



WOMEN IN INFRASTRUCTURE

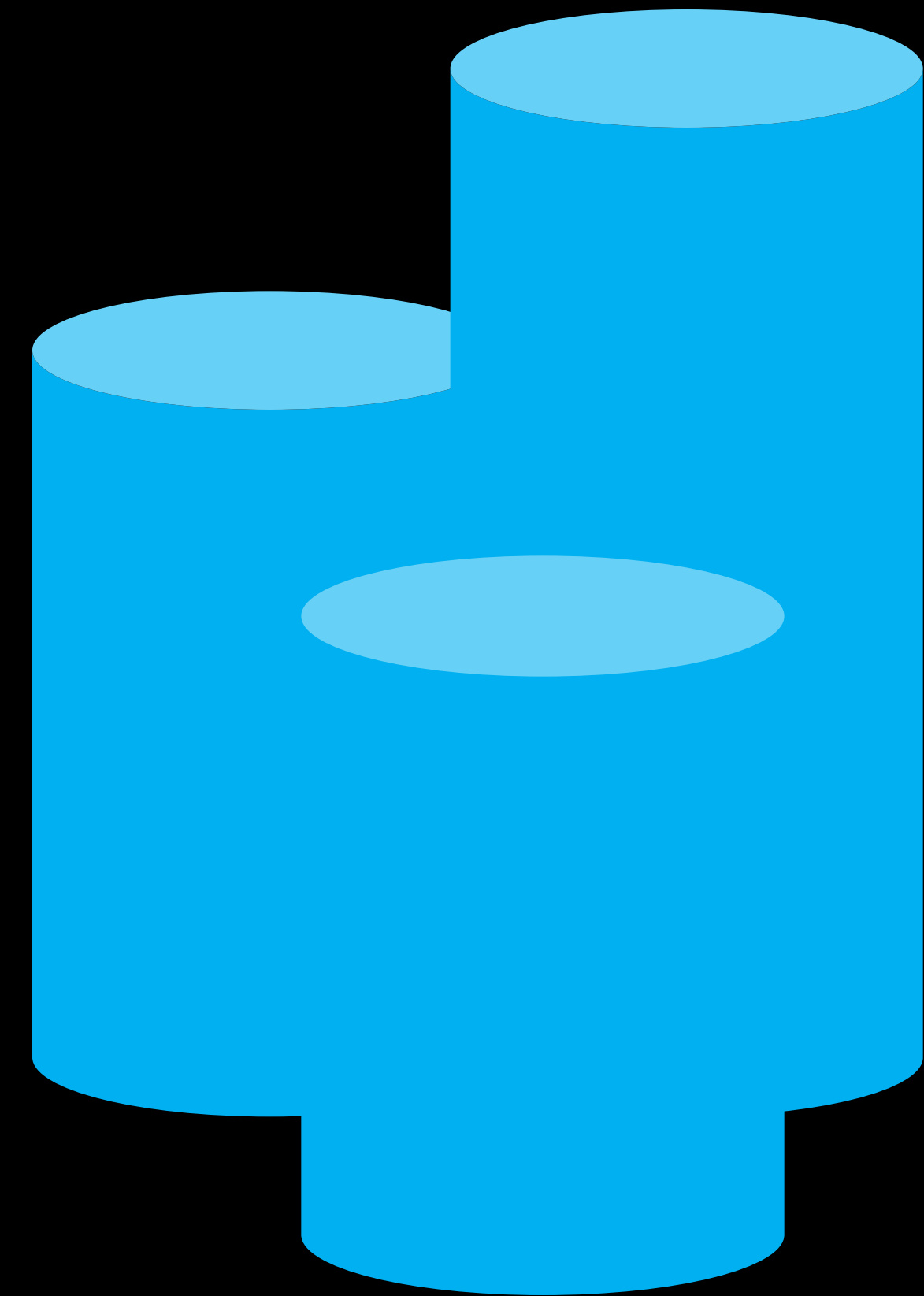
DATA-INTENSIVE WORKLOADS ON KUBERNETES

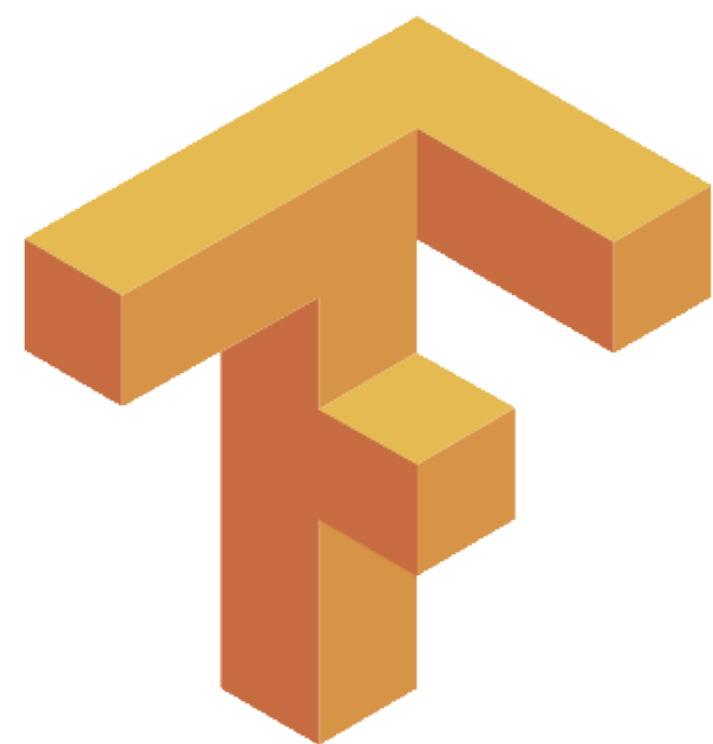
ALENA HALL
@LENADROID

SEATTLE, WASHINGTON
NOVEMBER 8, 2018

Data Systems

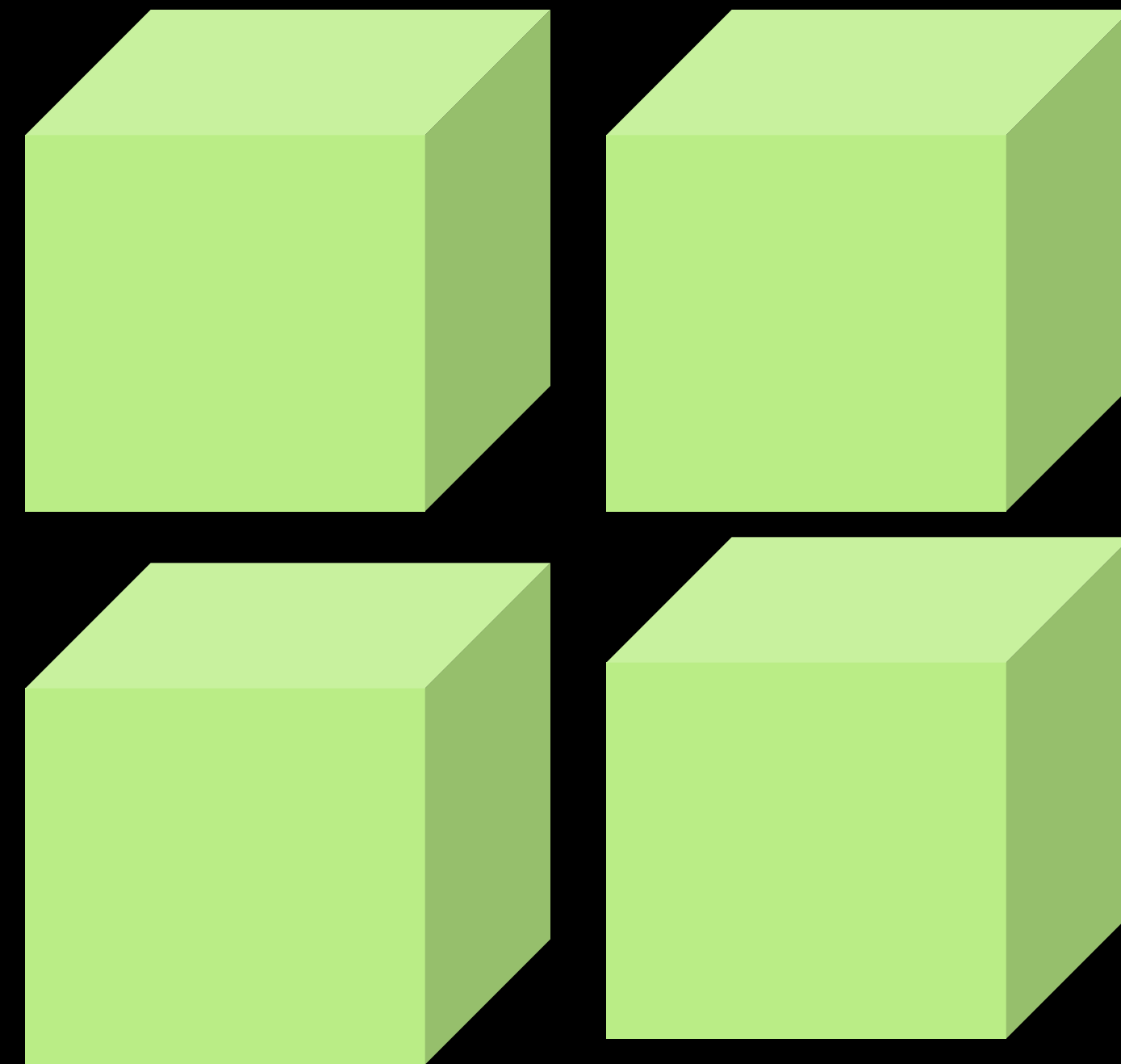
- Database
- Cache
- Stream processing system
- Any system that works with data

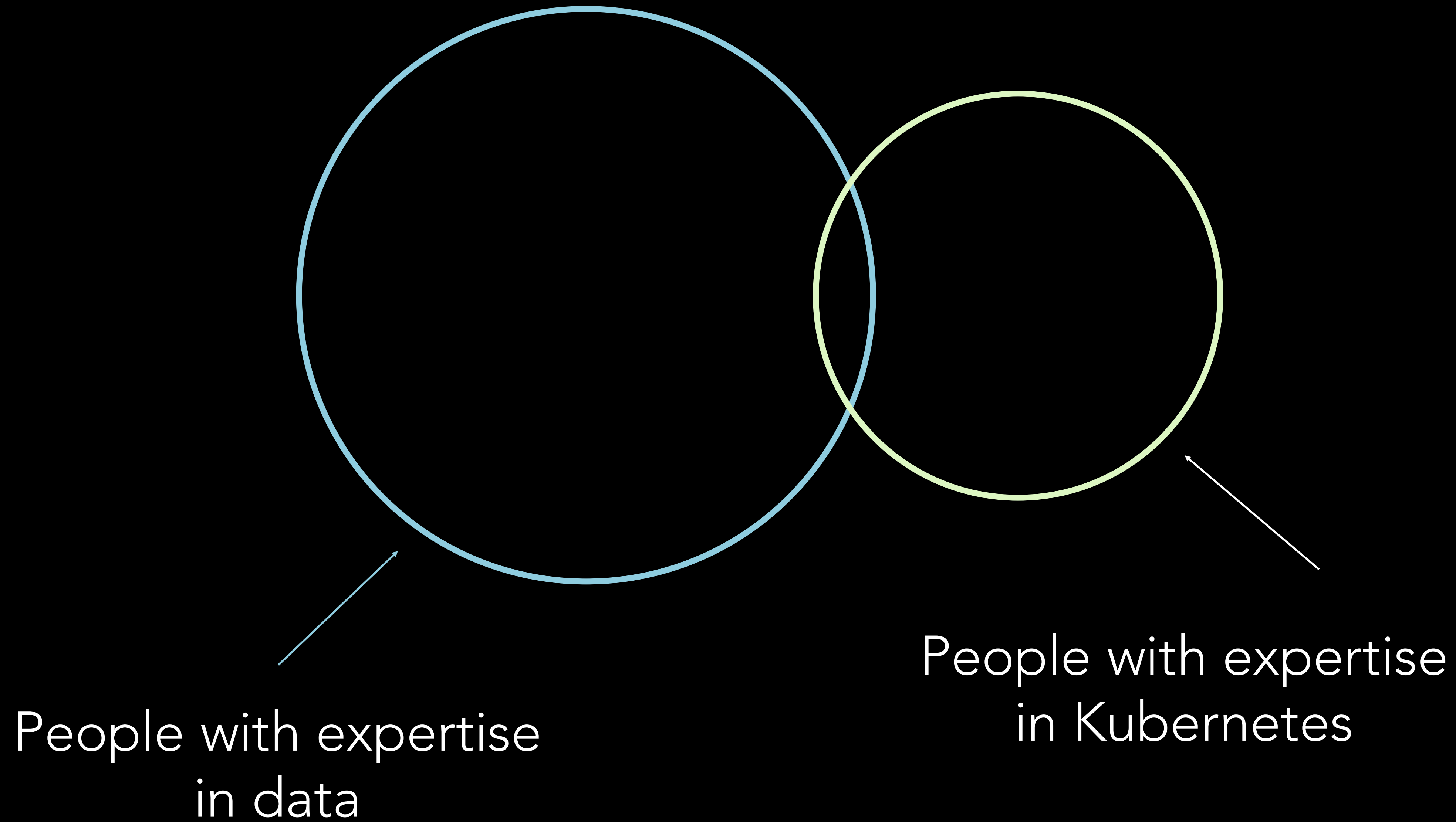


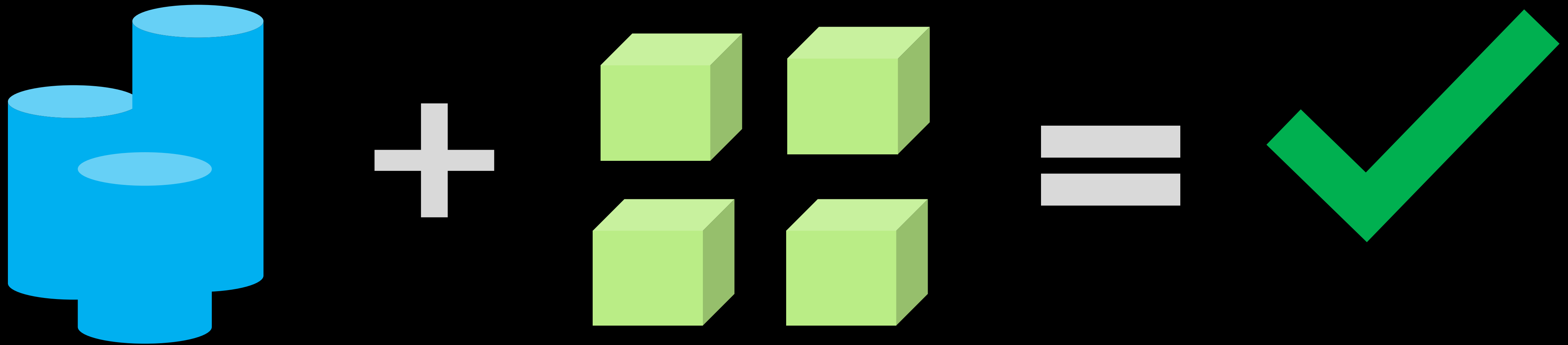


Kubernetes

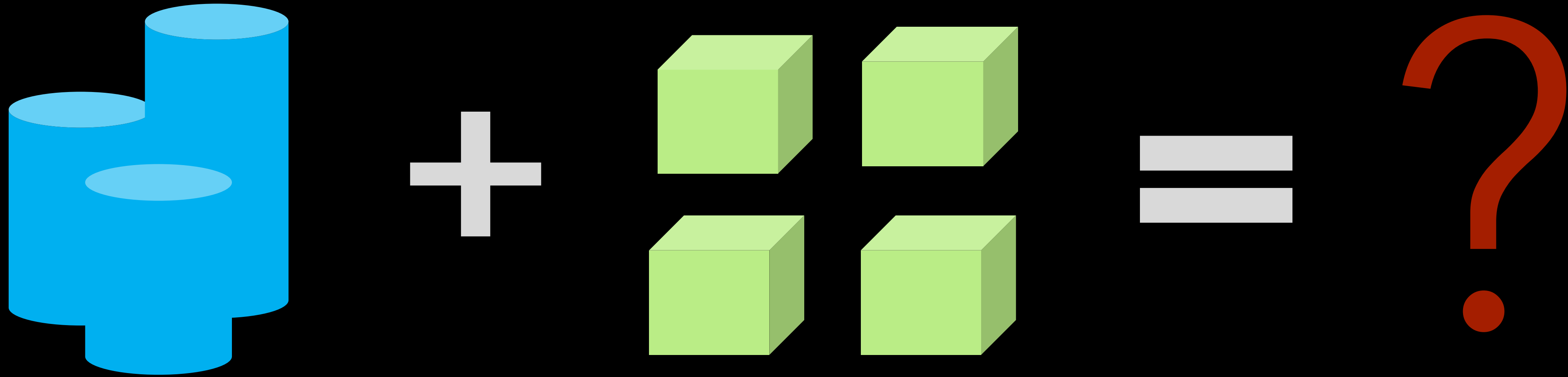
- Entire ecosystem of projects
- Known by "container people" or infrastructure people





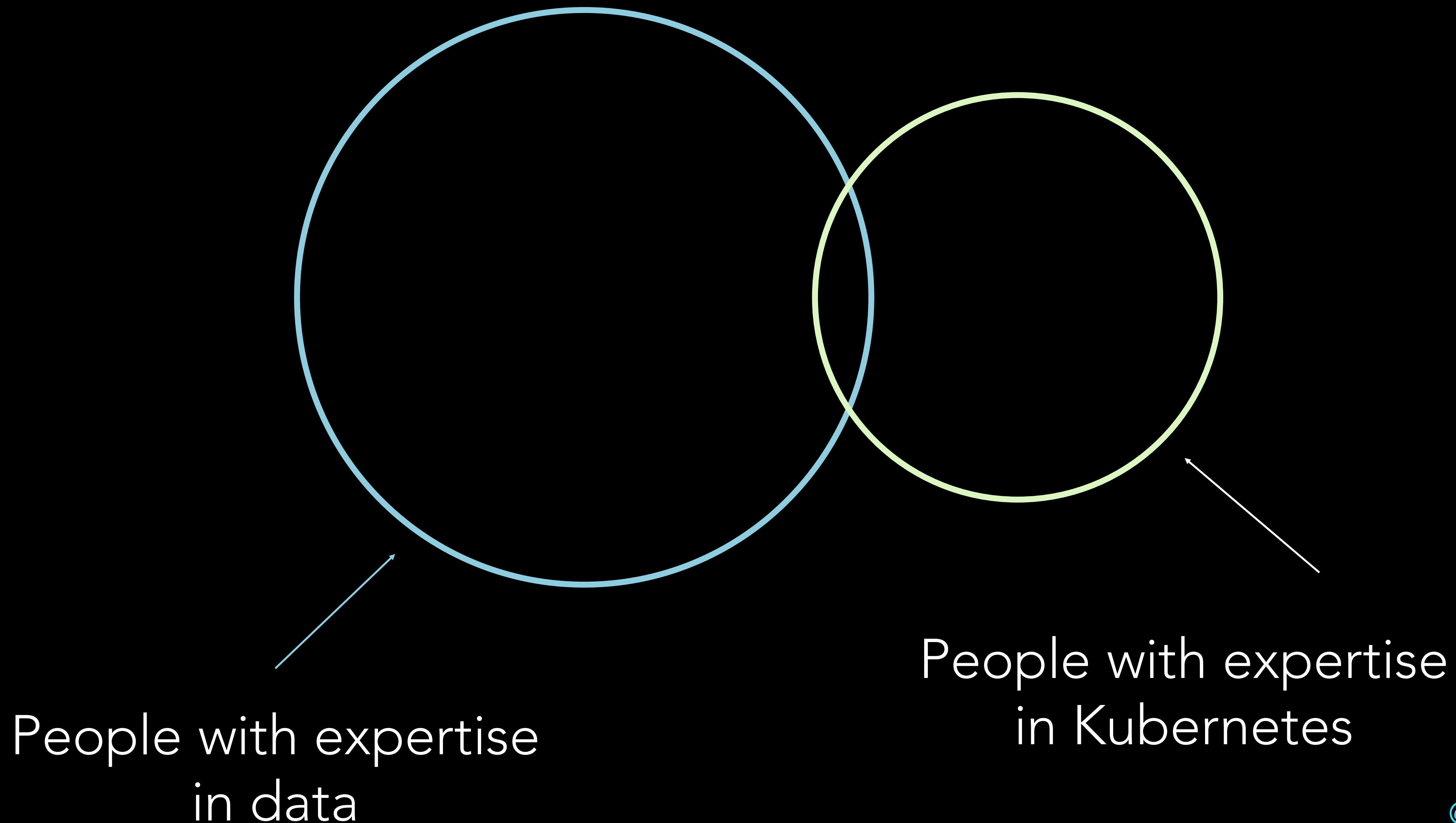


- ✓ Organizations who already use Kubernetes for many services and applications
- ✓ Organizations who can afford resources and time to troubleshoot possible networking and storage questions



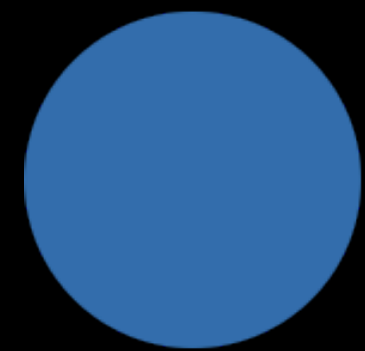
- ✓ Organizations who aren't actively using Kubernetes
- ✓ Organizations with no time/resources for troubleshooting
- ✓ Organizations with projects that require stability and can't yet achieve fully correct operation when running on top of Kubernetes

How to be successful?



THERE'S A BIG DIFFERENCE
BETWEEN RUNNING A
STATELESS MICRO-SERVICE
ON KUBERNETES AND
RUNNING A DISTRIBUTED
STATEFUL SYSTEM OR A
DATABASE.

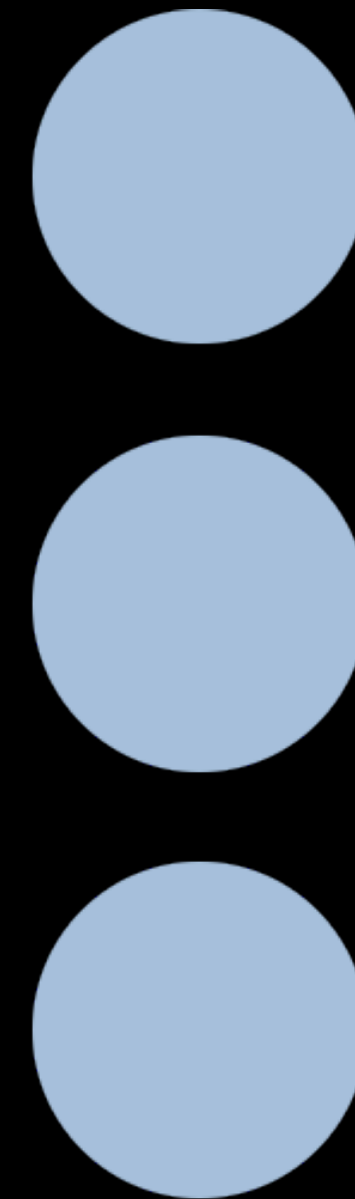
WHAT CAN WE DO WITH
BUILT-IN OBJECTS?



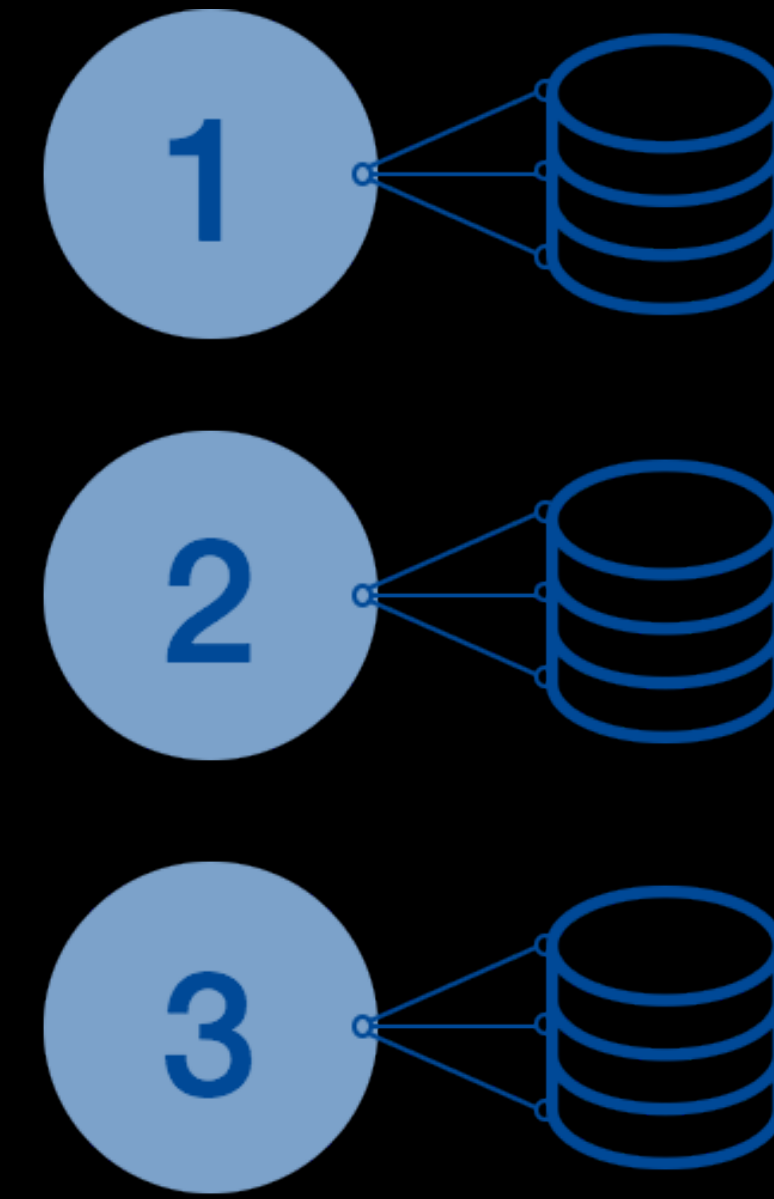
Pod



Job



Deployment



Stateful Set

and more...

SO DATA SYSTEMS MEANS
STATEFUL SETS? *

* NOT REALLY

When new data people discover Kubernetes they see stateful sets, and it might seem like a solution to running data systems.

But is it enough for all data intensive systems?

What if none of primitive Kubernetes types
fully work for our systems?

PICKY

DISTRIBUTED SYSTEMS

UP AND RUNNING
!=
OPERATING CORRECTLY

Things that need special care

Things like Stateful Sets can be powerful, but they alone have no idea what's different between e.g. Kafka and Cassandra.

They don't know if they should manage things differently. Unless we teach Kubernetes how to treat our systems. We still need to take nuances specific to the stateful application under consideration.

Bad news: built-in Kubernetes objects might not be fully sufficient for running more complex distributed data systems.

Good news: we can still run those systems on Kubernetes correctly, because there is a way to do so with custom controllers and custom resource definitions.

IF WE TEACH KUBERNETES
– IT WILL LEARN!

CUSTOM RESOURCE DEFINITIONS

(CRDs)

A custom resource is an extension of the Kubernetes API that is not necessarily available on every Kubernetes cluster, that can be added.

After it's added, users can work with objects of this resource, just as they do for built-in resources like pods.

Think of CRDs as new Kubernetes entities, new data types that we define, that become a part of the Kubernetes API.

CUSTOM CONTROLLERS

When we combine a Custom Resource Definition with a Controller – it becomes truly powerful, because it allows us to declare what is the desired state of your resource.

OPERATOR

CRD + CUSTOM CONTROLLER

Operators are important for engineers who want to benefit from running and managing their distributed systems reliably, consistently and correctly (whatever this means), when built-in Kubernetes resources aren't enough.

EXAMPLE

RUNNING A TENSORFLOW JOB WITH
KUBEFLOW AND TF-JOB OPERATOR
ON AZURE KUBERNETES SERVICE



BEHIND THE SCENES

TF-OPERATOR

What does this mean for us?

For solution architects and engineers, knowing which built-in Kubernetes objects exist and how operators work will help make decisions whether Kubernetes is the right choice for their distributed systems.

They will also be able to determine what “Kubernetes supports X” really means? They’ll understand whether it means *real* Kubernetes support where the system is behaving correctly, or it is just able to run in containers without preserving its initial guarantees.

For Open Source engineers and creators of distributed and stateful systems, understanding operators will help determine what knowledge Kubernetes already has, and what knowledge it lacks and needs to be taught about their systems to manage them correctly.

WHY THIS IS IMPORTANT

Kubernetes is on its way to become a truly extensible system for running other systems and applications.

It runs on any cloud and on premises and allows us to add new and missing functionality.

With these powerful features it might very well become a platform where we run most of our systems sometime in the future.

Open-service broker could act as a gateway for managing services and resources across cloud.

And operators could be a beginning of something that we might call a managed service, but not tied to any cloud.

Majority of operators are in extremely early stage now!

However, we already start seeing major progress.

There are new operators coming out frequently, and while most of them are in alpha state, there's a lot of work in progress.



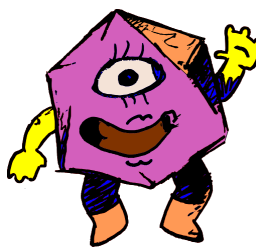

WHAT CAN THIS MEAN
FOR THE CLOUD?

It could mean that in the future the choice of cloud platform might not be based on what services it supports.

Because we would be able to run all of them on Kubernetes on top of any cloud, what might become more important is underlying storage and networking infrastructure that it runs on, the speed, cost, and reliability of underlying things would really matter.

Alena Hall - **lenadroid**



- ✓ Works on Azure at  Microsoft
- ✓ Lives in  Seattle
- ✓ F# Software Foundation Board of Trustees
- ✓ Organizes [@ML4ALL](https://twitter.com/ML4ALL) 
- ✓ Program Committee for Lambda World
- ✓ Has a channel: **You** **Tube** /c/**AlenaHall**

THANK YOU!