

Migrating 2 million CPU cores to Kubernetes



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

Uber



Uber

Uber



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

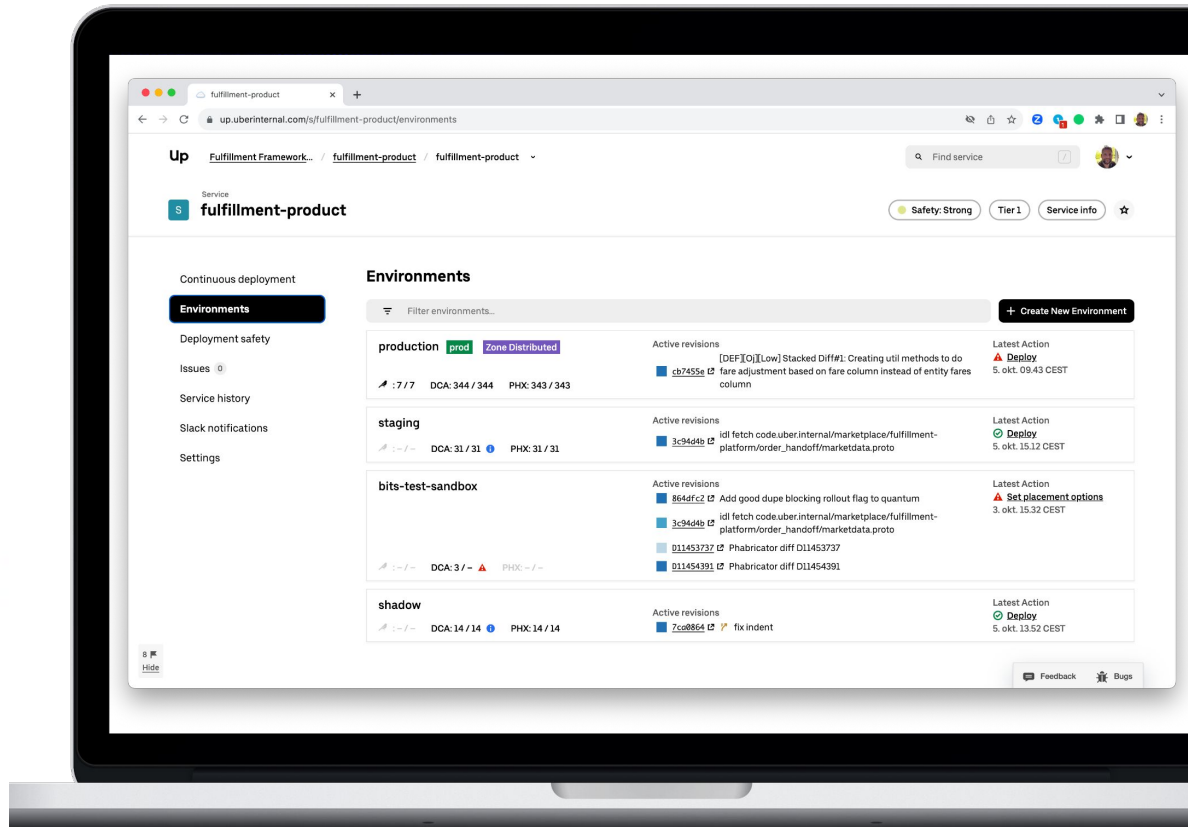
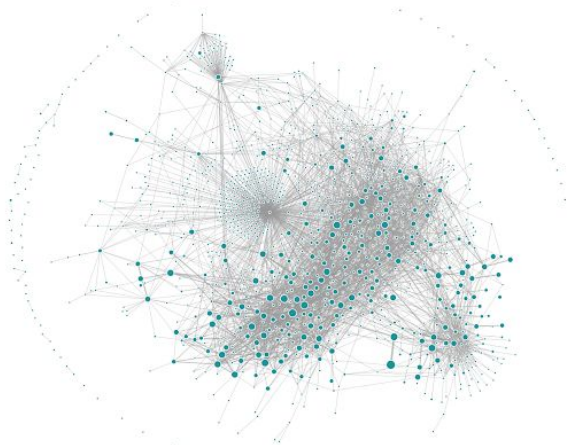


Global control plane

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

Horizontal scaleproduction

DCA

PHX

324

332

Canary

Turning on Canary will place 7 canary instances in DCA. Canary instances will be deployed before other instances.

	Instances	Cores	CPU Utilization	Cost
Cost preview	656	5576	24.49%	USD/Year

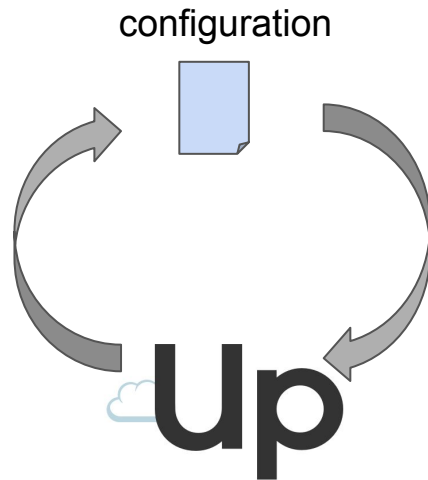
Rollout policy

Standard

Scale

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:

The screenshot shows the Uber Up dashboard for the 'lucy-test-service' in the 'production' environment. The interface includes a sidebar with navigation links, a main content area with deployment details, and a history panel on the right.

Deployment

Environment: **production** (Prod)

Safety: Unsafe | Tier: 5 | Service info

Deployment actions: Deploy, Hotfix, Restart

Active revisions: [Show details](#)

54cbc71 (BREAKGLASS) Revert [Go-code] sdul: re-enable upprivate and ulview linters'

Environment	Instances	Running	Requested
CANARY	3	3	3
DCA	2	2	2
PHX	2	2	2

Ongoing Operations

There are currently no ongoing operations. Details about each ongoing operation will be displayed here.

Sandboxes

No sandboxes exist for this environment.

[+ Create New Sandbox](#)

History (1-6 of 139)

- Deploy 718e974 → 54cbc71 (continuous-deployment) 3 Nov, 08:36 CET
- Deploy 81be8d9 → 718e974 (continuous-deployment) 3 Nov, 05:19 CET
- Deploy bf45d02 → 81be8d9 (continuous-deployment) 3 Nov, 03:47 CET
- Deploy bc5dbf → bf45d02 (continuous-deployment) 3 Nov, 01:52 CET
- Deploy 39523f1 → bc5dbf (continuous-deployment) 2 Nov, 22:10 CET

[Update addon](#) lucysweet 2 Nov, 16:29 CET

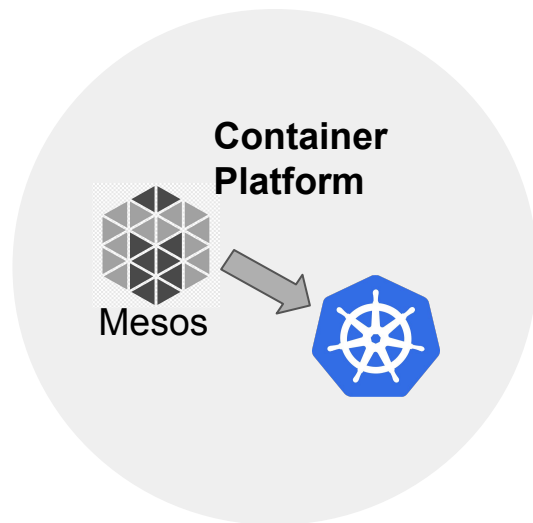
Migration to Kubernetes

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

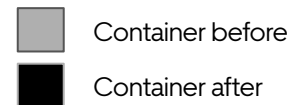
Decoupling services from the underlying physical infrastructure and scheduling software

Touchless migration

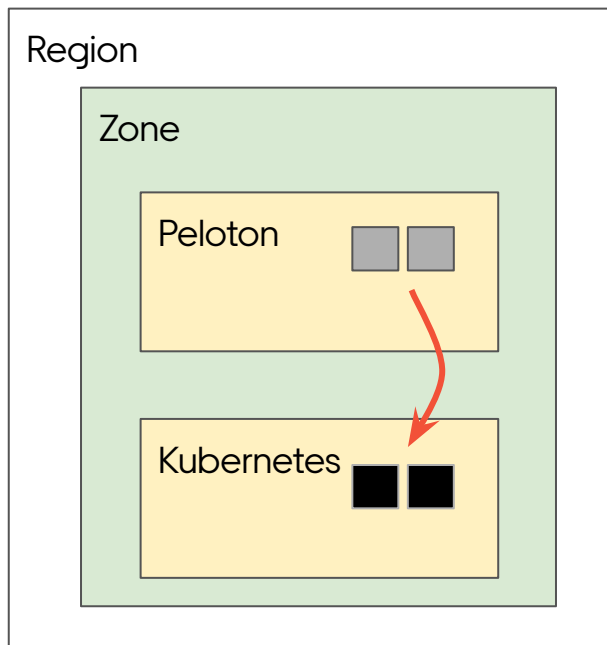
Zero downtime

Continuous migration

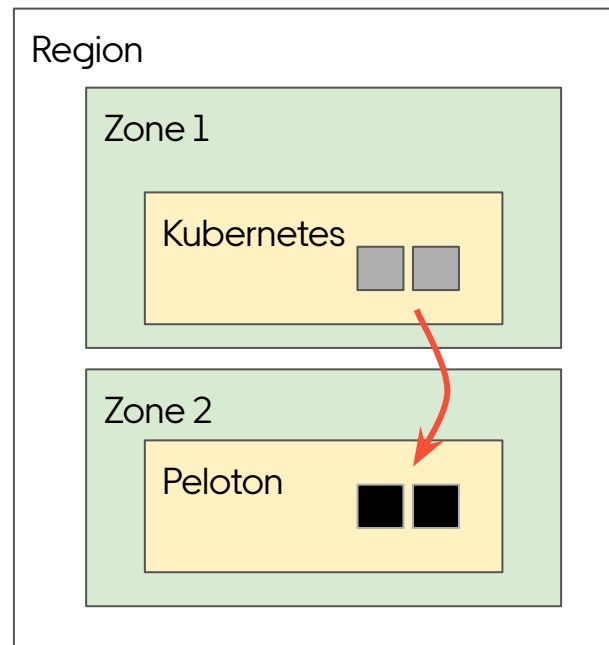
Portability - Movement tolerance



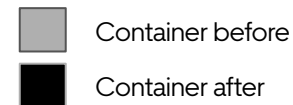
Onboarding to Kubernetes



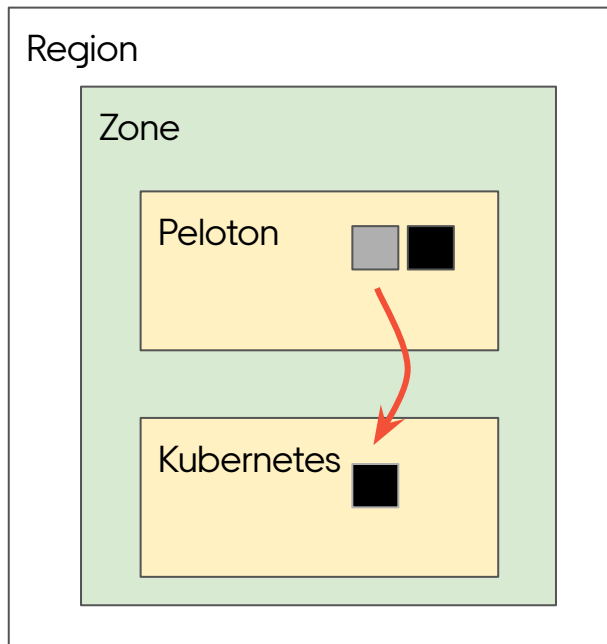
Offboarding from Kubernetes



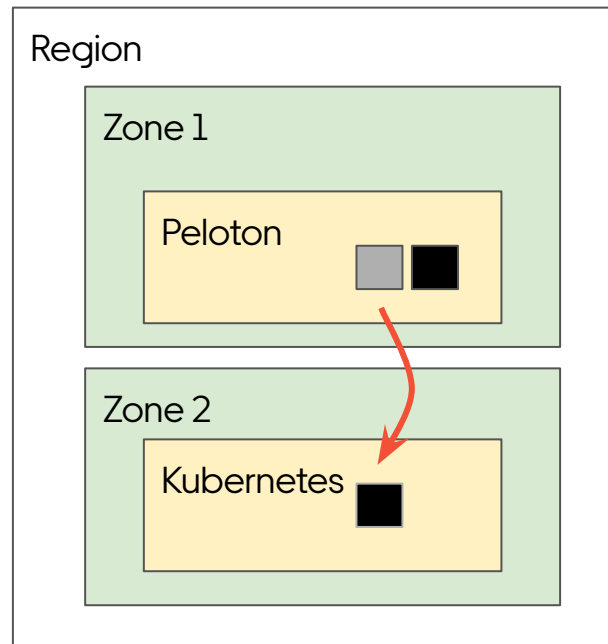
Portability - Distributability



Across clusters

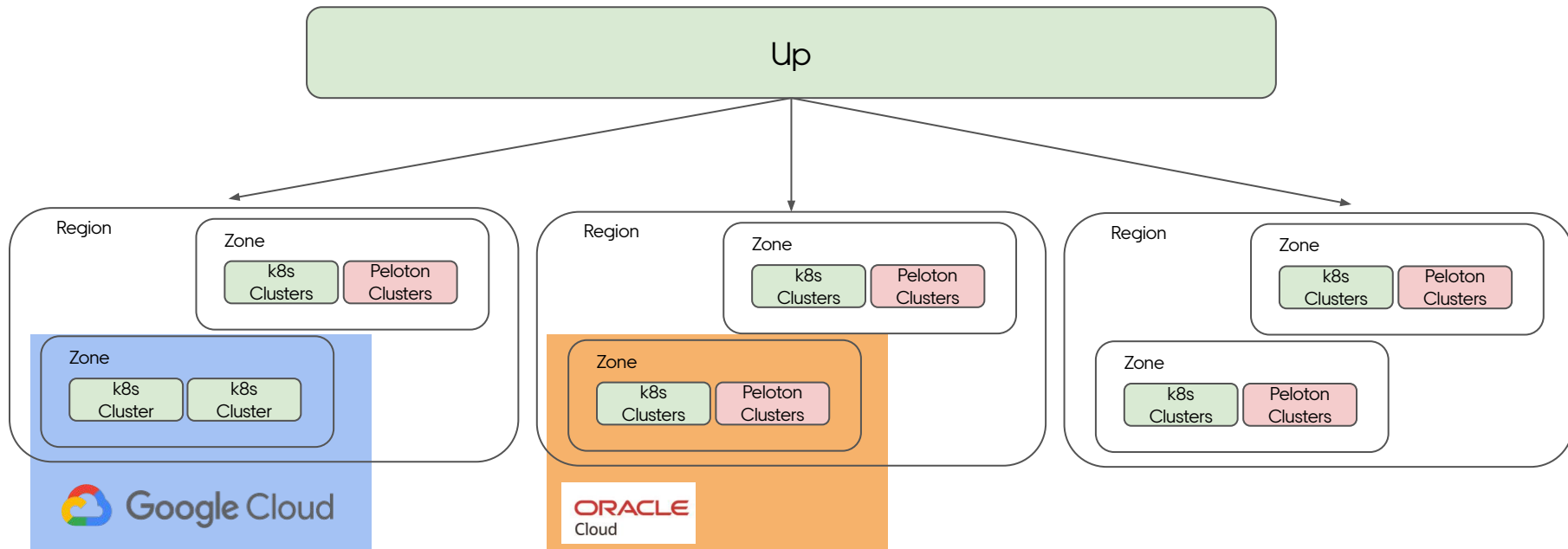


Across zones

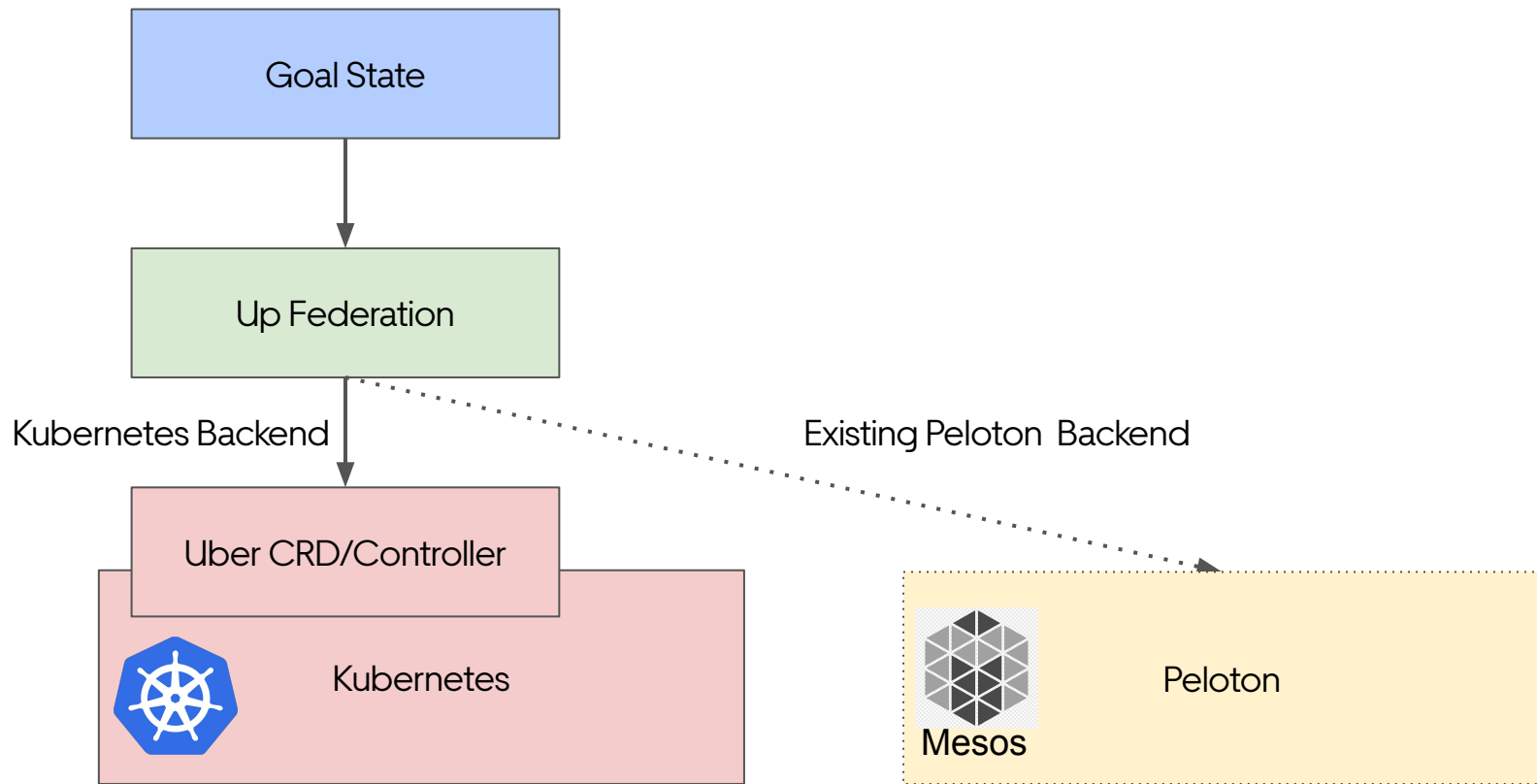


Our Kubernetes Fleet

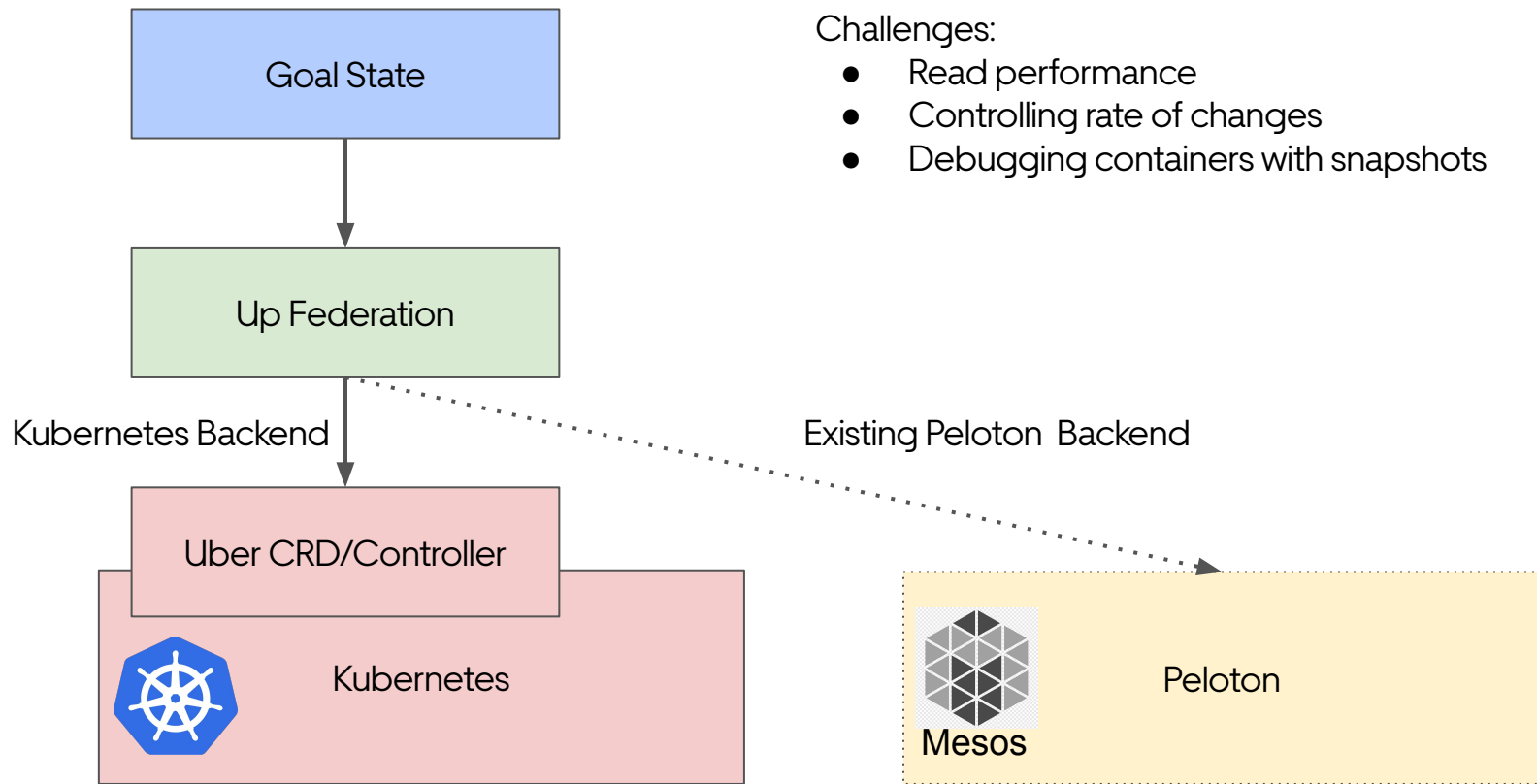
- 19 clusters in 4 “zones” and 2 “regions”
- 18,000 hosts / 2 million CPU cores
- Mix of CPU arches (amd64/arm64) and GPUs (RTX5000/etc)
- On prem and public cloud providers



Rebuilding Up in Kubernetes



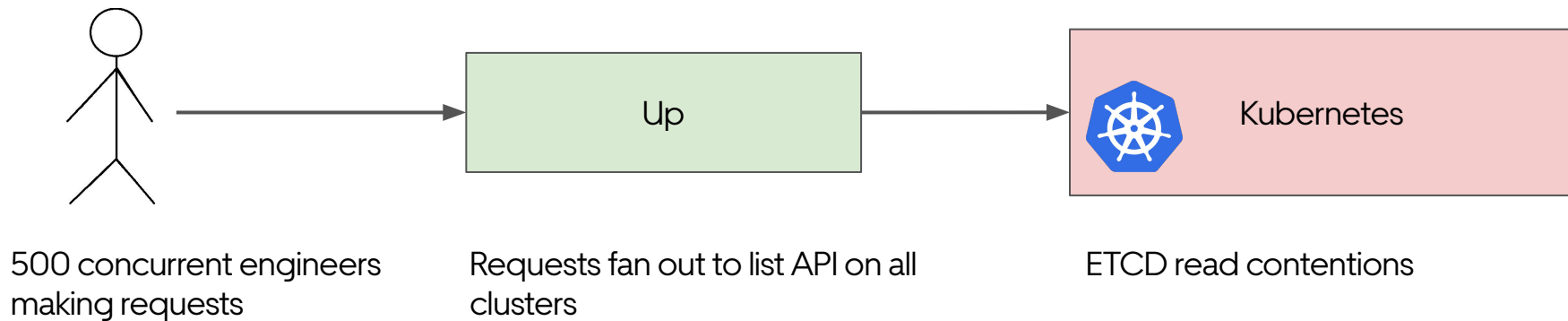
Rebuilding Up in Kubernetes



Challenges:

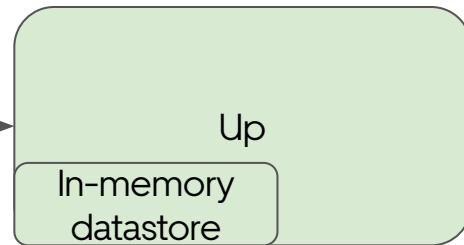
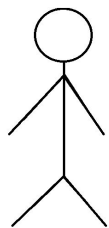
- Read performance
- Controlling rate of changes
- Debugging containers with snapshots

Read Performance



Read Performance

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



Informers on every Kubernetes cluster listen for updates



Kubernetes API server

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 restart 1 instances	11:47→11:48	11:48→11:49
	uMonitor — Monitoring 5 alerts ⓘ Kubernetes — Done (1 of 1) UNS — instances are present in UNS	00:17	
dca23			
11:49 → 11:51	dca23-kubernetes-stateless01 restart 2 instances	11:49→11:51	
	uMonitor — Monitoring 6 alerts ⓘ Waited Kubernetes — Done (2 of 2) UNS — instances are present in UNS	01:12 00:18	
phx6			
11:51 → 11:53	phx6-kubernetes-stateless01 restart 2 instances	11:51→11:53	
	uMonitor — Monitoring 6 alerts ⓘ Wait Upgrading compute job: up.lucy-test-service.production.000000000000000000000000 Kubernetes — 0.compute-0.f8fac343-95ef-4935-ad48-4e882a2c9d8c (0 of 2) UNS	02:00 00:00	

Controlling rate of changes

12:29 [] 12:30			
<div>  dca20 </div> <div>12:29 → 12:38</div>	<div> dca20-kubernetes-stateless01 update 1 instances </div> <div>12:29→12:30</div> <div>  uMonitor — Monitoring 5 alerts  </div> <div>  Kubernetes — Done (1 of 1) </div> <div>  UNS — instances are present in UNS </div> <div>00:13</div>	<div> dca20-kubernetes-stateless01 update 2 instances </div> <div>12:30→12:32</div> <div>  uMonitor — Monitoring 5 alerts  </div> <div>  Waited </div> <div>  Kubernetes — Done (2 of 2) </div> <div>  UNS — instances are present in UNS </div> <div>02:00</div>	00:09
	<div> dca20-kubernetes-stateless01 rollback update 1 instances </div> <div>12:38→12:38</div> <div>  Kubernetes — Done (1 of 1) </div> <div>  UNS — instances are present in UNS </div> <div>00:07</div>	<div> dca20-kubernetes-stateless01 rollback update 2 instances </div> <div>12:38→12:38</div> <div>  Kubernetes — Done (2 of 2) </div> <div>  UNS — instances are present in UNS </div> <div>00:09</div>	
<div>  dca23 </div> <div>12:32 → 12:38</div>	<div> dca23-kubernetes-stateless01 update 2 instances </div> <div>12:32→12:34</div> <div>  uMonitor — Monitoring 6 alerts  </div> <div>  Waited </div> <div>  Kubernetes — Done (2 of 2) </div> <div>  UNS — instances are present in UNS </div> <div>02:00</div>	<div> dca23-kubernetes-stateless01 rollback update 2 instances </div> <div>12:38→12:38</div> <div>  Kubernetes — Done </div> <div>  UNS — instances are present in UNS </div> <div>00:05</div>	00:18
<div>  phx6 </div> <div>12:34 → 12:38</div>	<div> phx6-kubernetes-stateless01 update 2 instances </div> <div>12:34→12:38</div> <div>  uMonitor — Monitoring 6 alerts  </div> <div>  Waited </div> <div>  Kubernetes — Done (2 of 2) </div> <div>  UNS — instances are present in UNS </div> <div>02:00</div>	<div> phx6-kubernetes-stateless01 rollback update 2 instances </div> <div>12:38→12:38</div> <div>  Kubernetes — Done (2 of 2) </div> <div>  UNS — instances are present in UNS </div> <div>00:11</div>	01:21

Container snapshots

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

33554433 Service crashed while starting dca dca94 dca94-qc 1dfea8f 56 seconds ago

What happened

The instance crashed before reporting healthy. This is likely due to an issue with the service code preventing startup.

Timestamp 2023-11-01 13:27 GMT+1

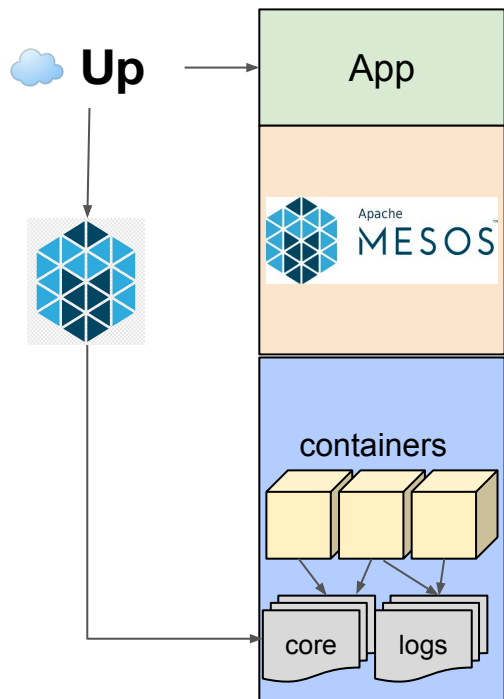
Build bkt1-produ-1682337112-420f2

Inspect logs

- Instance logs
- Instance console logs
- Container snapshot** (highlighted by a red arrow)

- We want to continue to support this into Kubernetes

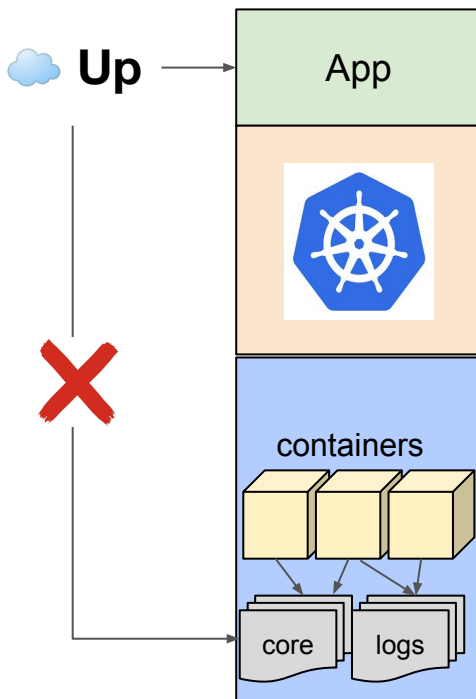
Container snapshots



Peloton

Container artifacts (core, heap, logs) not cleaned up on exit.

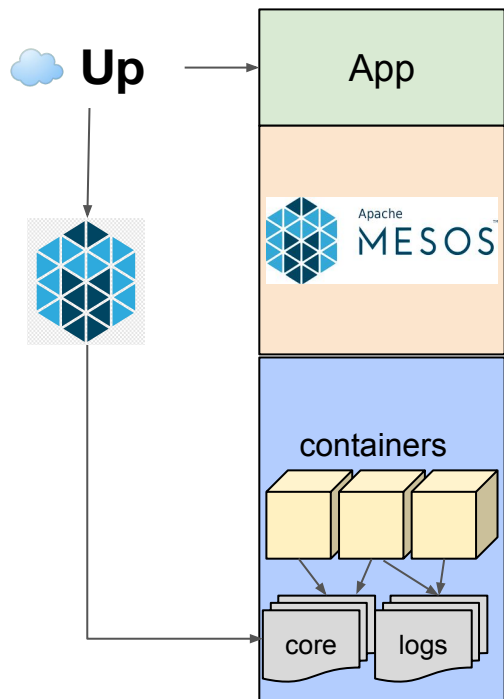
Accessible on Up via Mesos.



Native Kubernetes

Container artifacts are deleted on container exit.

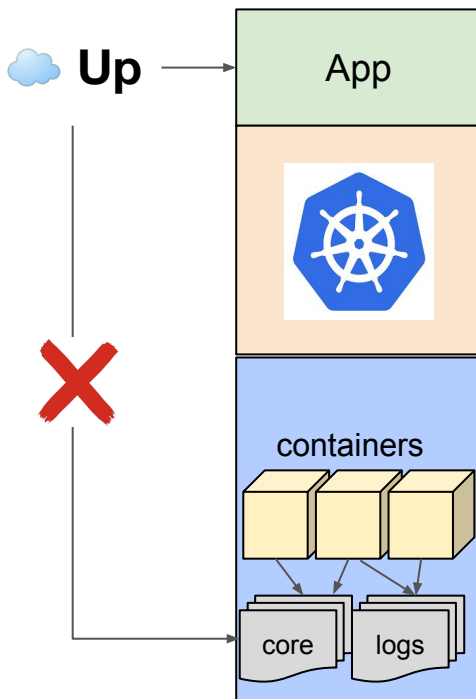
Container snapshots



Peloton

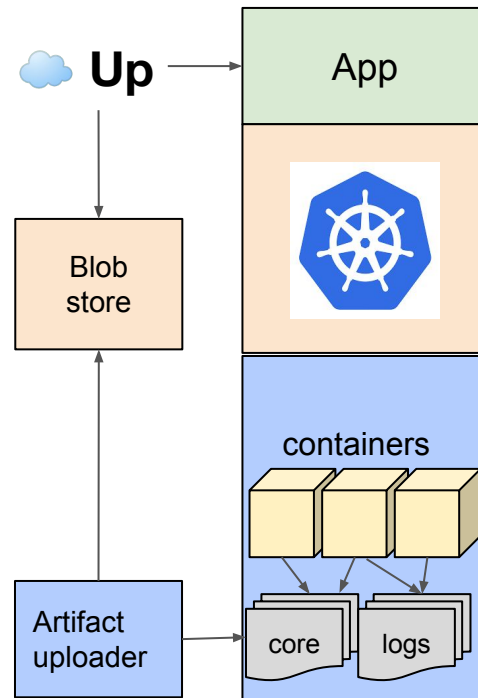
Container artifacts (core, heap, logs) not cleaned up on exit.
Accessible on Up via Mesos.

Uber | Kubernetes @ Uber



Native Kubernetes

Container artifacts are deleted on container exit.

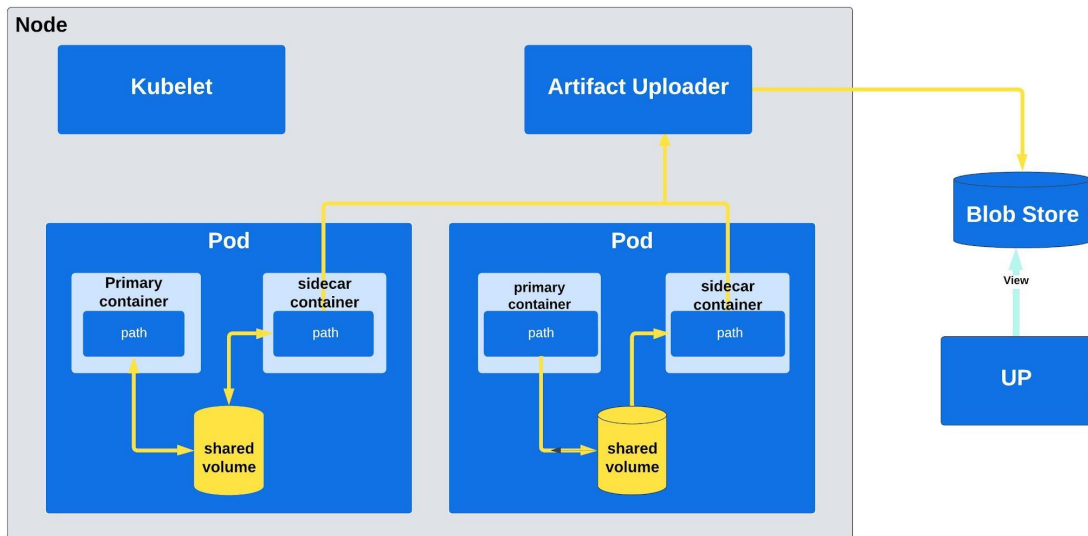


Uber Kubernetes Engine

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
<div>access-decider-production-compute-0-1325500264-2023-10-30_20-43-57.tar</div>	68.0M	31 Oct 2023, 04:55

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

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

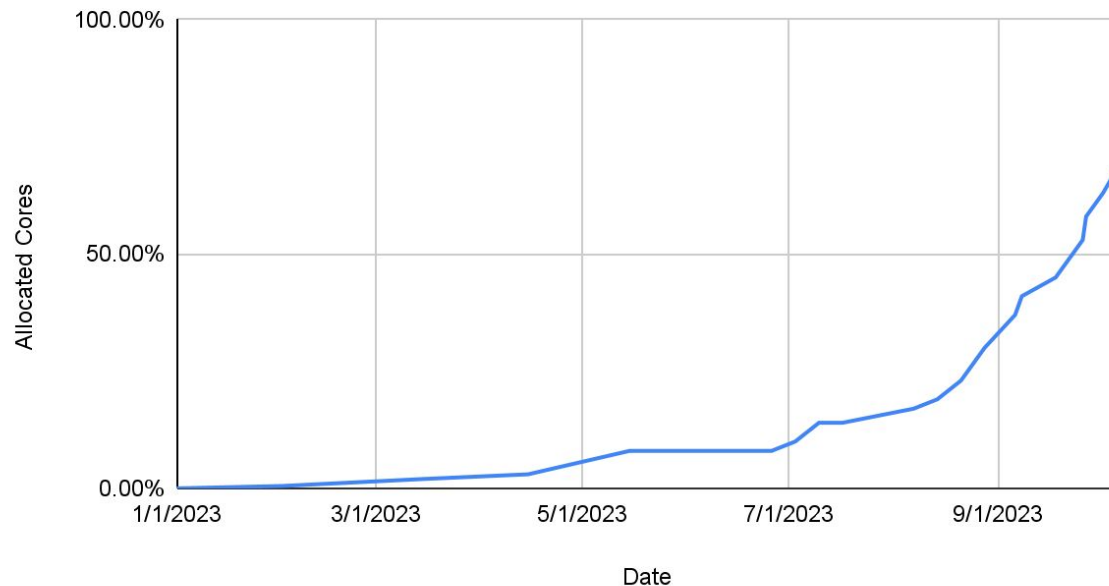
- We found lots of our users had built a dependence in their service on behaviours on our old system, Peloton
- Some of these behaviours weren't even documented, and some of the dependencies weren't either
- Our ability to slowly move services to Kubernetes or Peloton allowed us to acceptance test services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

Kube Stateless Migration over Time



Next steps...

STATELESS



BATCH



Michelangelo



Workbench



PIPER



presto



Apache Flink



APACHE
Spark

STATEFUL



cassandra



MySQL



redis



kafka

DAEMONSET

M3
Collector

Service
Mesh



Uber

[Blog Post](#)