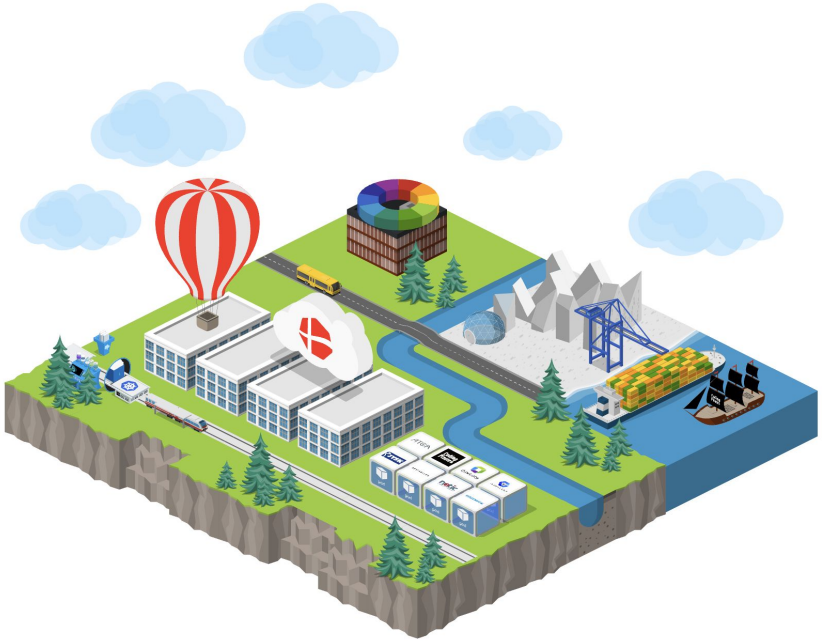
Migrating 2 million CPU cores to Kubernetes



Lucy Sweet (lucysweet@uber.com) and
Mathias Schwarz (schwarz@uber.com)

Uber


Uber


Uber
Uber
Eats

Uber Denmark Engineering

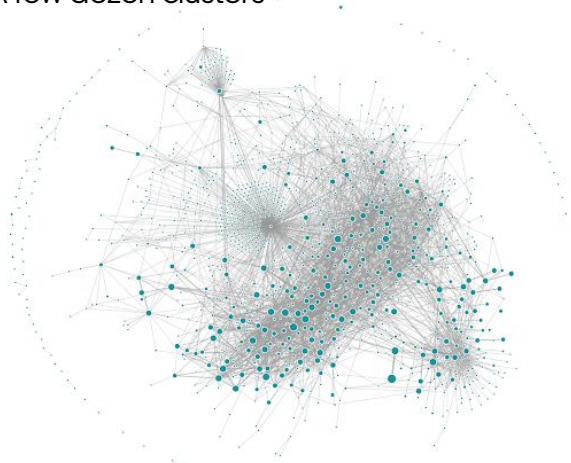
- Located in Aarhus C
- Opened in 2014
- 120+ Employees
- 2 engineering charters: Infrastructure and Delivery
- The team works on critical parts of Uber’s infrastructure and Delivery products

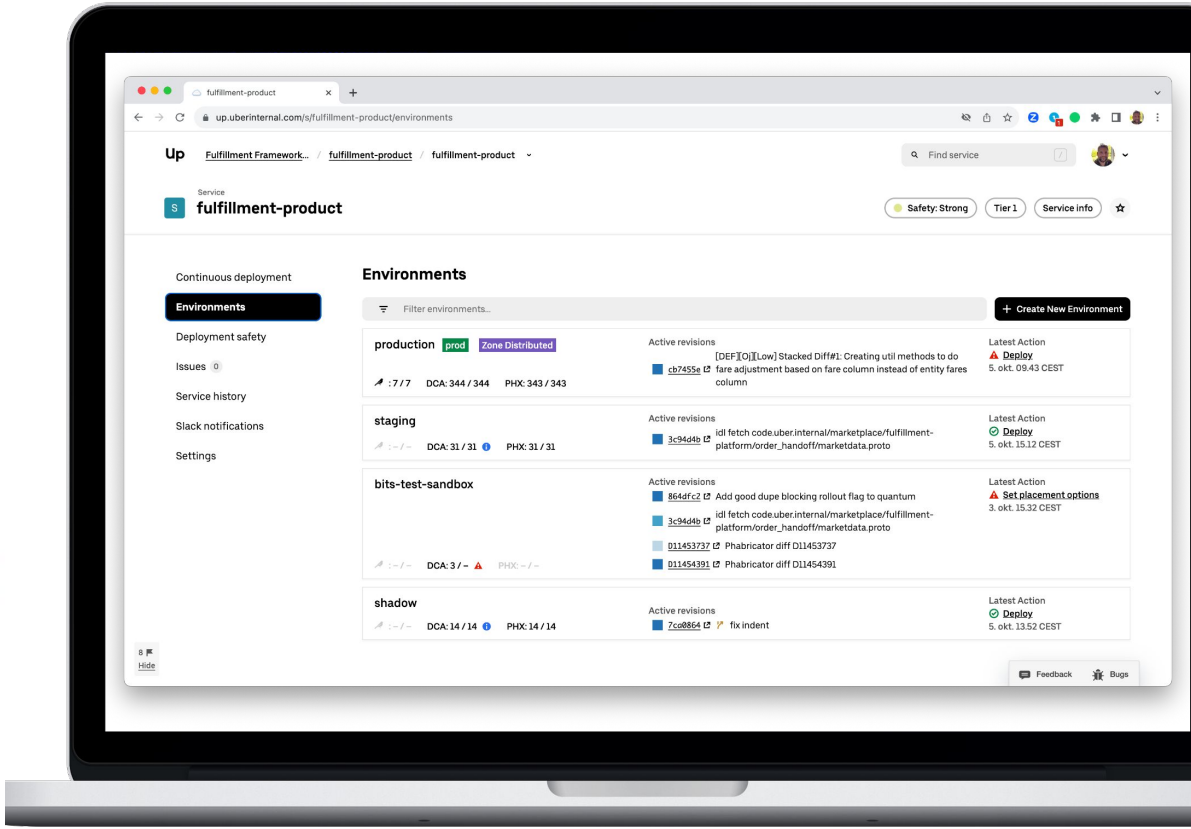
Global control plane

Up global control plane

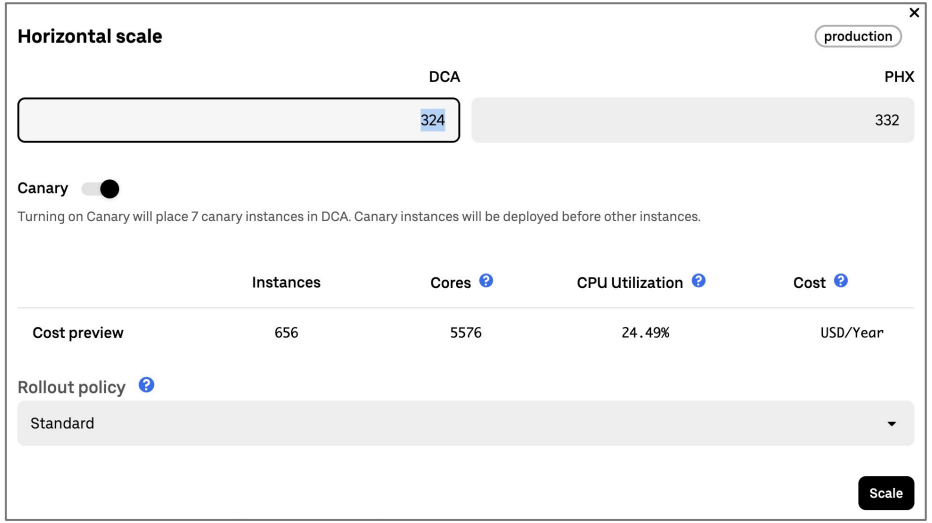
Common platform for all stateless services:

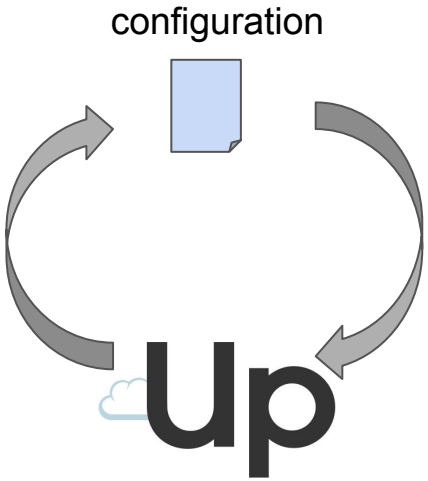
- **4500** services
- **2.78M** CPU cores
- **31K** weekly production deploys
- A few dozen clusters





High level federated service goalstate





Declarative configurations

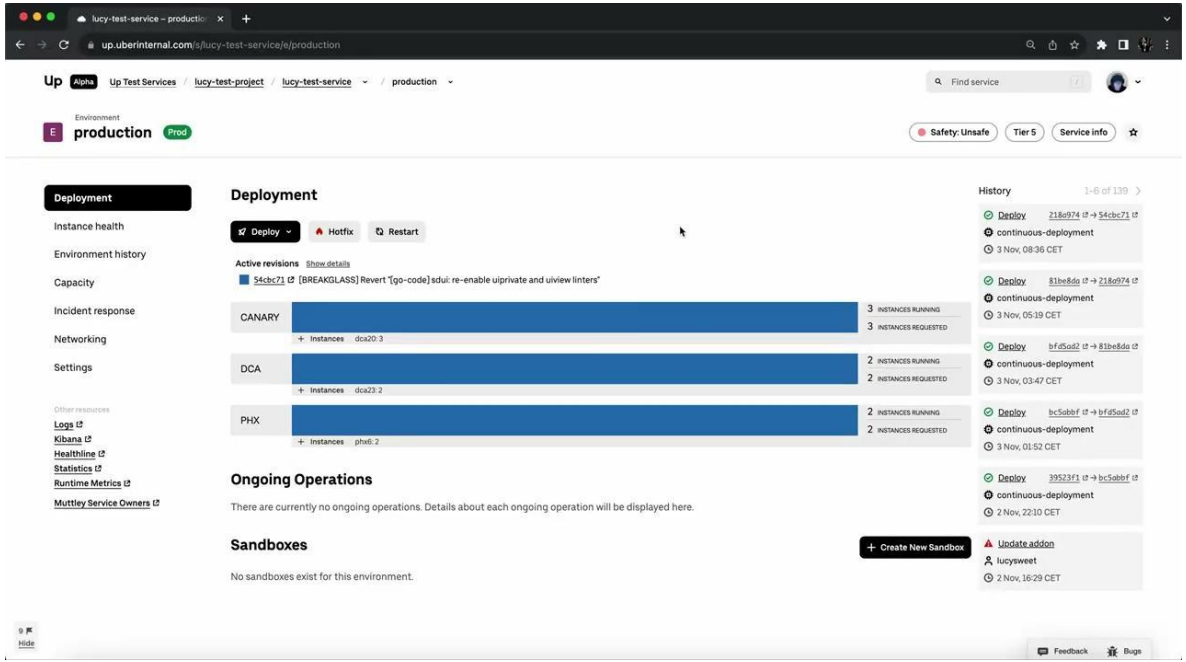
- Region placement
- Dependencies

Continuous evaluation

Optimal application of the configuration

Demo time

- https://up.uberinternal.com/s/lucy-test-service/e/production
- Contingency if demo doesn't work:



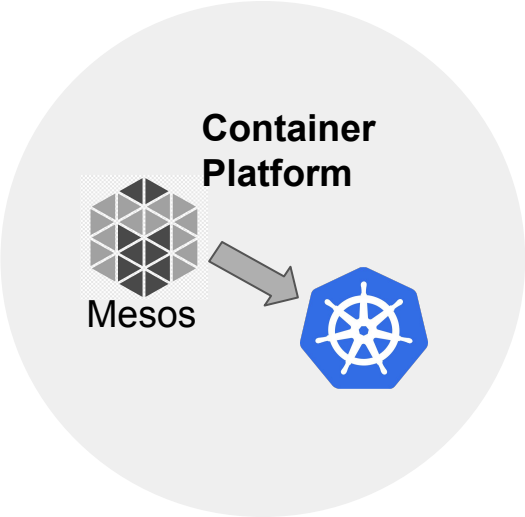
Migration to Kubernetes

Why?

- De facto standard
  - Used by 50% of Fortune 500
  - More than 80% of SMB
- Mesos was past its prime
- Well supported by all cloud providers
- Consolidation of all container orchestration
  - Stateless services
  - Stateful Databases
  - LLMs
  - Batch jobs

How?

- No impact to our product engineers
- Transparent migration
- Federation



Defining the right contract

Portability

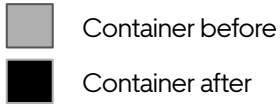
Decoupling services from the underlying physical infrastructure and scheduling software

Touchless migration

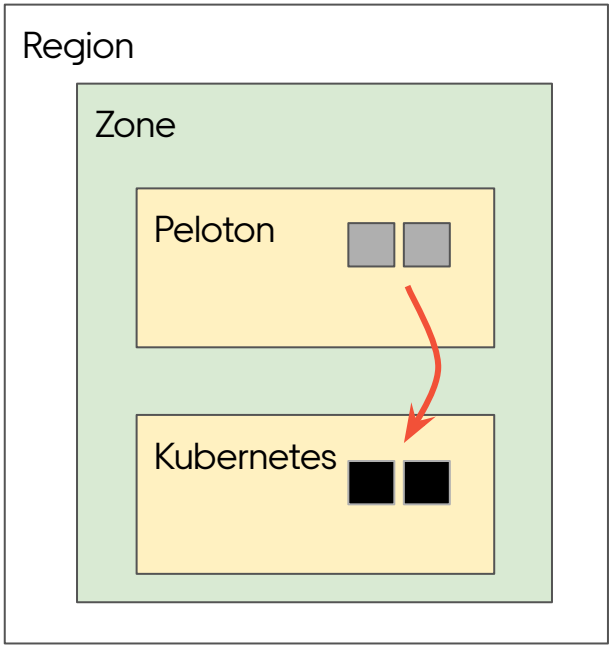
Zero downtime

Continuous migration

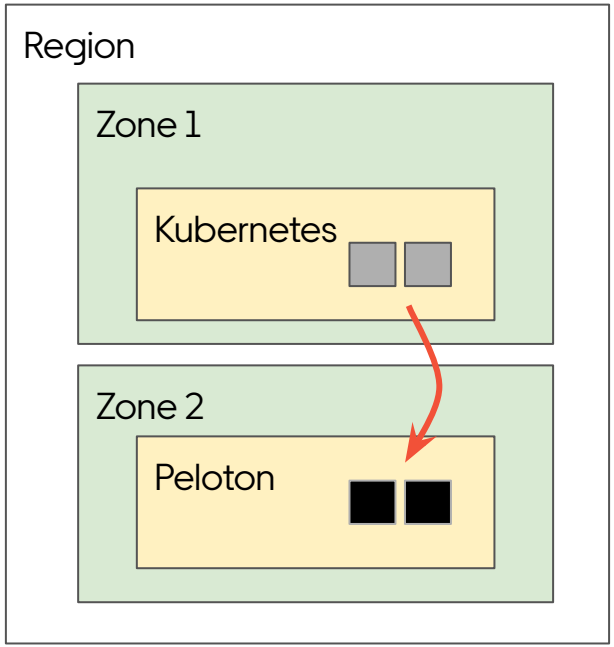
Portability - Movement tolerance



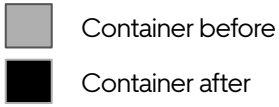
Onboarding to Kubernetes



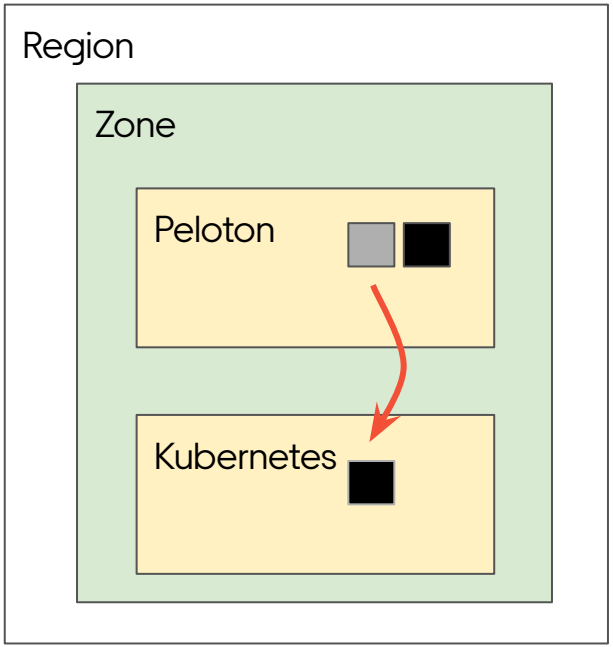
Offboarding from Kubernetes



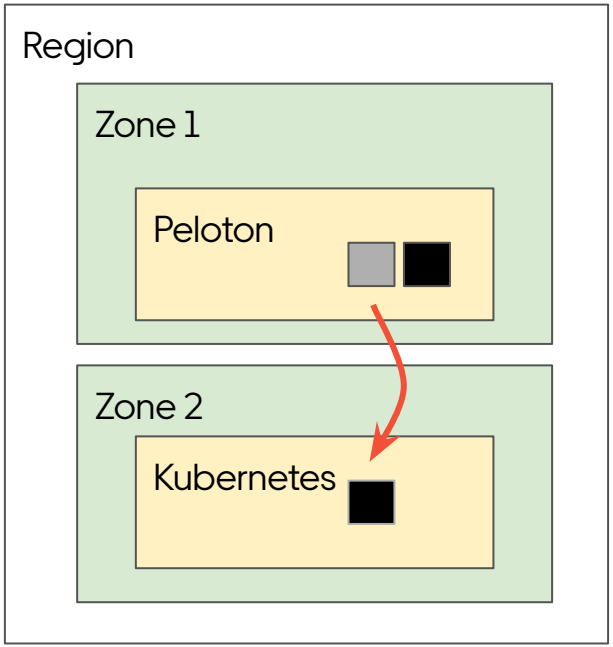
Portability - Distributability



Across clusters

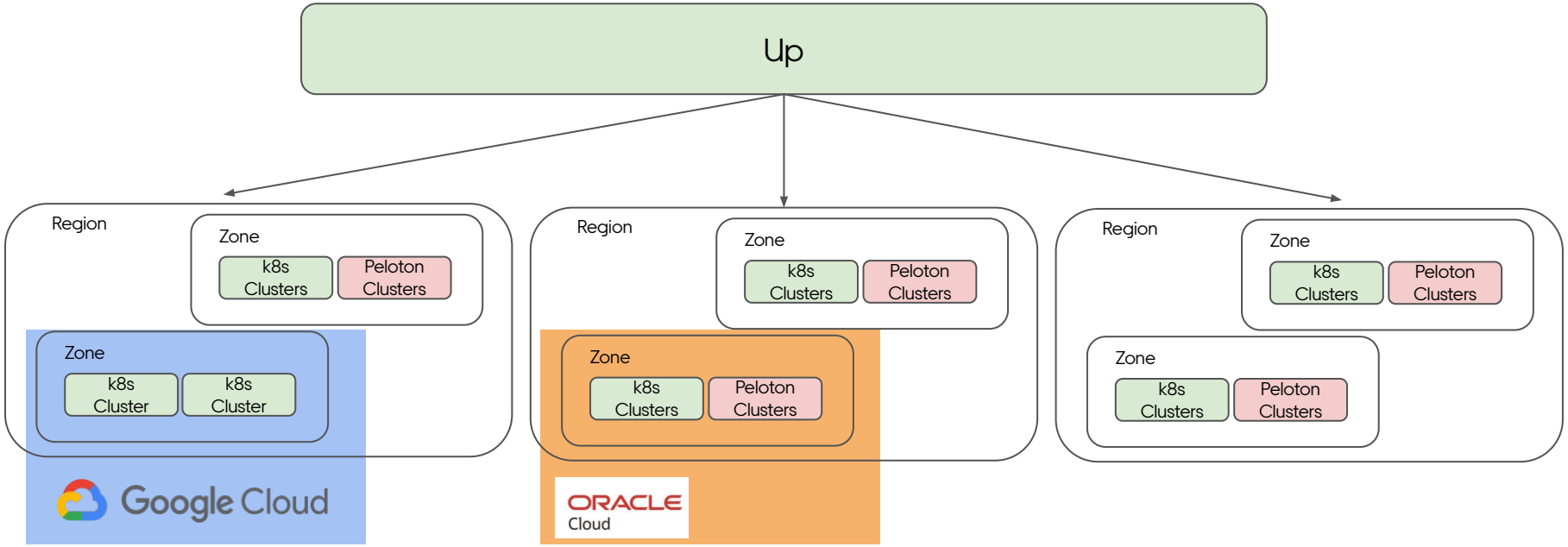


Across zones

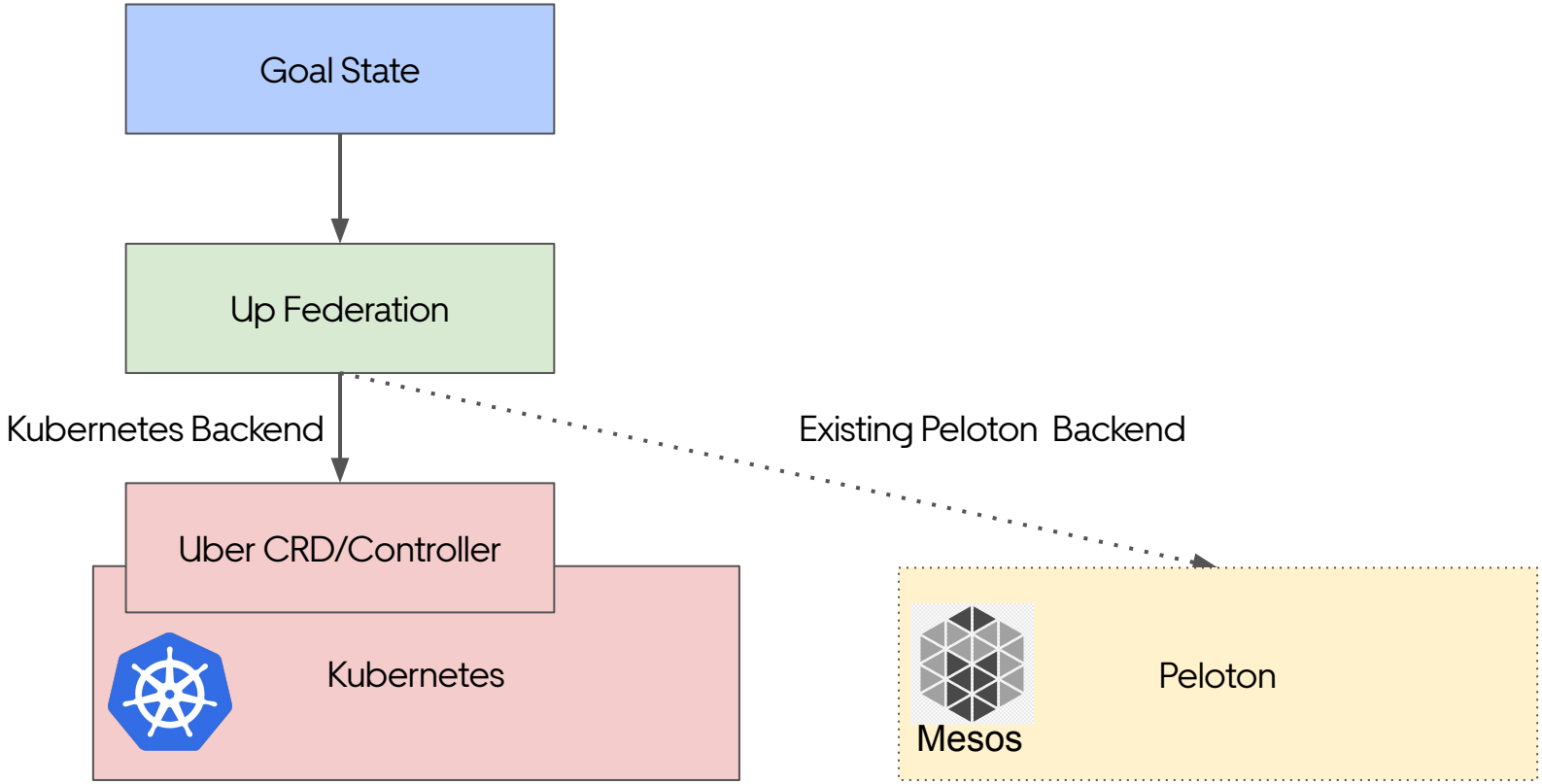


Our Kubernetes Fleet

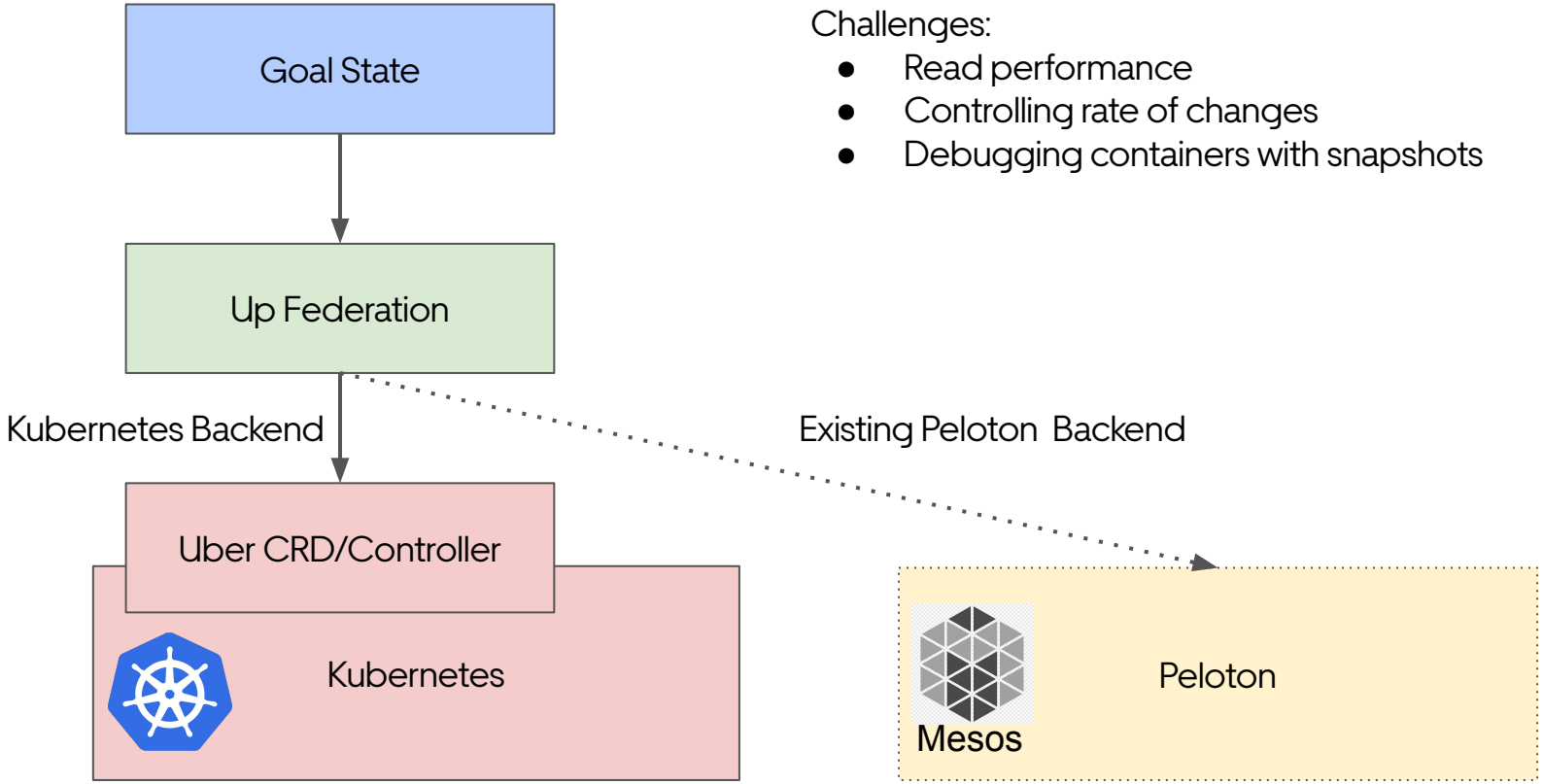
- 19 clusters in 4 “zones” and 2 “regions”
- 18,000 hosts / 2 million CPU cores
- Mix of CPU arches (amd64/arm64) and GPUs (RTX5000/etc)
- On prem and public cloud providers



Rebuilding Up in Kubernetes



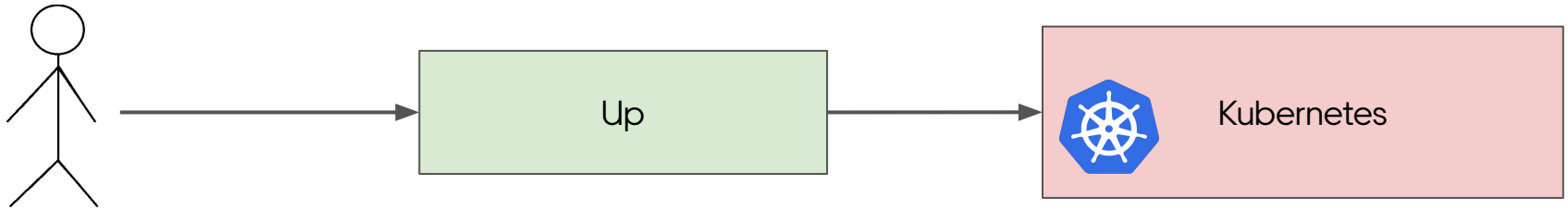
Rebuilding Up in Kubernetes



Challenges:

- Read performance
- Controlling rate of changes
- Debugging containers with snapshots

Read Performance



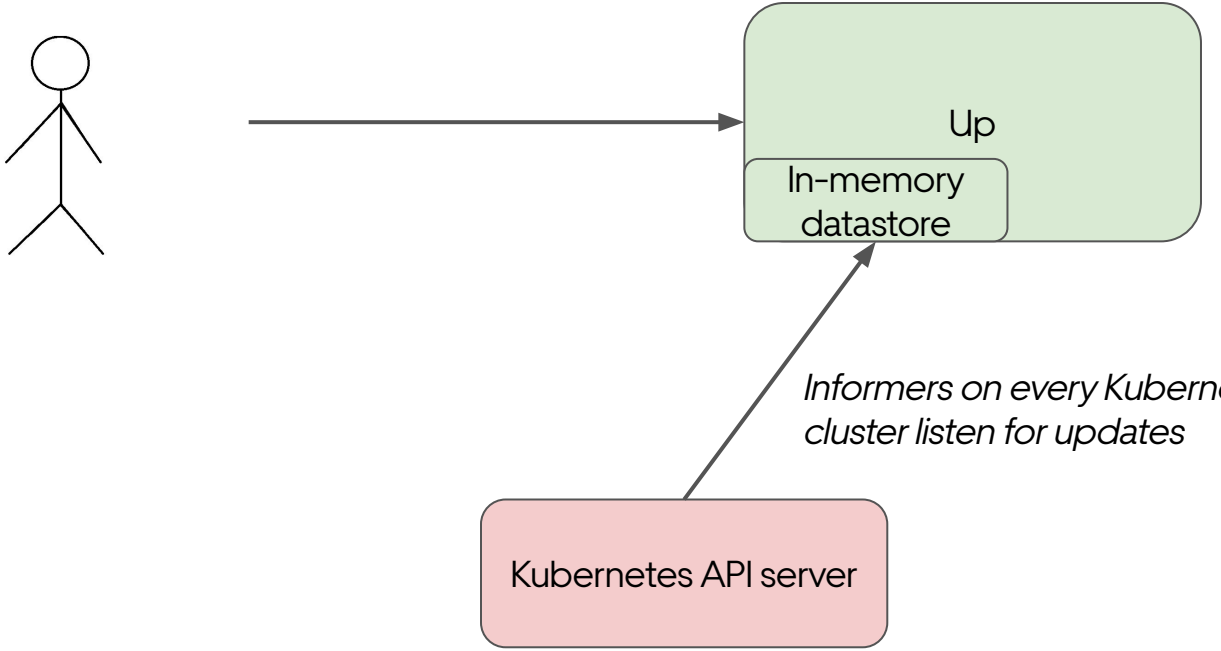
500 concurrent engineers making requests

Requests fan out to list API on all clusters

ETCD read contentions

Read Performance

"Tell me about every pod associated with service X across all our clusters"



Controlling rate of changes

- Some systems can't tolerate large amounts of disruption (particularly sharded services)
- There's a large amount of risk to moving everything at once to a new service version (what if the new version doesn't work properly)
- Introducing changes gradually allows us to monitor their effects
  - If we see signs of problems (health checks, error rates, etc) we can roll back to the previous state

Controlling rate of changes

11:47 [ ] 11:53

dca20
11:47 → 11:49

dca20-kubernetes-stateless01 11:47→11:48
restart 1 instances

- uMonitor — Monitoring 5 alerts
- Kubernetes — Done (1 of 1) 00:17
- UNS — instances are present in UNS

dca20-kubernetes-stateless01 11:48→11:49
restart 2 instances

- uMonitor — Monitoring 5 alerts
- Waited 01:12
- Kubernetes — Done (2 of 2) 00:08
- UNS — instances are present in UNS

dca23
11:49 → 11:51

dca23-kubernetes-stateless01 11:49→11:51
restart 2 instances

- uMonitor — Monitoring 6 alerts
- Waited 01:12
- Kubernetes — Done (2 of 2) 00:18
- UNS — instances are present in UNS

phx6
11:51 → 11:53

phx6-kubernetes-stateless01 11:51→11:53
restart 2 instances

- uMonitor — Monitoring 6 alerts
- Wait 02:00
- Kubernetes — Upgrading compute job: up.lucy-test-service.production.00000000000000000000000 0.compute-0.f8fac343-95ef-4935-ad48-4e882a2c9d8c (0 of 2) 00:00
- UNS

Controlling rate of changes

12:29 [— 12:38 —] 12:

dca20
12:29 → 12:38

dca20-kubernetes-stateless01 12:29→12:30
update 1 instances
- uMonitor — Monitoring 5 alerts
- Kubernetes — Done (1 of 1) 00:13
- UNS — instances are present in UNS

dca20-kubernetes-stateless01 12:30→12:32
update 2 instances
- uMonitor — Monitoring 5 alerts
- Waited 02:00
- Kubernetes — Done (2 of 2) 00:09
- UNS — instances are present in UNS

dca20-kubernetes-stateless01 rollback 12:38→12:38
update 1 instances
- Kubernetes — Done (1 of 1) 00:07
- UNS — instances are present in UNS

dca20-kubernetes-stateless01 rollback 12:38→12:38
update 2 instances
- Kubernetes — Done (2 of 2) 00:09
- UNS — instances are present in UNS

dca23
12:32 → 12:38

dca23-kubernetes-stateless01 12:32→12:34
update 2 instances
- uMonitor — Monitoring 6 alerts
- Waited 02:00
- Kubernetes — Done (2 of 2) 00:18
- UNS — instances are present in UNS

dca23-kubernetes-stateless01 rollback 12:38→12:38
update 2 instances
- Kubernetes — Done 00:05
- UNS — instances are present in UNS

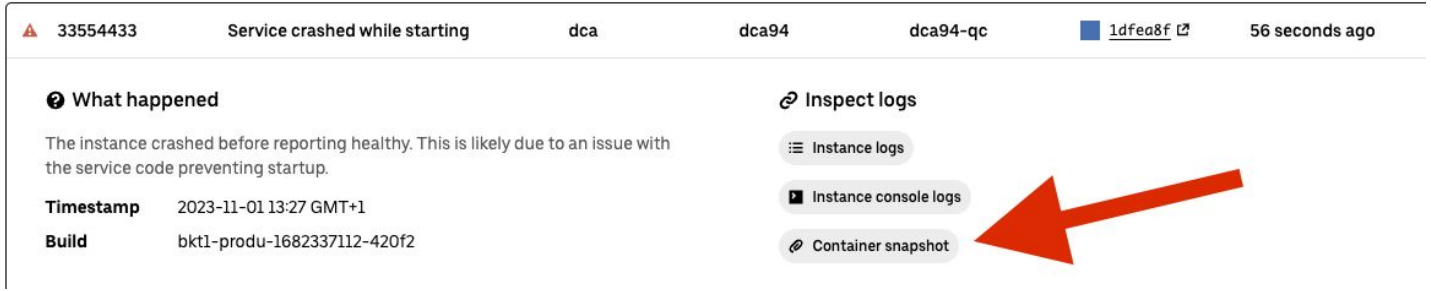
phx6
12:34 → 12:38

phx6-kubernetes-stateless01 12:34→12:38
update 2 instances
- uMonitor — Monitoring 6 alerts
- Waited 02:00
- Kubernetes — Done (2 of 2) 01:21
- UNS — instances are present in UNS

phx6-kubernetes-stateless01 rollback 12:38→12:38
update 2 instances
- Kubernetes — Done (2 of 2) 00:11
- UNS — instances are present in UNS

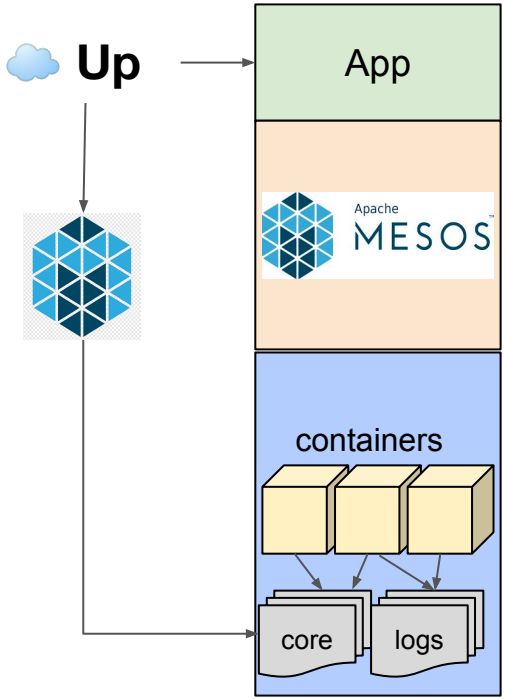
Container snapshots

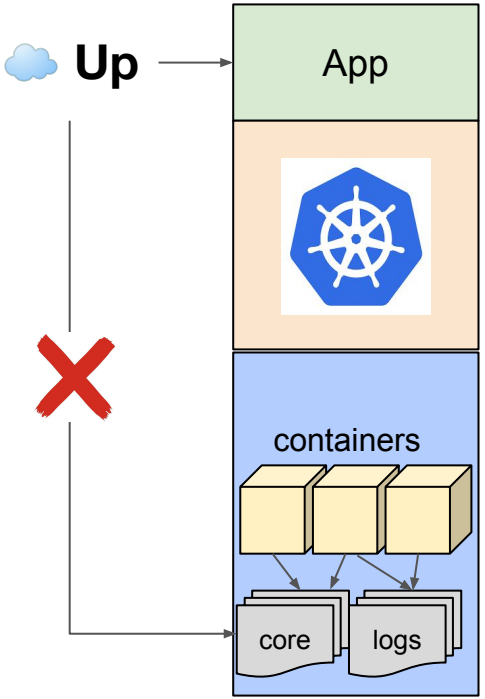
- Up allows Uber engineers to see a snapshot of their container at the point it terminated



- We want to continue to support this into Kubernetes

Container snapshots

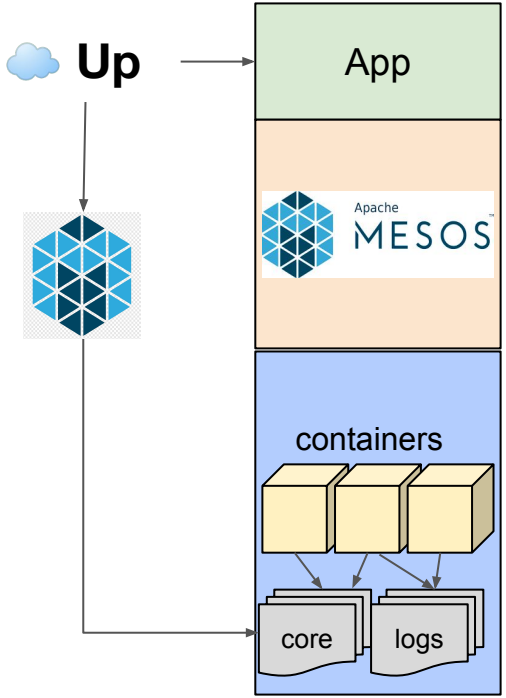




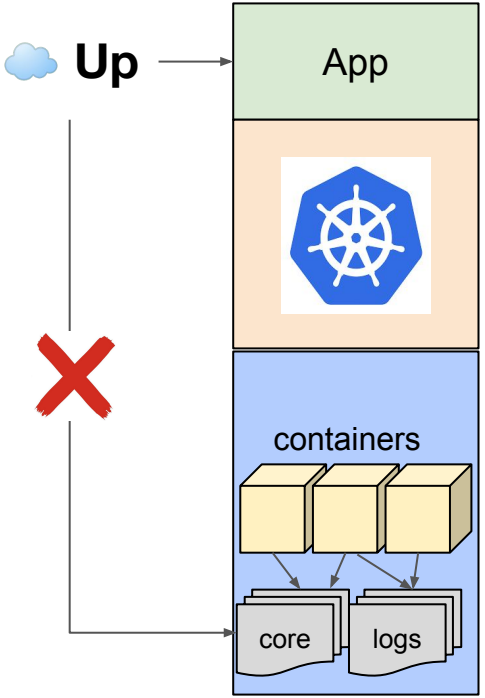
Peloton
Container artifacts (core, heap, logs) not cleaned up on exit. Accessible on Up via Mesos.

Native Kubernetes
Container artifacts are deleted on container exit.

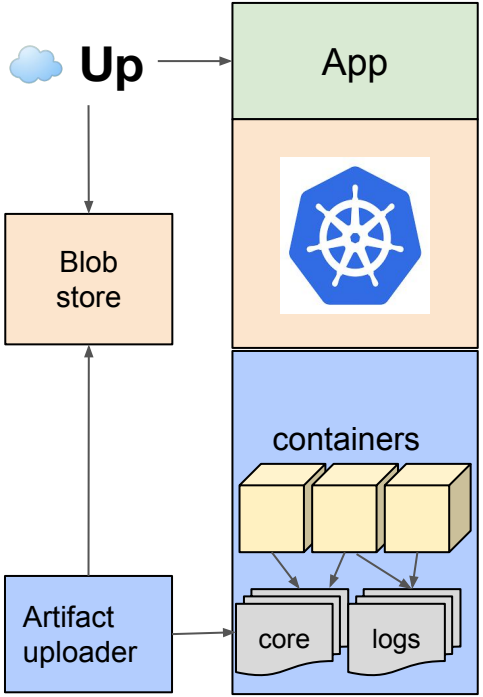
Container snapshots



Peloton
Container artifacts (core, heap, logs) not cleaned up on exit.
Accessible on Up via Mesos.



Native Kubernetes
Container artifacts are deleted on container exit.

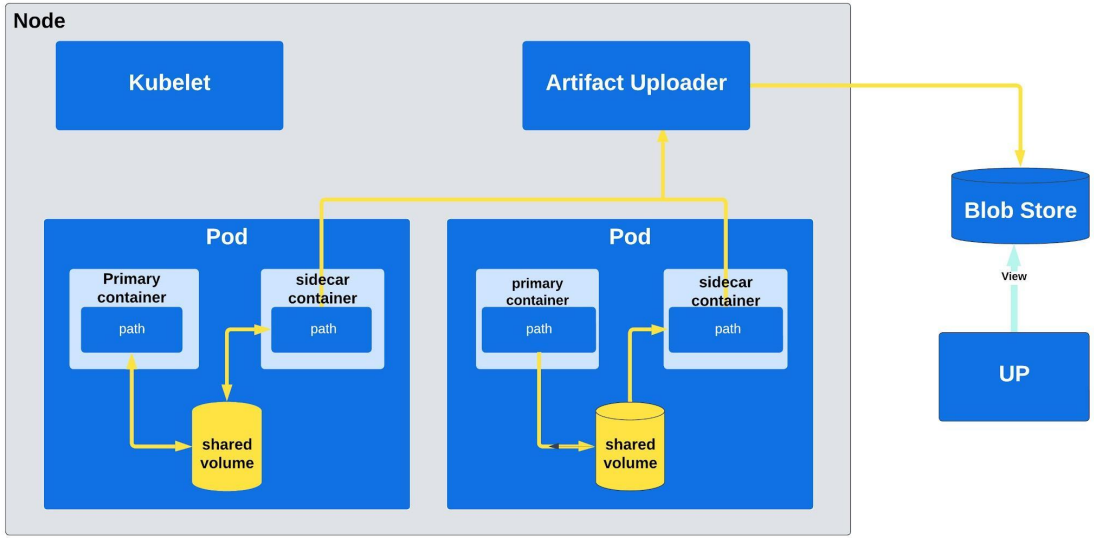


Uber Kubernetes Engine
Container artifacts uploaded to blob store on container exit.
Accessible on Up via blob store.

Container snapshots

- Sidecar Container
  - Share a volume with the primary container
  - Pauses pod deletion after primary container exits

- Artifact Uploader
  - Upload artifacts after primary container exists
  - Kill sidecar container, enable pod deletion



Container snapshots

TerraBlob

Read Docs | Change Theme | + Create Datastore

/prod/k8s-sandbox-uploader/dca22-kubernetes-stateless01/stateless

stateless

Upload Blobs | Create Directory

Browse | README | Configuration

access-decider-production-compute-0-1325500264

Name	Size	Created at
access-decider-production-compute-0-1325500264-2023-10-30_20-43-57.tar	68.0M	31 Oct 2023, 04:55

Return to Demo

Learnings

Kubernetes is not a silver bullet...

- We had to build a lot of things on top of Kubernetes, it didn't work "out of the box for us"
- Lots of CRDs in use across our setup
- We don't directly use most of the typical resources (Deployments, StatefulSets) because of specific requirements we have

...but its flexibility is its strength

- The fact we can diverge from the default behaviour so easily is part of the strengths of Kubernetes
- It isn't opinionated about how you should run and deploy your infrastructure, which makes it easier for us to onboard all different shapes and sizes of services

Learnings

The first million cores are the easiest…

- Most services are rather non-opinionated and don't have special requirements
- In our portability drive before starting our Kubernetes migration we managed to move >80% of services with very minimal effort

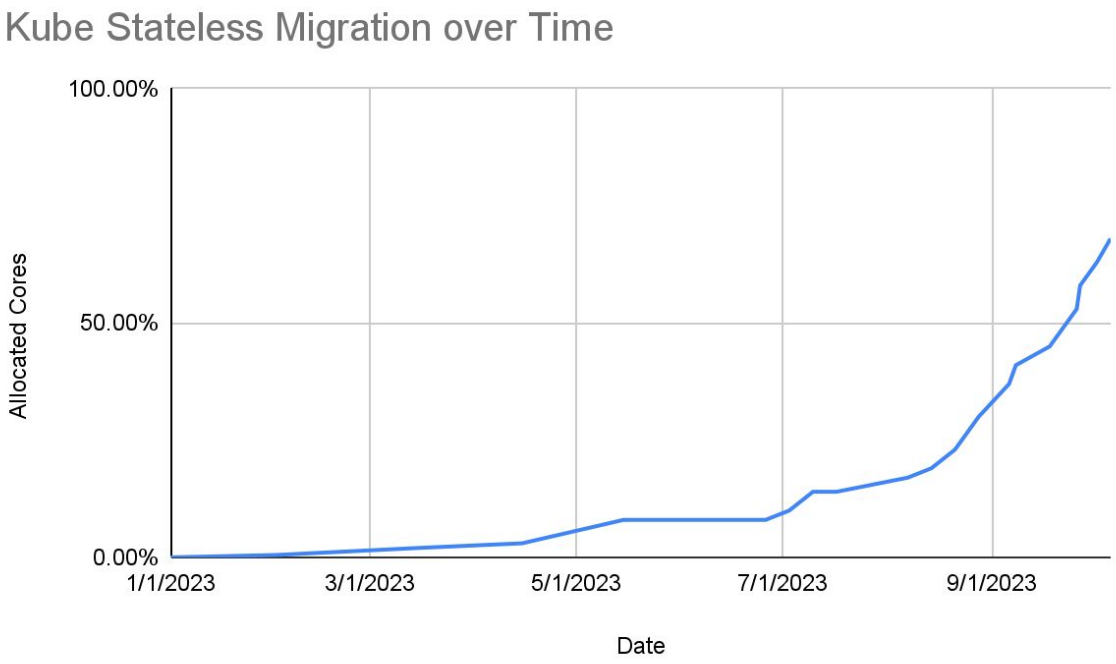
… because users will build dependencies on all sorts of undocumented behaviour

- We found lots of our users had built a dependence in their service on behaviours on our old system, Peloton
- Some of these behaviours weren't even documented, and some of the dependencies weren't either
- Our ability to slowly move services to Kubernetes or Peloton allowed us to acceptance test services and see which rollouts ended in a rollback

Migration status

Migration Progress

- > 70% of the fleet
- Multiple 5K node clusters
- Largest cluster stats:
  - 50k+ pods
  - 7k+ deployments
  - 5.5k+ nodes



Next steps...

STATELESS

BATCH

Michelangelo

Workbench

Apache Flink

STATEFUL

DAEMONSET

M3 Collector

Service Mesh

Uber

Blog Post