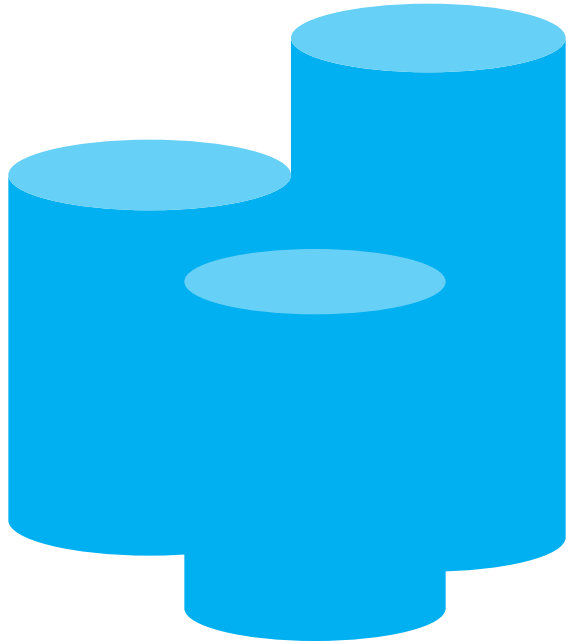
Demystifying Data-Intensive Systems on Kubernetes

Lena Hall

@lenadroid

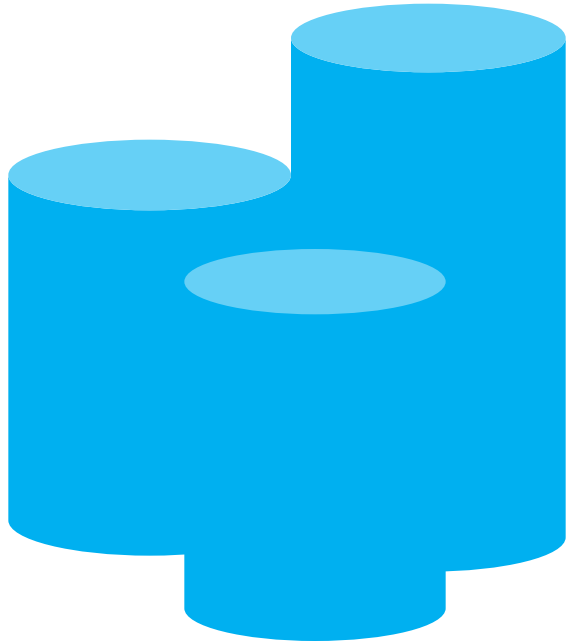
Data-Intensive Systems

- Databases
- Caches
- Stream-processing systems
- Any system that works with data



Data-Intensive Systems

- Databases
- Caches
- Stream-processing systems
- Any system that works with data



… or other non-trivial, **not necessarily stateful** systems


cassandra


kafka


Spark


elastic


redis

Big Difference

Stateless microservice

vs

Distributed stateful system, or other non-trivial system

on Kubernetes

Why should you care

- Software Engineers and Solution Architects
- Engineering Managers and CTOs
- Maintainers and Contributors of Distributed Systems

Talk Spoilers

- **DECISIONS**
  Making Decisions– is Kubernetes a good choice
- **SOLUTIONS**
  Current state, challenges, and solutions
- **FUTURE**
  The Future of systems on Kubernetes

Decision Making

Decision (p1, p2, p3, …) = Yes | No

p1 = required guarantees
p2 = existing skills and resources
p3 = acceptable risks
pN = …

Decision Making Best Practices

- ? What are downsides and challenges of your current environment for running your system
- ? What problems will switch to Kubernetes solve
- ? What new problems will it create
- ? How big will increase/decrease in costs be
- ? What team or process changes will need to happen

Decision Making Variables

For example:

- Ability to afford **resources/time** to troubleshoot issues
- Requirement to be **independent** from a cloud provider or environment
- Consistency/performance/availability/other **guarantees**
- Readiness to accept **possible risks**

... many more

Examples of motivation and challenges

Common motivation

Examples:

- Workload portability
- Convenience of deployment, operations, automation
- Independence from a cloud provider or environment
- Open and rapidly developing, rich ecosystem
- Flexibility and cost savings
- Faster start-up times
- Extensible and open API
- To benefit from existing Kubernetes infrastructure

Common challenges

Examples:

- Limited options for running certain workloads
- Limited examples of production-ready architectures
- Limited time, or resources to support the system
- Limited functionality/integrations for storage/networking
- Stability, reliability, etc. of existing solutions
- Need to build the solution almost from scratch
- Need to gain new skills to troubleshoot new environment

Decision Making Best Practices

How to determine possible risks and challenges?

Decision Making Best Practices

How to determine possible risks and challenges?

Understand what Kubernetes *can* or *can't* do.
What it *is* or *isn't* responsible for.

Decision Making Best Practices

How to determine if Kubernetes can satisfy guarantees required by my system?

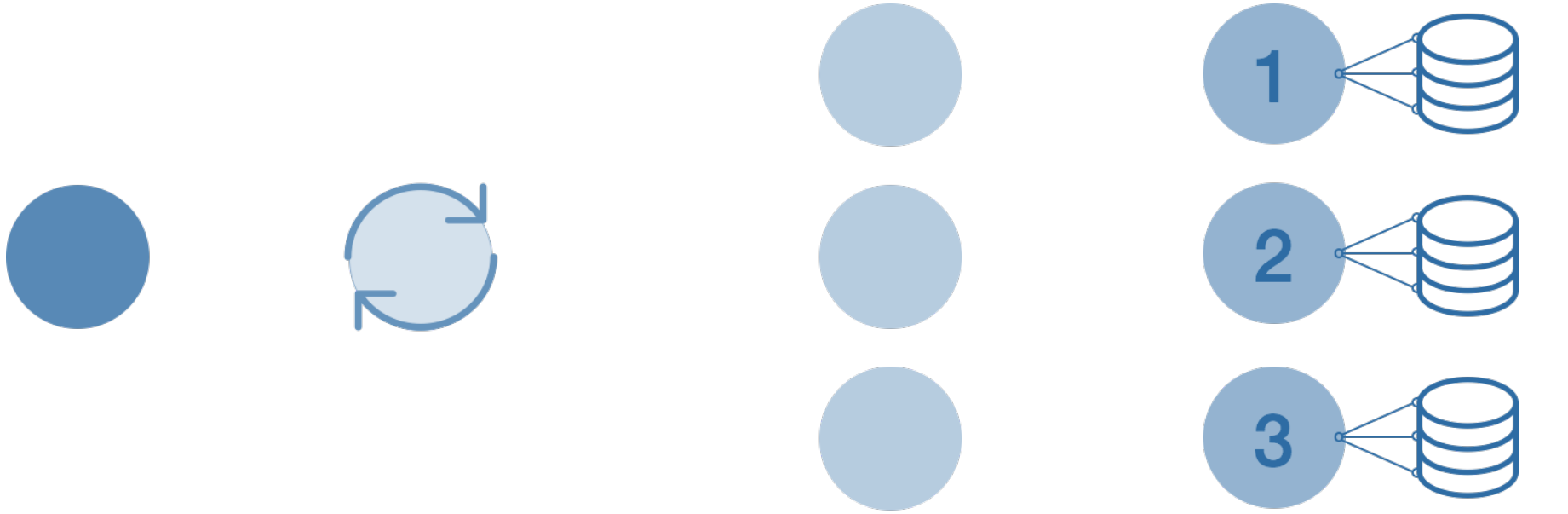
Decision Making Best Practices

How to determine if Kubernetes can satisfy guarantees required by my system?

Learn what abstractions and instruments Kubernetes and its API have to guarantee or implement your system requirements.

To make the right decision is to understand how things work

What Kubernetes **can** and **can't** do with built-in objects



Pod Job Deployment Stateful Set

and more…

Stateful Systems == Stateful Sets?

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

Things that need special care

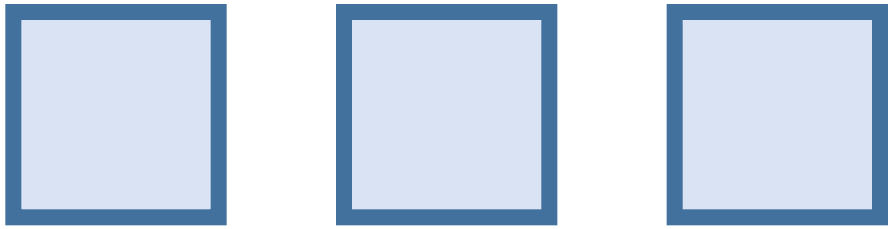
UP AND RUNNING

!=

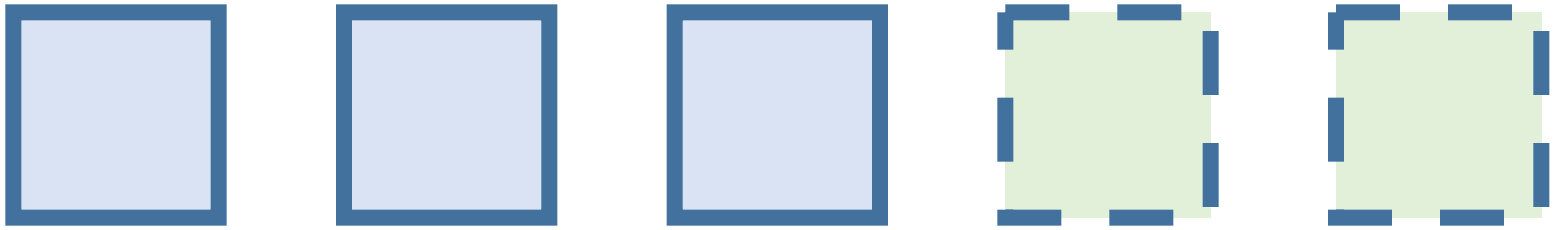
OPERATING CORRECTLY

Example: Scaling Clusters

From



To

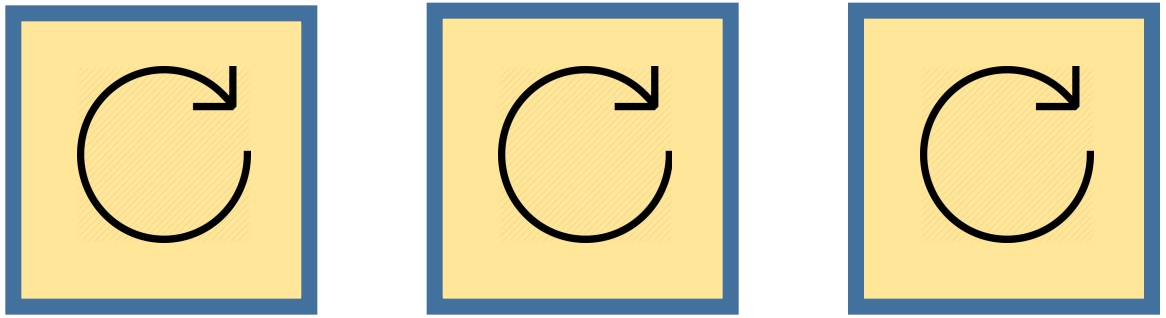


- ✓ Manually assign partitions to new nodes
- ✓ Rebalance data to maintain even load
- ✓ Apply cluster configuration to new nodes

Example: Safe Cluster Restarts

? Are there any under-replicated partitions?

? Is the cluster in healthy state?



✓ Restart one node at a time

✓ Wait for each node to catch up

Custom Resource Definitions

CRDs

Custom Controllers

OPERATOR

CRD + Custom Controller

Anatomy of an Operator

Controller

Queue that the Controller subscribes to

Informer

Resource Structs/Types

CRD

Example

Running a TensorFlow Job with KubeFlow and tf-job Operator on Azure Kubernetes Service

Behind the scenes

tf-operator

Ways to build an Operator

Example Operators

Apache Kafka/Confluent Operator

confluent.io/confluent-operator

Apache Cassandra Operator

github.com/instaclustr/cassandra-operator

TensorFlow Operator

github.com/kubeflow/tf-operator

Apache Flink Operator

osdir.com/apache-flink-development/msg09830.html

Apache Spark Operator

github.com/googlecloudplatform/spark-on-k8s-operator

Awesome Operators

/operator-framework/awesome-operators

Takeaways

Understand your goals, benefits, and challenges

Define “correct” operation for your system

Know when to use existing core Kubernetes types, and when to create custom resources

What's next?

Future

- More Operators
- Better Operators
- Easier to write Operators

Future

- Better workload portability
- More focus on writing your apps
- More automation

Future

- Multi-Cloud becomes reality
- More independence

Share your data-intensive scenarios

bit.ly/data-k8s

Resources

Lena's talk from GOTO Chicago 2018 - running a distributed database on Kubernetes, specifics of Stateful Sets, with examples of using Cassandra and Spark:

bit.ly/statefulsets-gotochgo

Lena's blog and other talks:

bit.ly/lena-blog and bit.ly/lena-talks

Lena Hall lenadroid

- ✓ Works on Azure at Microsoft
- ✓ Lives in Seattle
- ✓ F# Software Foundation Board of Trustees
- ✓ Organizes @ML4ALL
- ✓ Program Committee for Kafka Summit
- ✓ Has a channel: YouTube **/c/AlenaHall**


KubeCon
CloudNativeCon
North America 2018
Thank you!