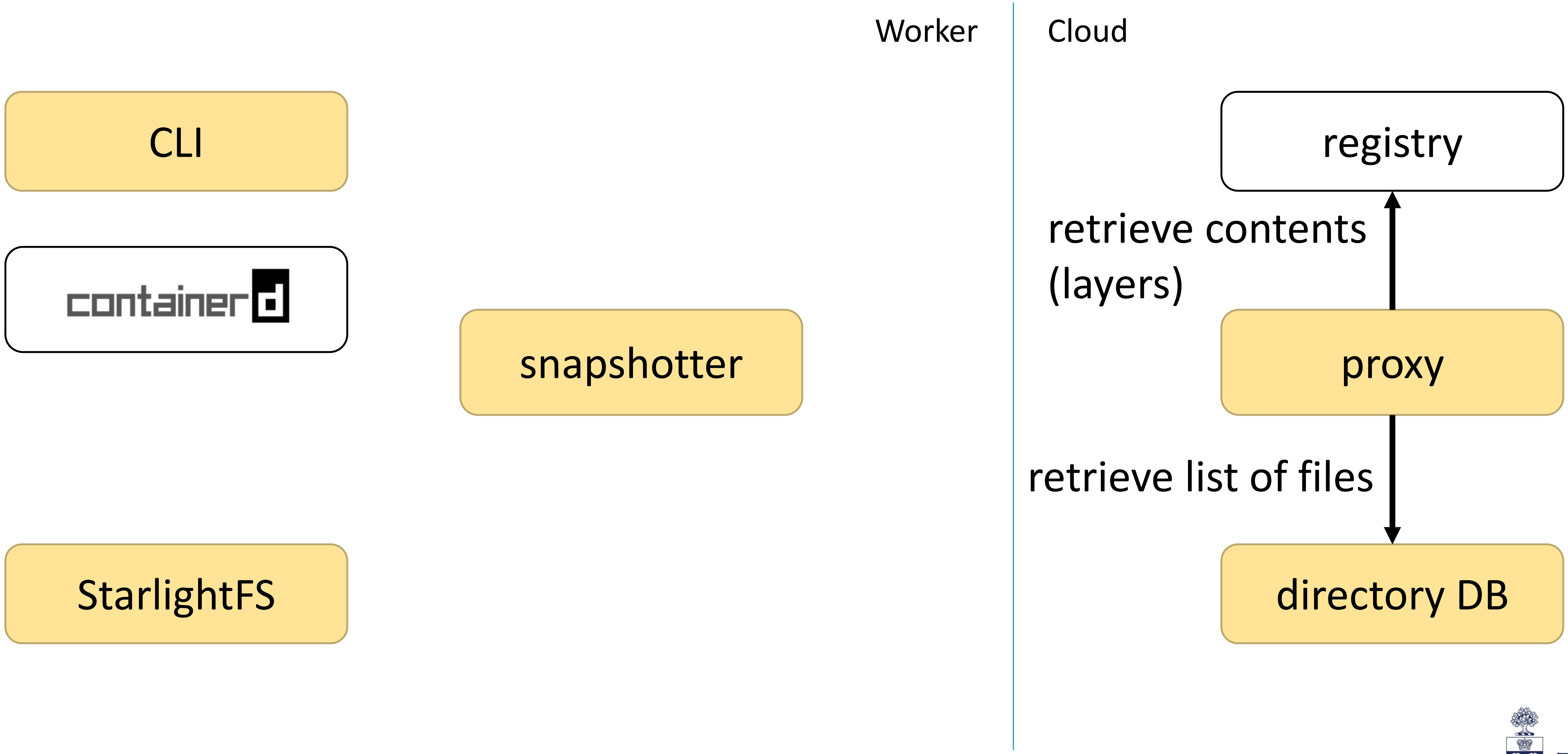
Starlight Operation



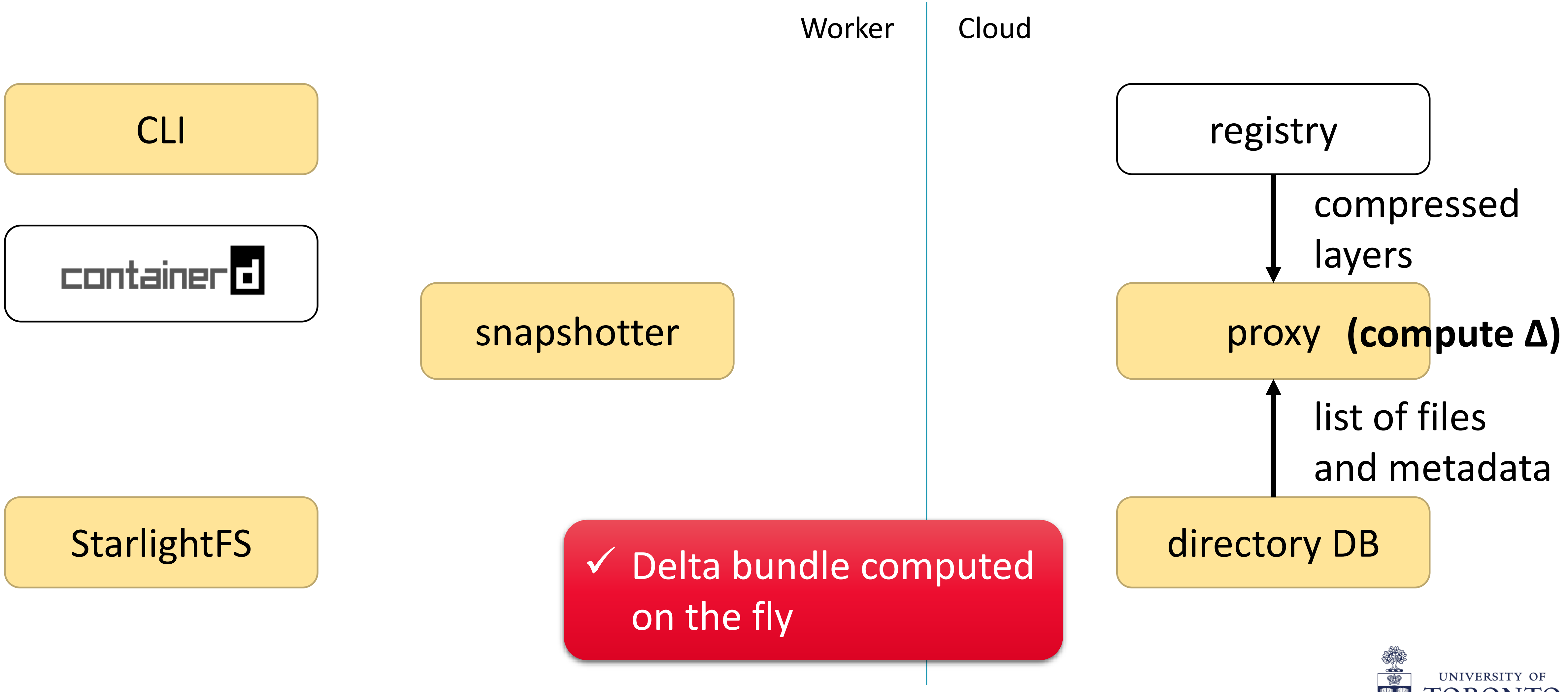
Starlight Operation



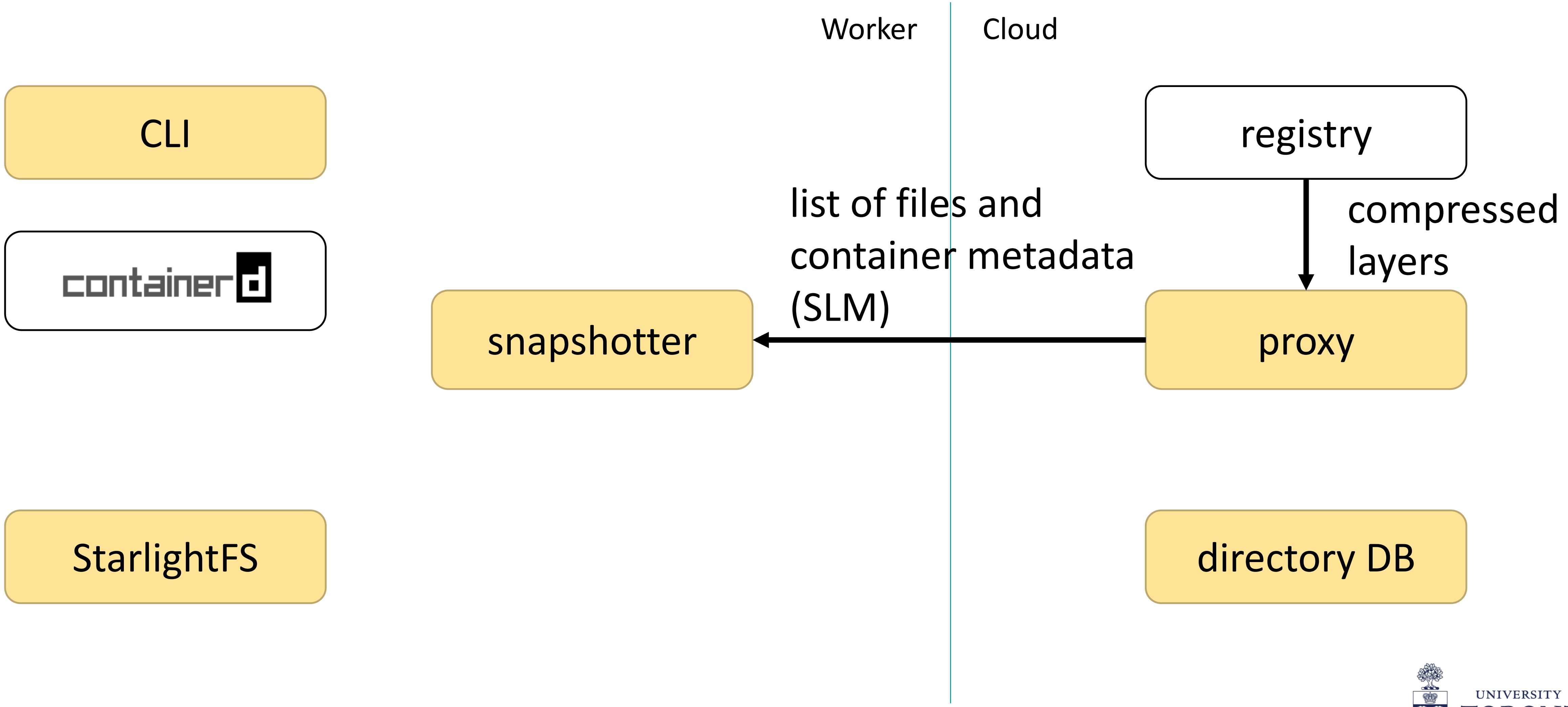
Starlight Operation



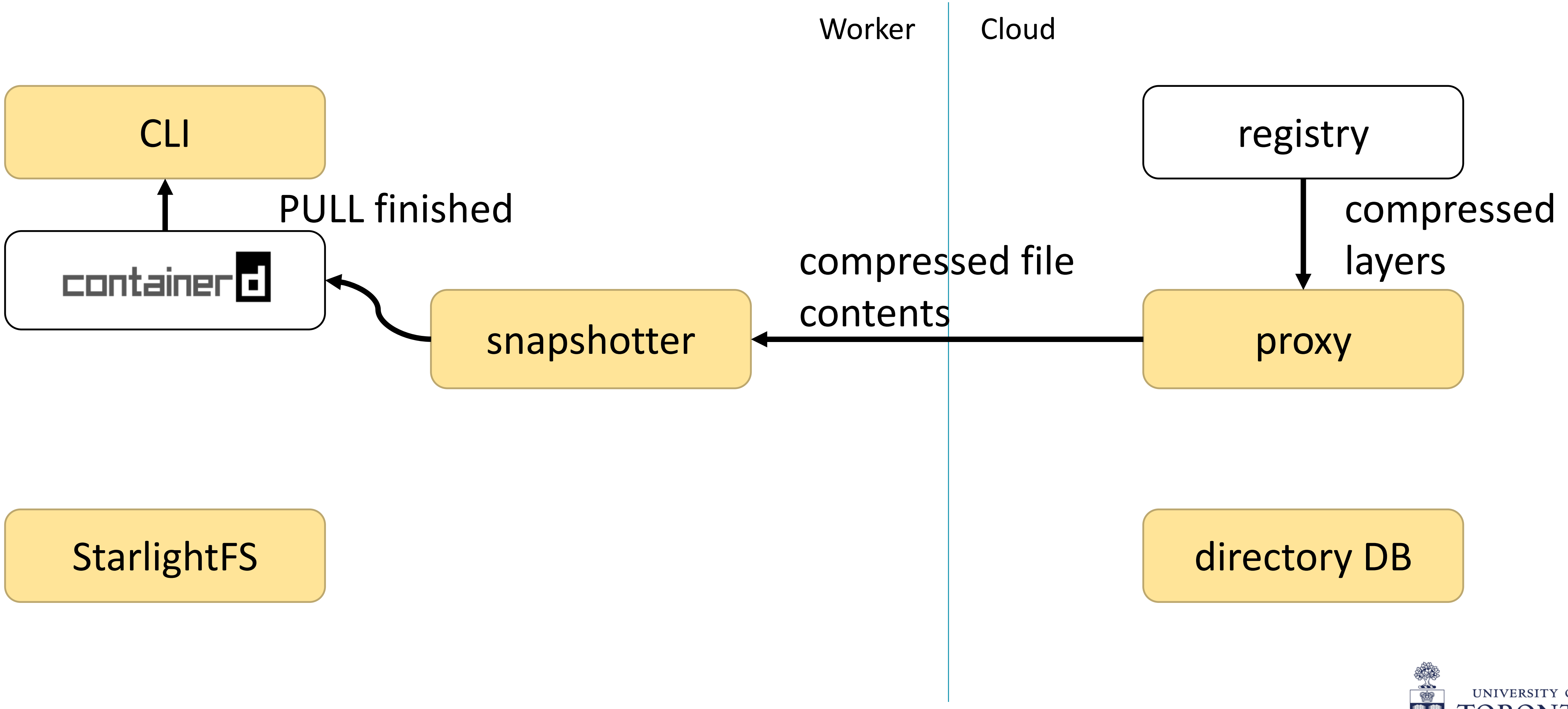
Starlight Operation



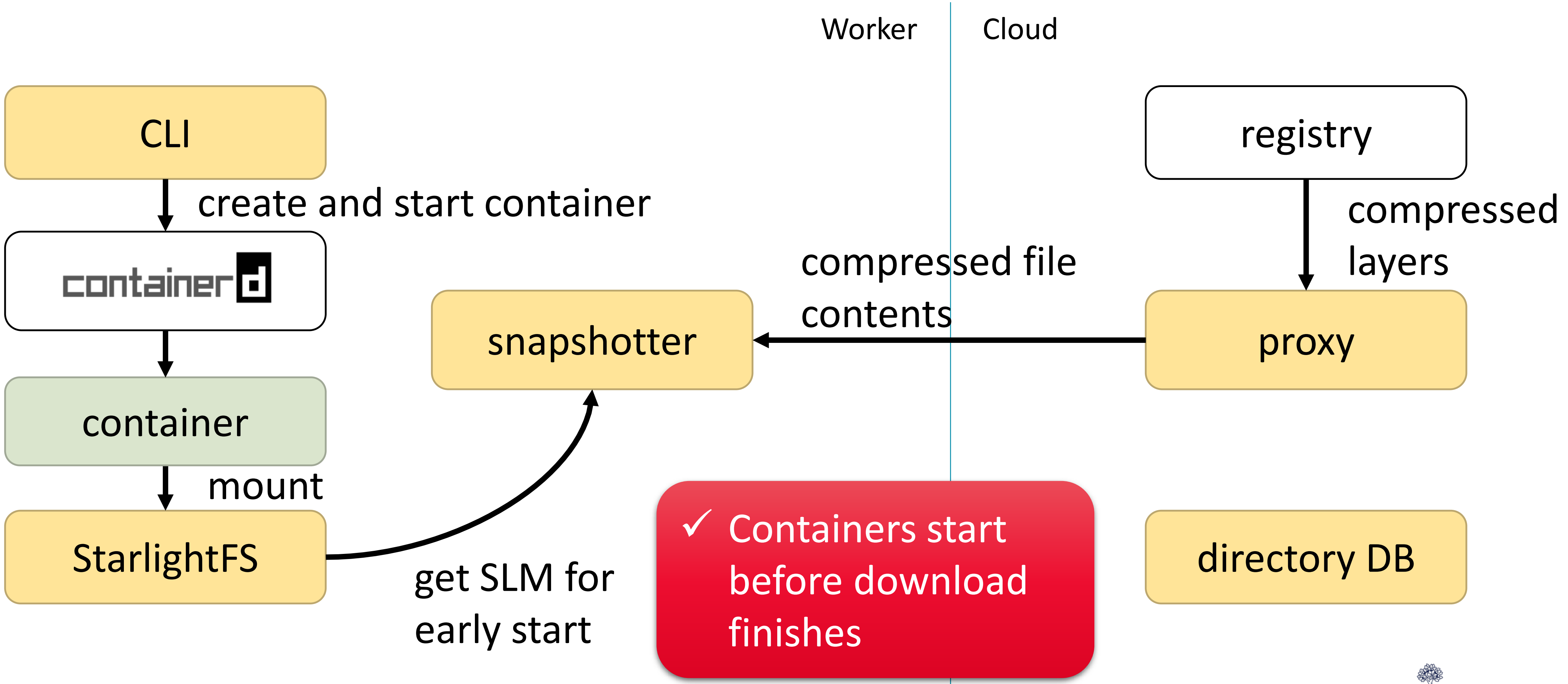
Starlight Operation



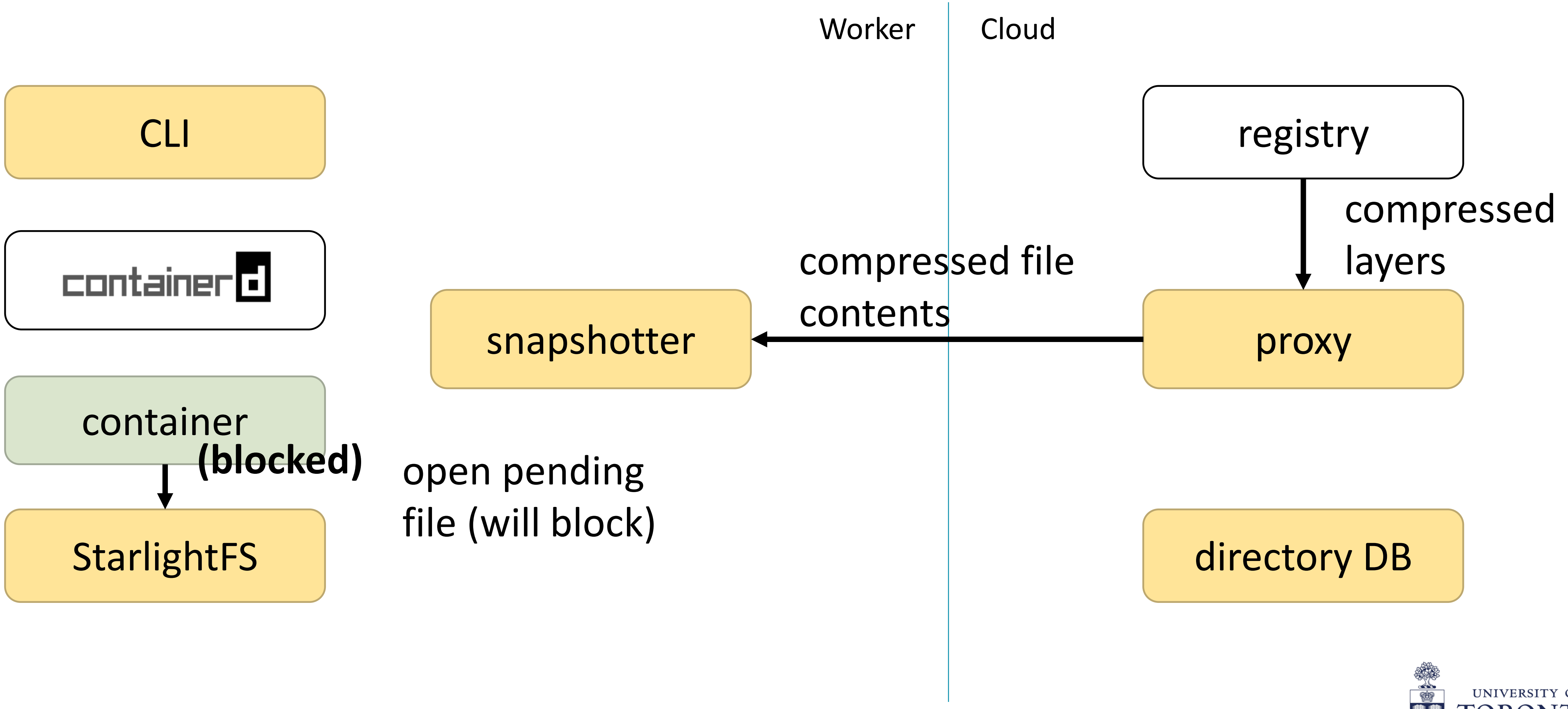
Starlight Operation



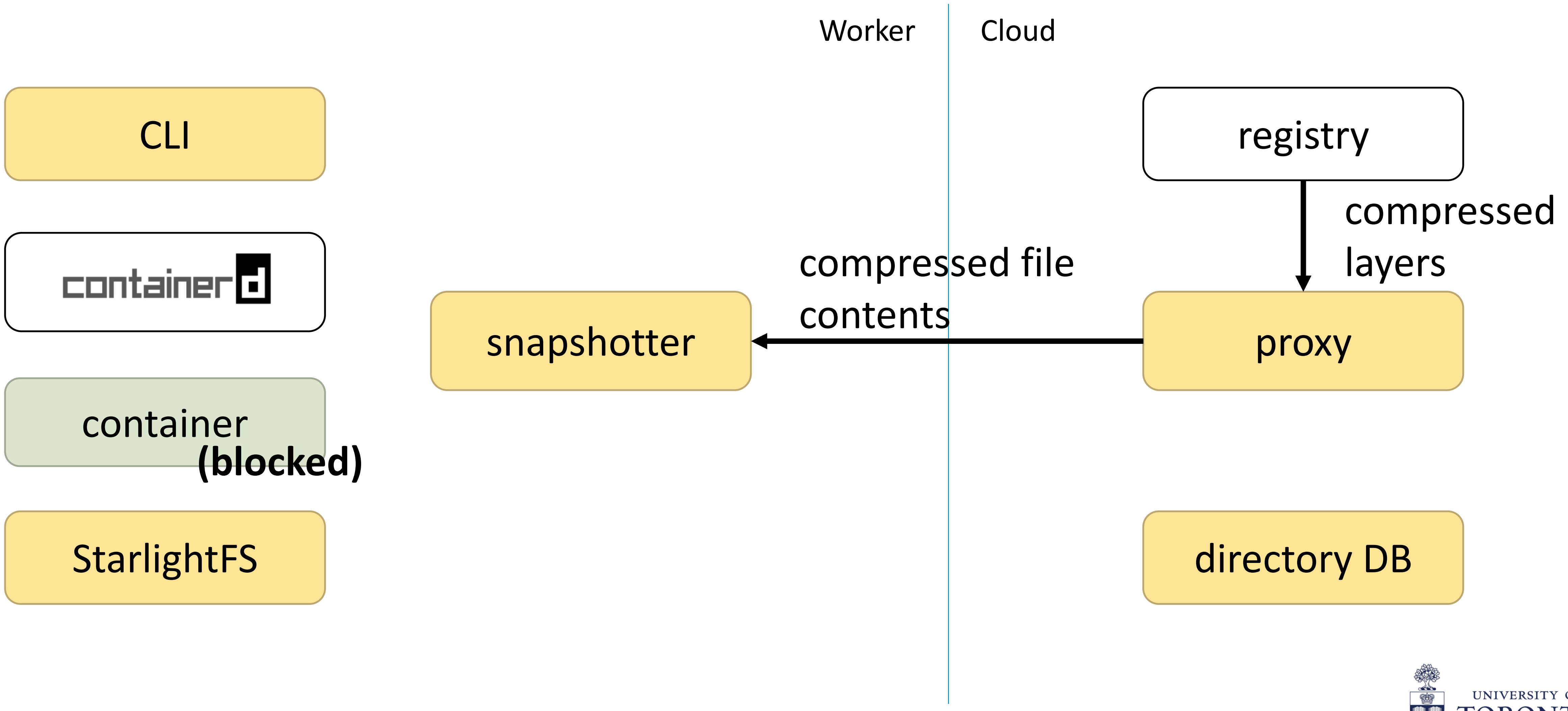
Starlight Operation



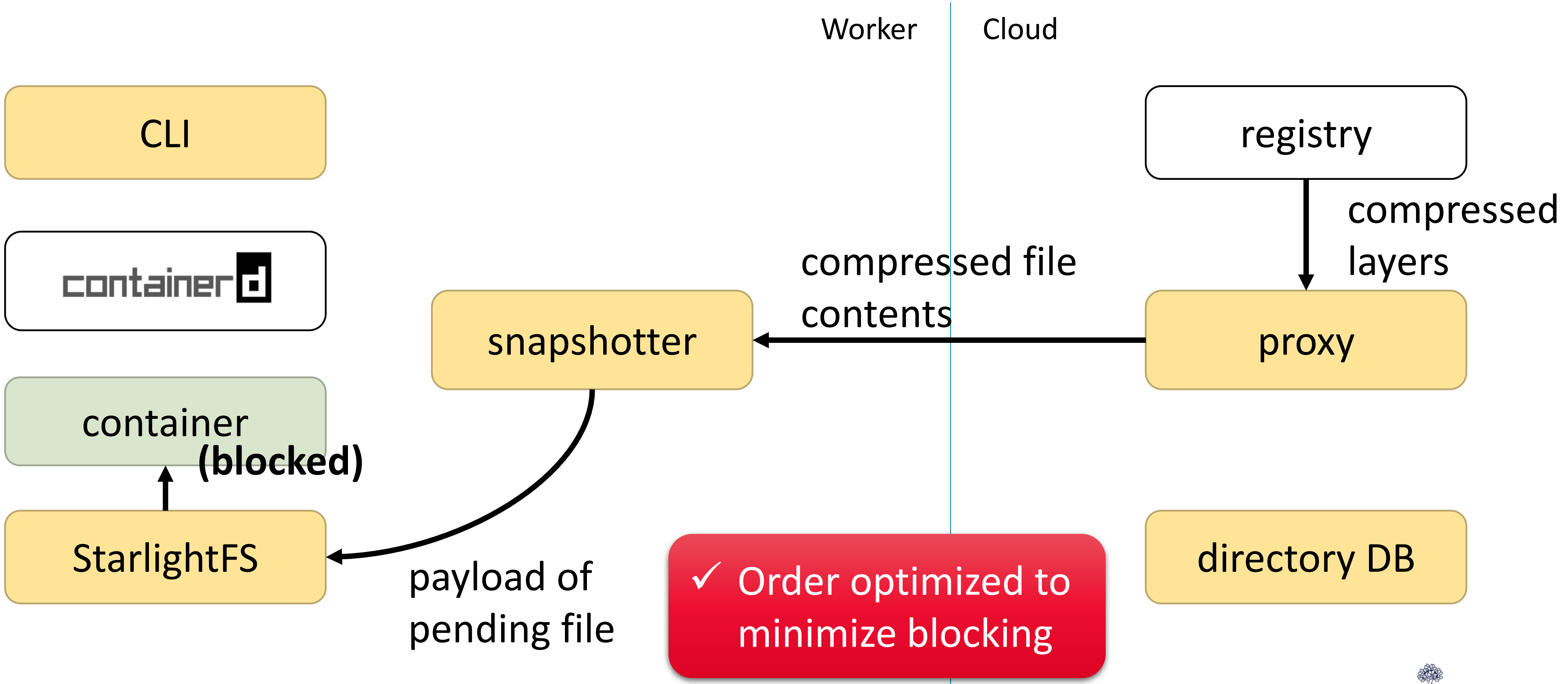
Starlight Operation



Starlight Operation



Starlight Operation



Details I Didn't Discuss

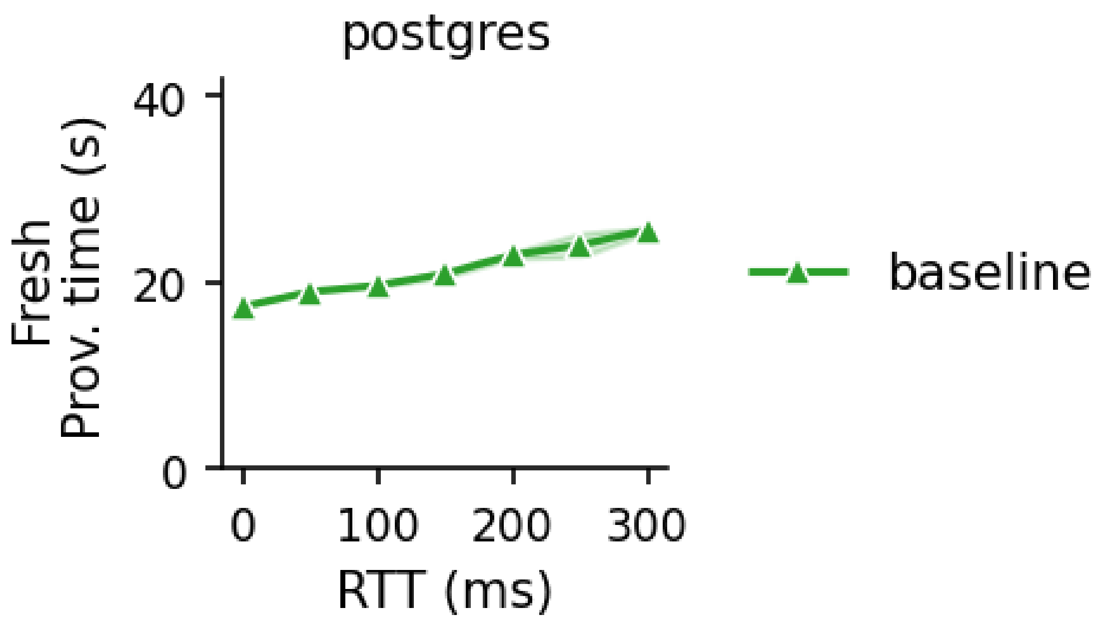
- ▶ Generating optimized SLMs
- ▶ Starflight filesystem
- ▶ Trace collection
- ▶ Seekable compressed layers format
- ▶ Directory DB
- ▶ Downloader and metadata manager

Evaluation

- ► 21 popular containers
  - ➢ 15B+ downloads in Docker Hub
  - ➢ Run each until ready
- ► Controlled deployments
  - ➢ tc controls bandwidth, RTT
- ► Real-world deployment
  - ➢ WAN covers 3 continents

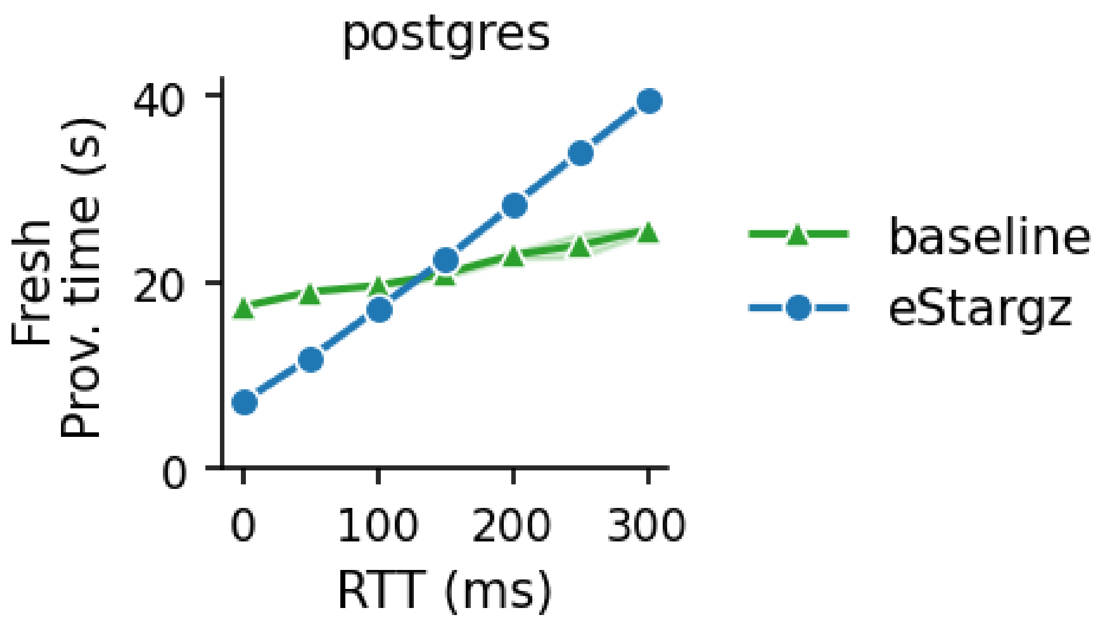
Effect of Round-trip Time

- X = RTT 0 to 300ms, 100Mbs
- Y = resulting provisioning time
  - **containerd v1.5.0**



Effect of Round-trip Time

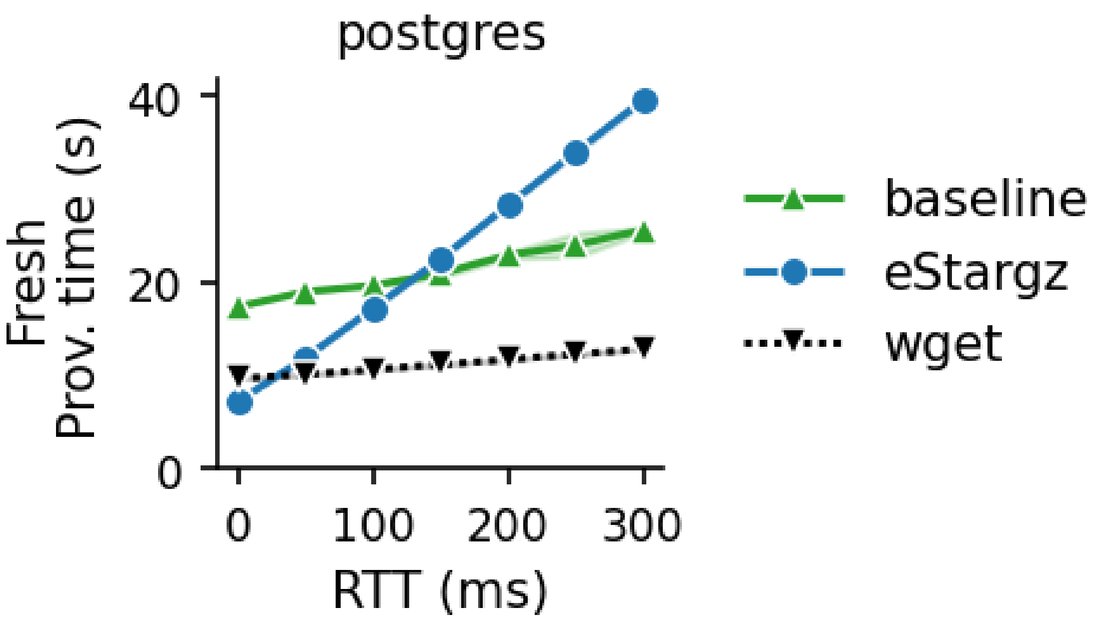
- ▶ X = RTT 0 to 300ms, 100Mbs
- ▶ Y = resulting provisioning time
  - ➢ **containerd v1.5.0**
  - ➢ **eStargz v0.6.3**: pull-based, on-demand



UNIVERSITY OF TORONTO

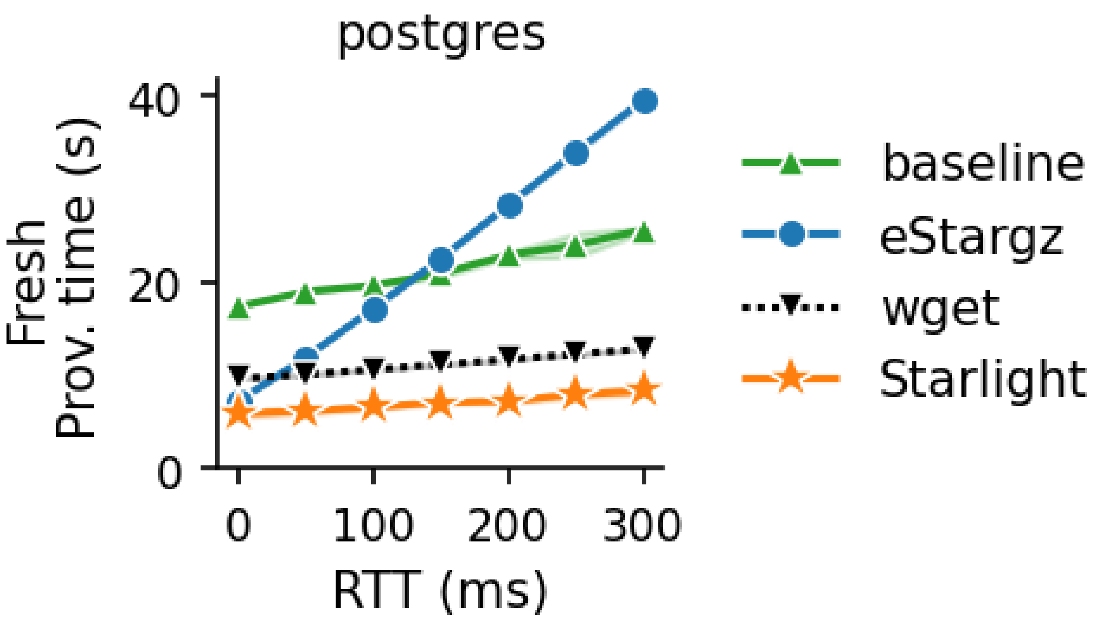
Effect of Round-trip Time

- X = RTT 0 to 300ms, 100Mbs
- Y = resulting provisioning time
  - **containerd v1.5.0**
  - **eStargz v0.6.3**: pull-based, on-demand
  - **Download** optimized delta bundle
    - Lower bound without early start



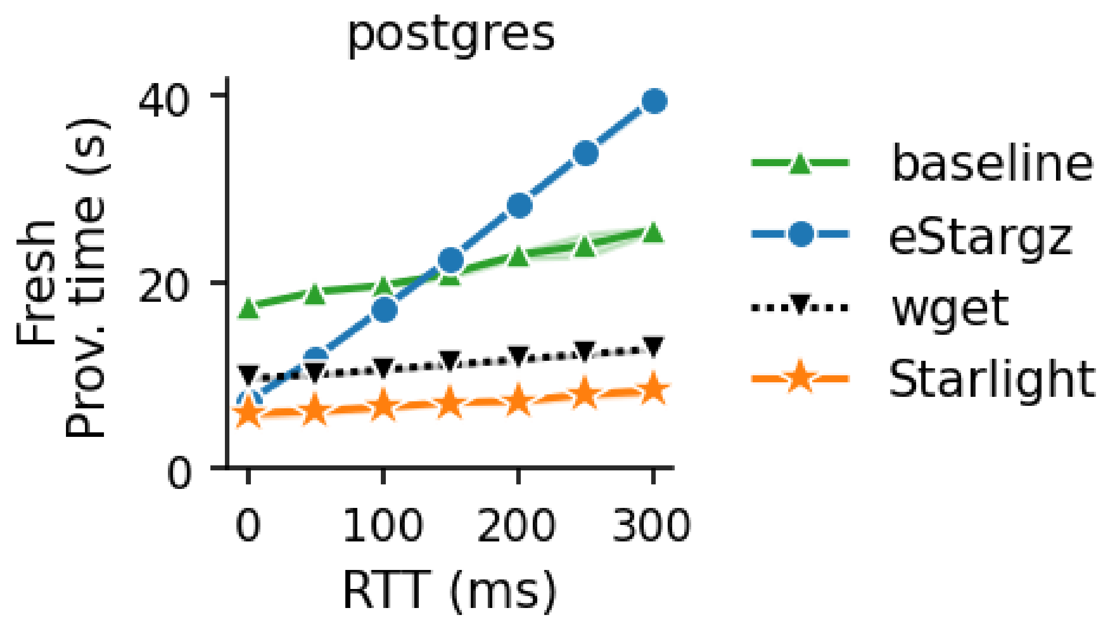
Effect of Round-trip Time

- ▶ X = RTT 0 to 300ms, 100Mbs
- ▶ Y = resulting provisioning time
  - ➢ **containerd v1.5.0**
  - ➢ **eStargz v0.6.3**: pull-based, on-demand
  - ➢ **Download** optimized delta bundle
    - ▪ Lower bound without early start
  - ➢ **Starlight**



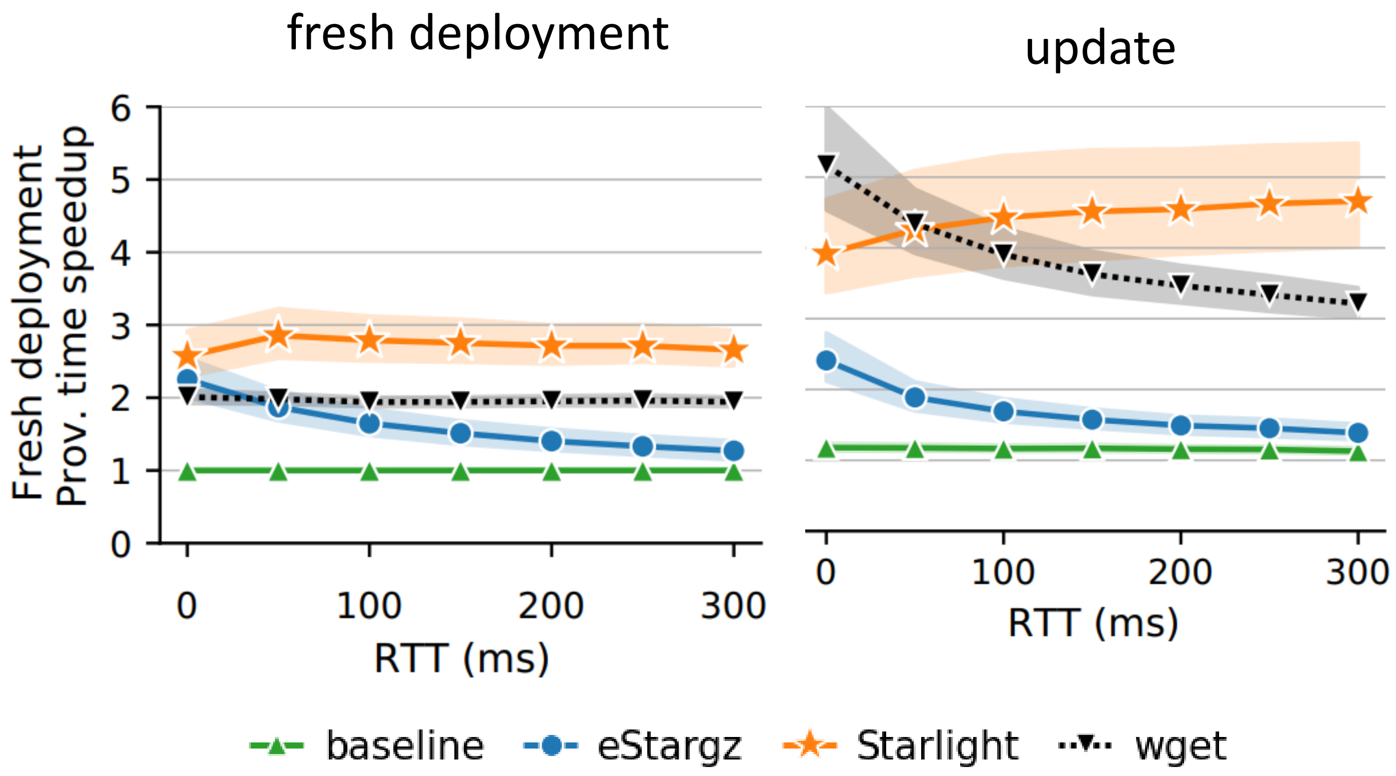
Effect of Round-trip Time

Starlight:
✓ Fastest (even with cloud RTT)
✓ Scales well with latency.
✓ Outperforms wget.



How Much Faster Are We?

speedup over containerd
(harmonic mean of 21 containers)

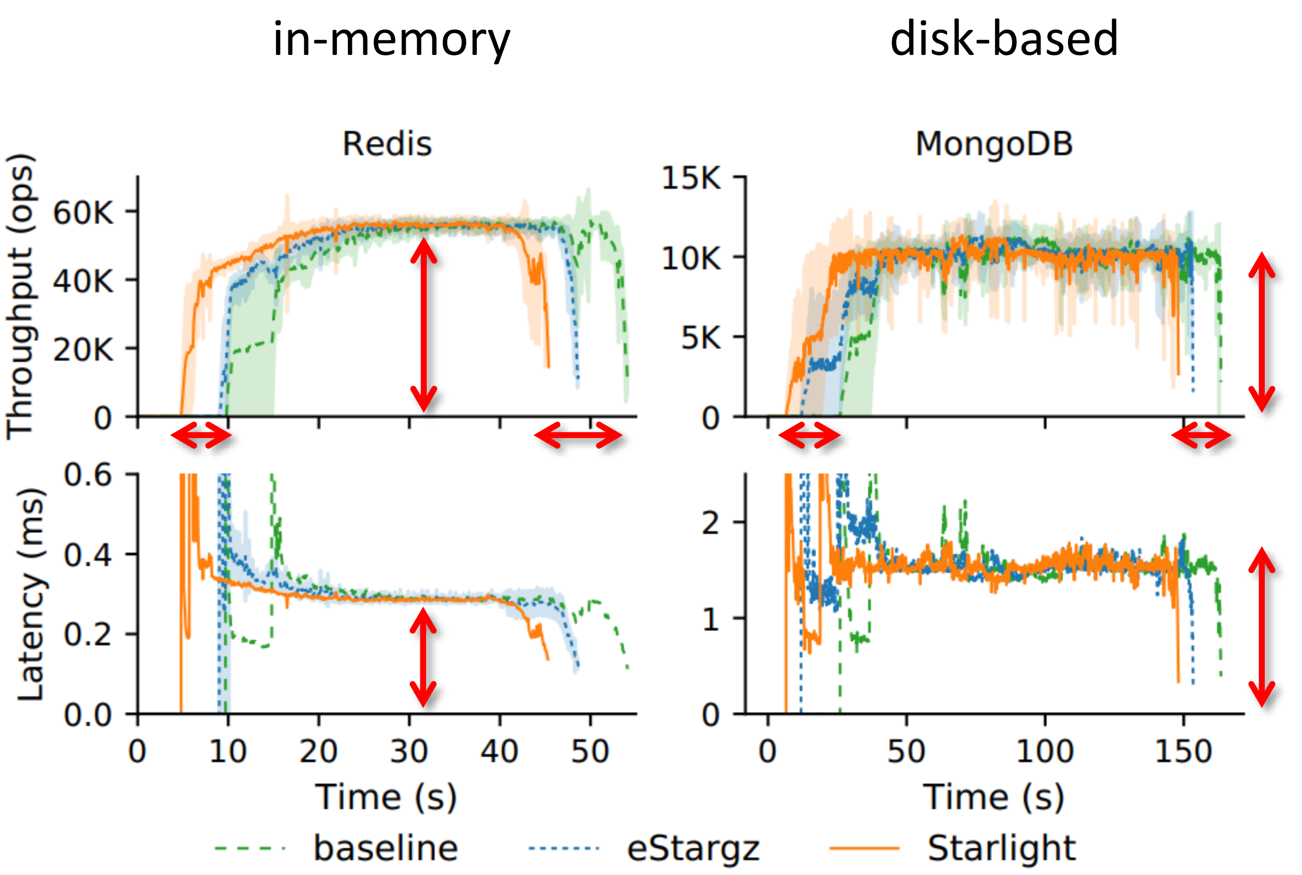


Starlight…
- ✓ Outperforms containerd 3x, estargz 1.9x
- ✓ Faster than wget
- ✓ Scales better with RTT.
- ✓ Extremely fast updates: 4—5x compared to fresh

Runtime Overhead?

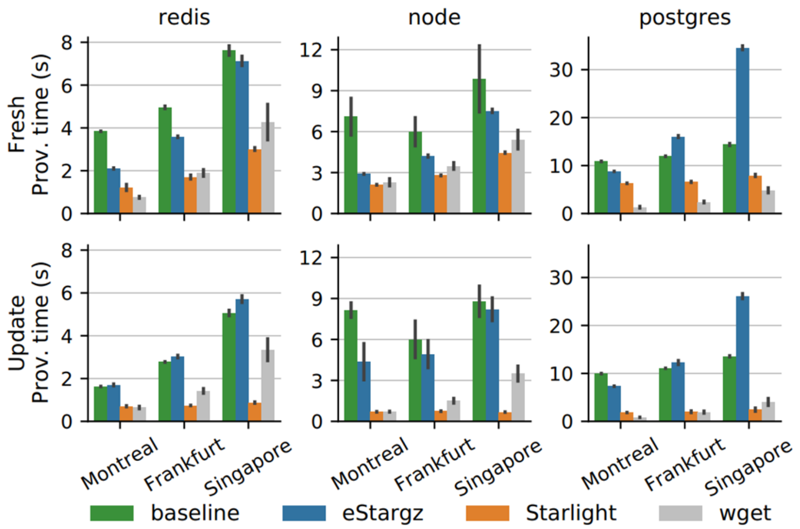
- ▶ 150ms, 100Mbs
- ▶ YCSB Workload A
- ▶ In-memory and on-disk database.

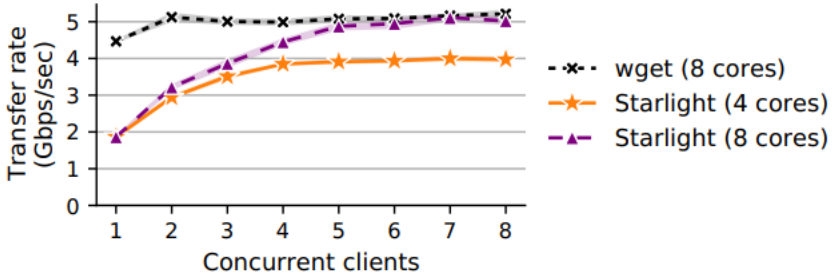
No overhead:
✓ Workers start earlier, finish faster.
✓ Same peak performance.



Additional Experiments

- ▶ **On WAN → similar results.**
- ▶ **In cloud → Starlight fastest.**
- ▶ Bandwidth
- ▶ Scalability
- ▶ Proxy memory and runtime
- ▶ Detailed case analysis





Conclusion and Future Work

- Container provisioning is slow.
  - Layered structure
  - Pull-based protocols
- Starlight: new provisioning protocol, filesystem, storage format.
  - ✓ Faster deployment on edge, cloud
  - ✓ Backwards compatible, transparent
  - ✓ No overhead
  - ✓ Open source

- Future work:
  - Collect traces online
  - Predict/learn file order
  - Jointly optimize multi-container deployments

https://github.com/mc256/starlight

mgabel@cs.toronto.edu