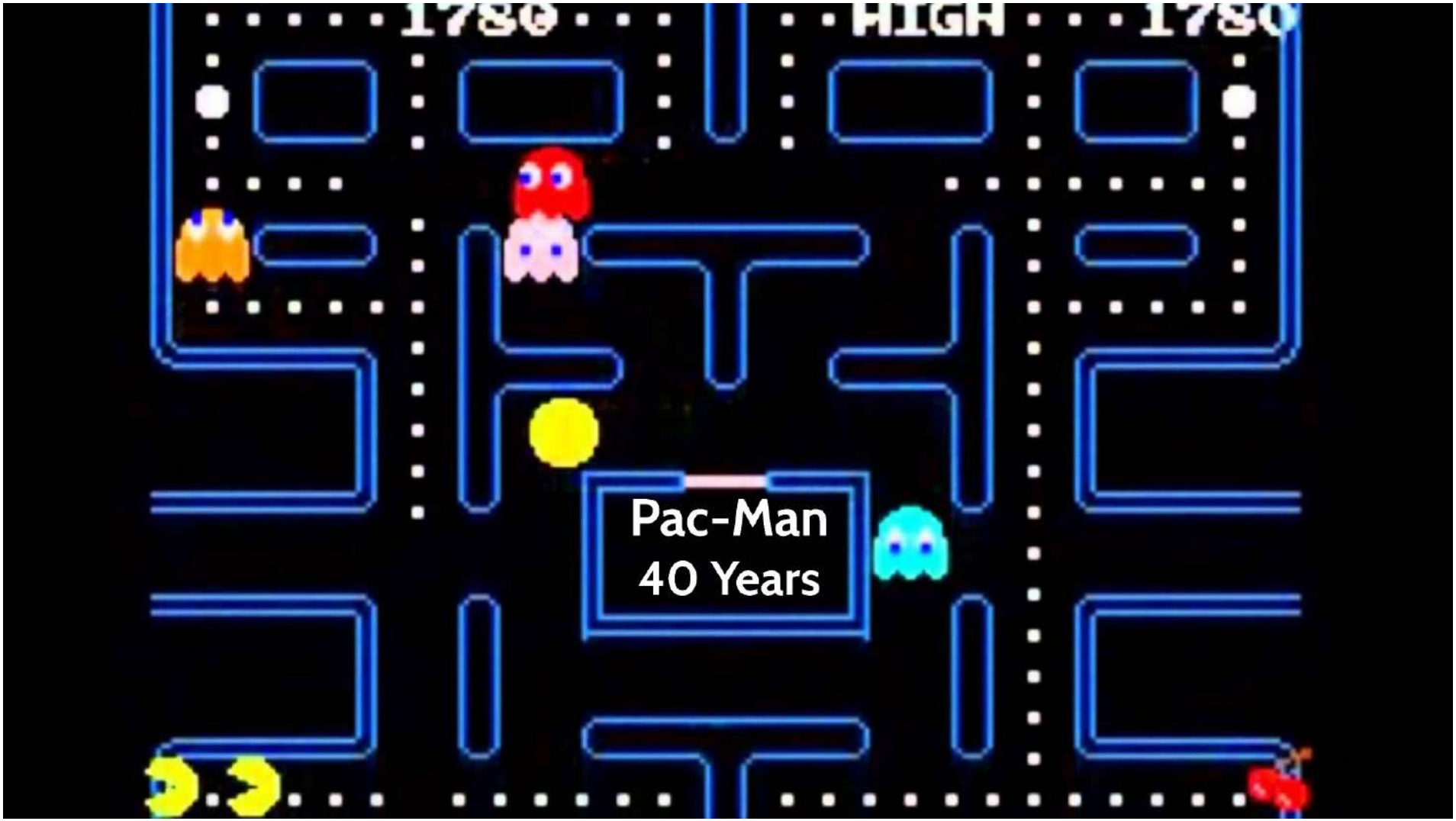
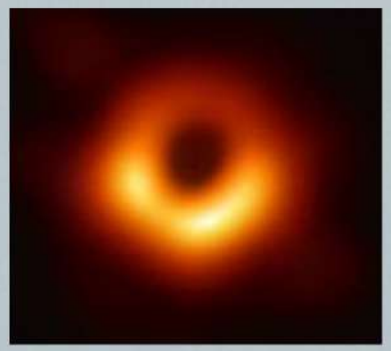
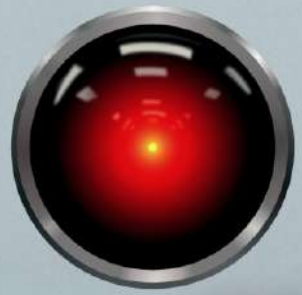

Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community


1780
HIGH
1780
Pac-Man
40 Years

Next 40 Years?

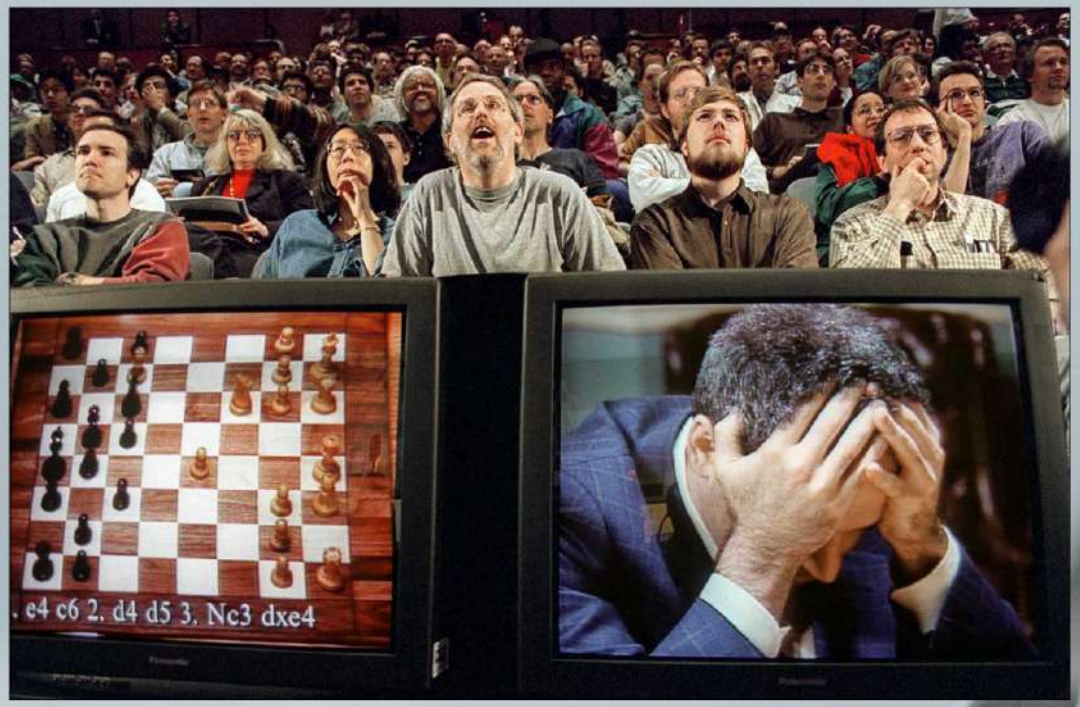
Zenith, Z89, 1979, Wikipedia





Garry Kasparov vs IBM Deep Blue

The day a computer beat a chess world champion, 1997



https://rarehistoricalphotos.com/kasparov-deep-blue-1997/

25 Years of Java

Continues to adapt

- Java in containers
- Java 15, Sept 2020

> Some analysts believe that [Java] has the potential to transform the internet.

News

SunWorld preview

Sun will pour out HotJava Web tools, introduce Internet firewall

By Ellen Messmer
Mountain View, Calif.

Sun Microsystems, Inc. this week will unveil a Webful of Internet software that companies can use to support electronic commerce applications.

While World-Wide Web browser and server software have become almost commonplace, Sun hopes to attract a second look with Web software called Java that adds high-speed image motion to any Web site. Java-made simulations can be activated and controlled with the HotJava Web browser client software that Sun is also planning to spotlight next week.

Sun's *Web page showcases HotJava.*

"People doing advertising on the 'Net can use HotJava to create image motion, such as creating a view of their product running," said Katherine Webster, Sun's manager of Internet market development.

The shared Web site run by the Internet Shopping Network as a retail outlet will be making use of HotJava, according to Webster.

Interactive language

Some analysts believe that the Java programming language has the potential to transform the Web.

"The current programming languages used on the Internet, such as Tckl, Perl, C or G++, are used a lot. But they are slow in how they carry out commands, and they can't do much interactively," said Jerry Michalski, managing editor of the "Release 1.0" newsletter.

Java is a distributed real-time language with software "knobs" for activating distributed applications, such as to run a simulation and manipulate images.

"It could be used with payment systems. And it's possible it could enhance the entire 'Net," Michalski said.

The object-oriented Java programming language can be used on any scalable processor architecture (SPARC) workstation, and Sun is now working on porting Java to the Macintosh and personal computer platforms, as well.

To encourage Internet users to utilize the HotJava browser, Sun has posted the browser at its Internet Web sites for free at http://www.sun.com.

Sun this week also will begin shipping a previously announced Internet server called Netra.

Netra is a hardware/software bundle that includes Sun's Solaris 2.4 operating system with Web server software from Netscape Communications, Inc.

Using SunScreen for protection

Although Sun has faced criticism in the past for the security holes in its Unix software, it hopes to change that image with the introduction of a network firewall built to control communications from a corporate TCP/IP network to the public Internet.

Called SunScreen S-100, the firewall prevents unauthorized remote access to the Netra server. The firewall can also encrypt corporate traffic flowing across it.

SunScreen, which will ship in September, is currently being beta-tested within the network services division of Union Bank of Switzerland in Zurich.

So far, it has worked well to secure communication between multiple Union Bank of Switzerland sites, said Giorgio Scherla, network administrator for the bank.

The firewall, which uses RSA Data Security, Inc. public-key encryption, allows the bank to authenticate data traveling from site to site through what is known as a digital signature. The firewall can also encrypt the data for confidentiality.

Pricing for Sun's Internet suite of products has not been announced, but up-to-date data can be obtained at its Web site.

©Sun: (415) 786-8199.

WilTel getting close to offering end-to-end frame relay service

By David Rohde
Tryland, Pa.

An Internet access provider here has become the first customer to utilize a Network-to-Network Interface (NNI) between frame relay services offered by WilTel and a local exchange carrier (LEC).

The vote of confidence from provider VoiceNet, Inc. sets the stage for WilTel to become the first interexchange carrier (IXC) to launch an end-to-end frame relay service, with an announcement now expected in June or July.

End-to-end frame relay is expected to benefit users that have a large concentration of nodes within a single local access and transport area, such as retailers and banks.

Until WilTel formally announces its end-to-end service

Where you can get it

WilTel interfaces to LEC frame relay services are available in LATAs surrounding the following cities:

- Atlanta
- Baltimore
- Boston
- Dallas
- Denver
- Hartford, Conn.
- Houston
- Kansas City, Mo.
- Los Angeles
- Miami
- Minneapolis
- Newark, N.J.
- New York

The NNI connection costs $220 per month for each 64K bit/sec of bandwidth traveling from the WilTel cloud into the Bell Atlantic net, plus $50 to $156 a month per 64K bit/sec going the other way.

NNI arrangements have been slow to roll out as IXCs and LECs have struggled to reconcile differences among their frame relay net designs.

For example, WilTel allows the assigned speed of each individual permanent virtual circuit (PVC) be set anywhere up to the speed of the port connection to the carrier's network. But Bell Atlantic only allows an individual PVC to be assigned at up to half the port speed.

Likewise, WilTel allows the assigned speed of all the PVCs associated with a port to total 200% of

Network World, May 22, 1995, Page 10

Drive

Industries profiting by scaling applications to larger demands

Amazon plans to deliver packages ... in just one day ... will up the ante for retail rivals such as Walmart.

- Reuters, April 25, 2019

Drive

Industries profiting by scaling applications to larger demands

Many other business

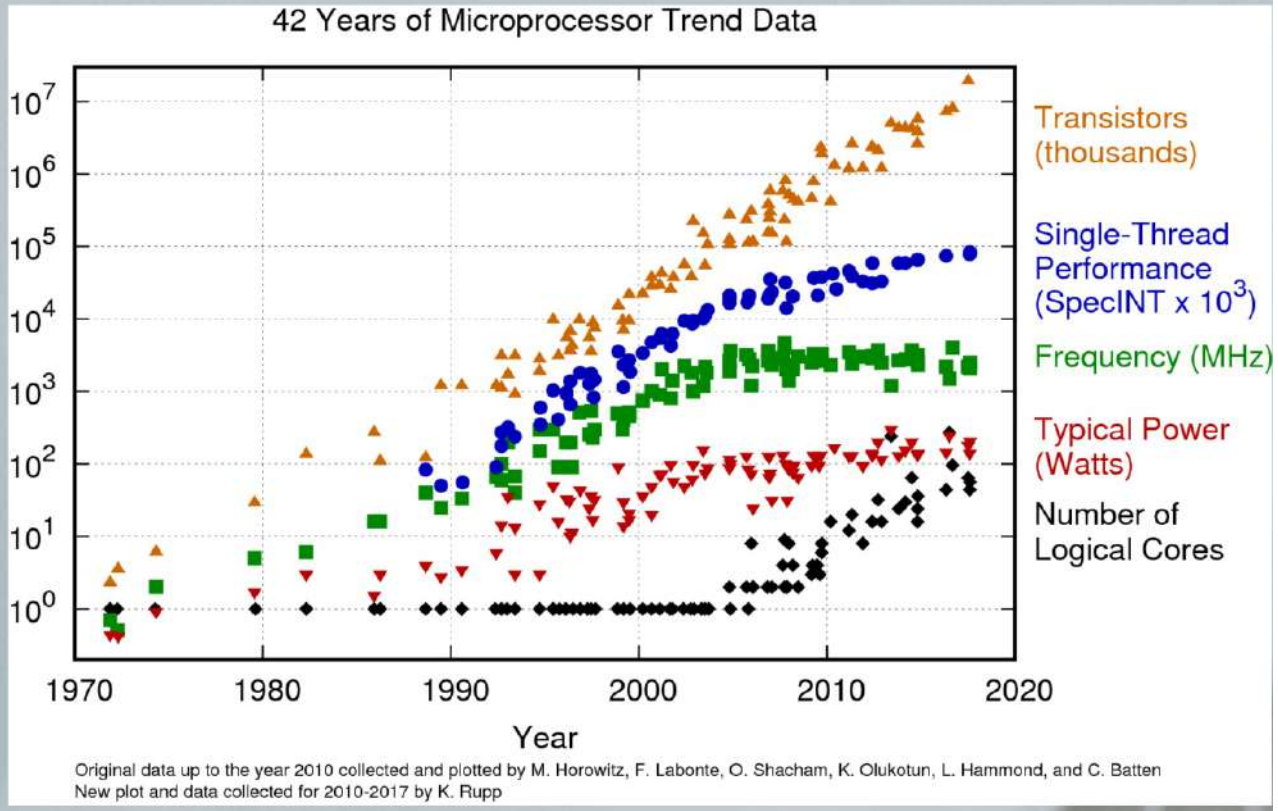
- Inventory management
- Custom relationships
- Global logistics
- Medicine
- Machine/Business intelligence
- Robotics
- Insurance & Financial

Amazon plans to deliver packages ... in just one day ... will up the ante for retail rivals such as Walmart.

- Reuters, April 25, 2019

It can't continue forever.

- Gordon Moore



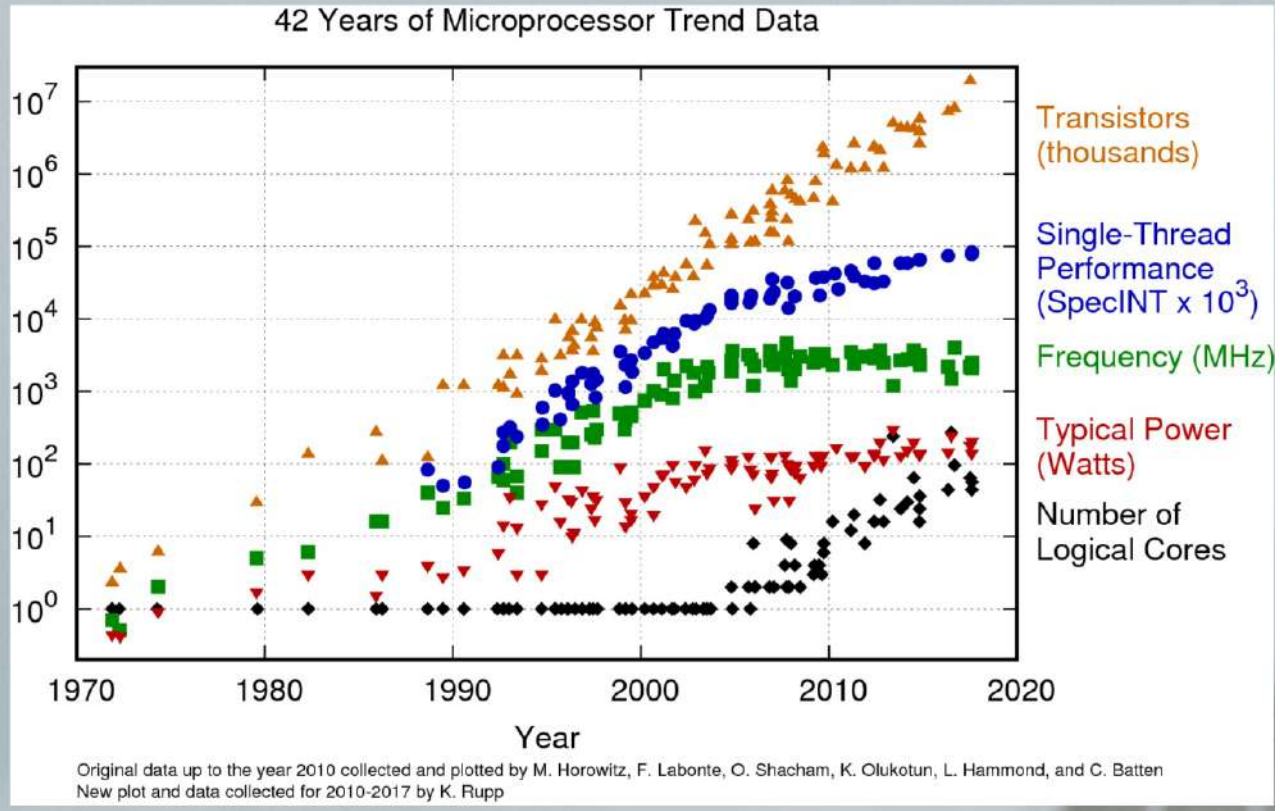
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

It can't continue forever.

- Gordon Moore

MOSFET scaling

10 µm 1971
14 mn 2014
10 nm 2016
7 nm 2018
5 nm 2020
3 nm 2022
2 nm 2023
1.4 nm 2029



https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

It can't continue forever.

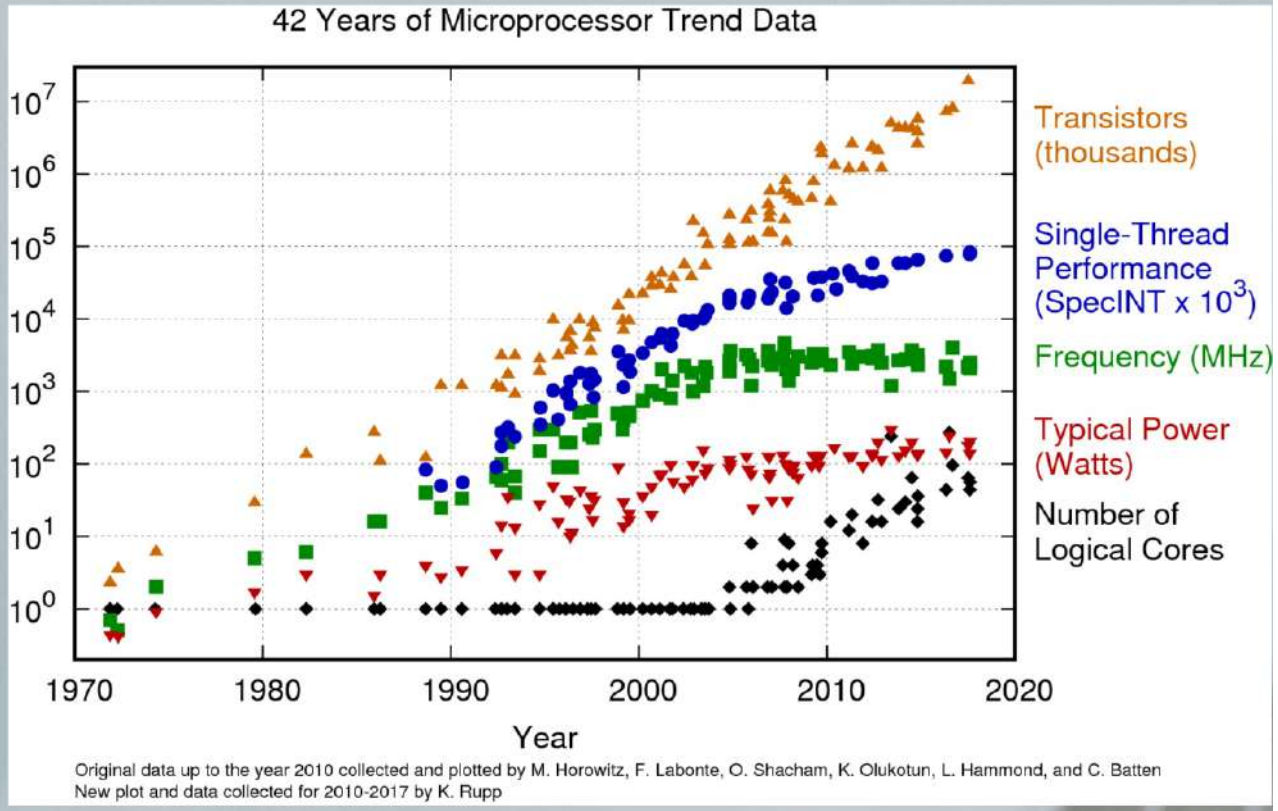
- Gordon Moore

MOSFET scaling

10 µm 1971
14 nm 2014
10 nm 2016
7 nm 2018
5 nm 2020
3 nm 2022
2 nm 2023
1.4 nm 2029

...to benefit from future processors over what you have now, make sure to have **parallel** workloads.

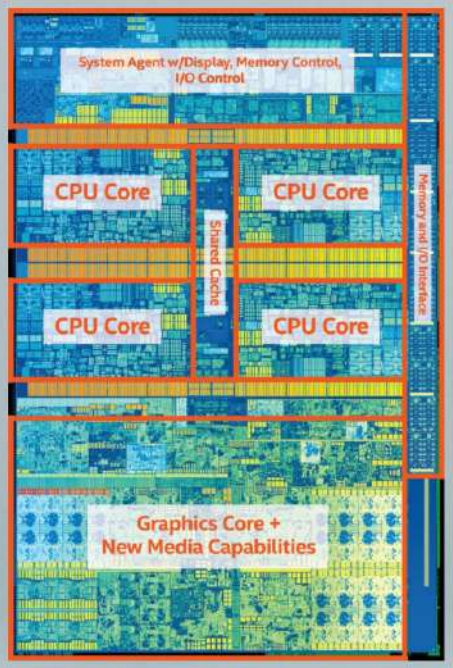
- Karl Rupp



https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Multiple Cores

Scaling with one machine

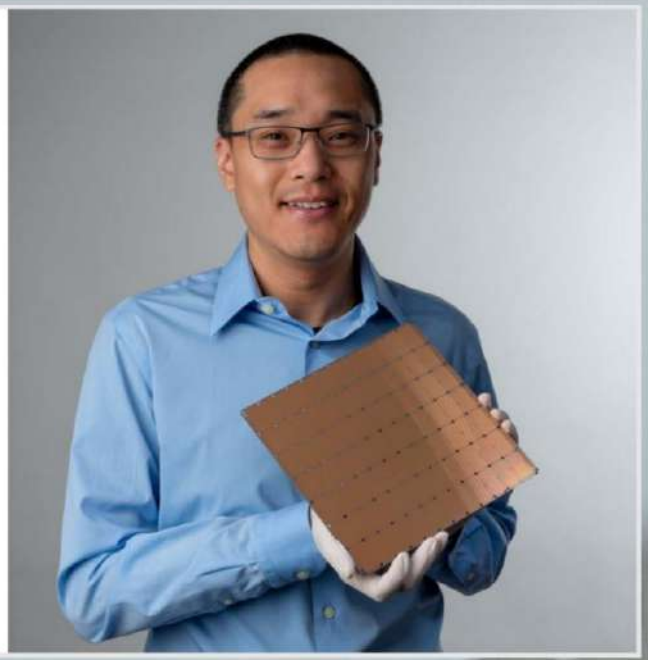


PCWorld: Official Intel 7th-gen Kaby Lake Review: One big change makes up for smaller ones
By Gordon Mah Ung Executive Editor, PCWorld JAN 3, 2017 9:00 AM PST

Largest Chip Ever Built

- 46,225 mm^2 silicon
- 1.2 trillion transistors
- 400,000 AI optimized cores
- 18 Gigabytes of On-chip Memory
- 9 PByte/s memory bandwidth
- 100 Pbit/s fabric bandwidth
- TSMC 16nm process

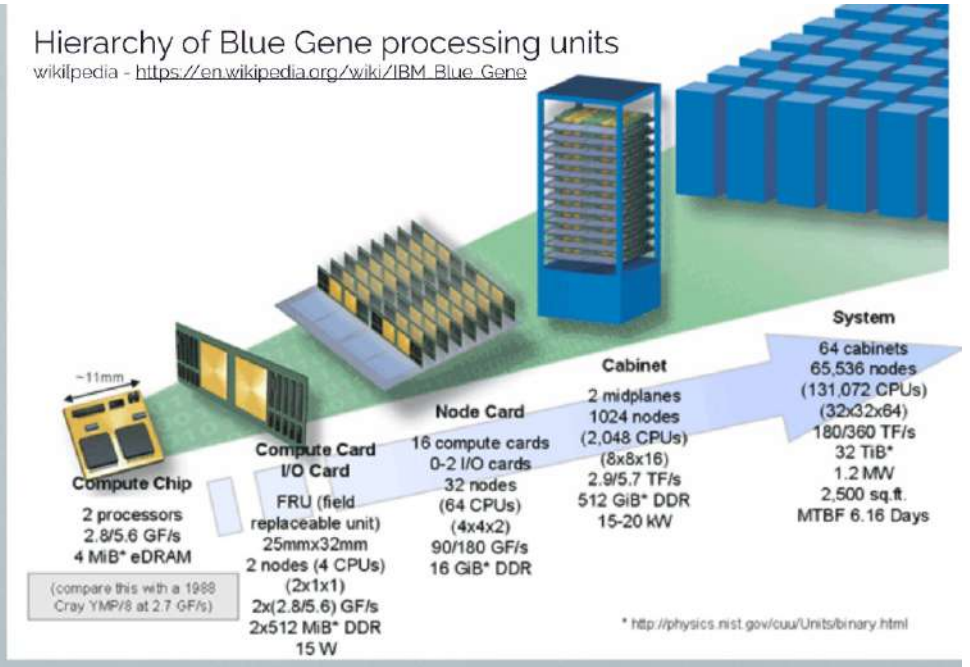
cerebras



PCWorld: Cerebras Systems' new deep-learning chip is as big as your keyboard, and the largest ever.
By Mark Hachman, Senior Editor, PCWorld AUG 19, 2019 6:19 PM PDT

Scaling Hardware

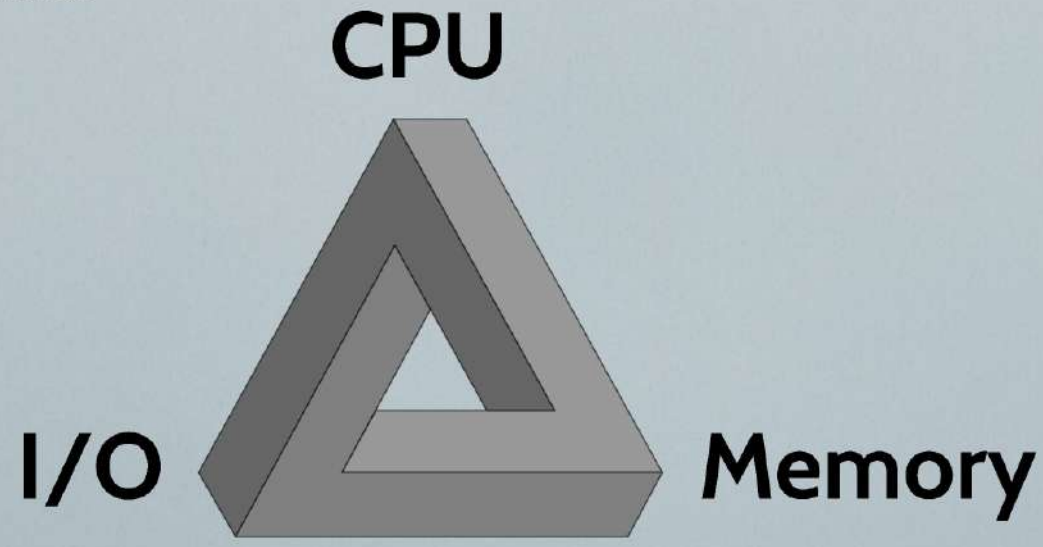
Computer clusters
High Performance Computing (HPC)
Distributed computing
Grid computing
Parallel computing
High-throughput computing
Job processing
Batch processing
Service Oriented Architectures



Scaling Software

Beowulf
Borg and Omega
Cloud providers
JXTA

Resource Commodities



Operating systems manage these resource
Virtual machines parallel access to these resources
Containers are lighter technique for resource access

High Cohesion and Low Coupling

CS-101

Modular business features
Replications for scaling
Sharding processing and data
Inverse Conway maneuver

Mel Brooks - "Oy!"

History of the World, ~10~15 commandments

https://youtu.be/Zuvb3zi7eZc?t=43

8 Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

- L. Peter Deutsch at Sun Microsystems

- James Gosling added 8th, 6 years later.

Why are you making your life more complicated?

Why are you making your life more complicated?

- Monoliths, are often easier.
- Cluster are expensive

Why are you making your life more complicated?

- Monoliths, are often easier.
- Cluster are expensive

- Distributed computing is deceivingly hard, Kubernetes attempts to make is easier

Why are you making your life more complicated?

- Monoliths, are often easier.
- Cluster are expensive

- Distributed computing is deceivingly hard, Kubernetes attempts to make is easier

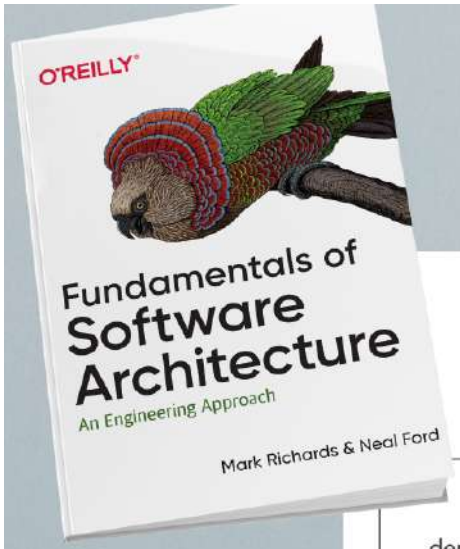
- Can your team handle this? What's your Maturity Model?
- Does leadership and management understand commitment?

space-based architecture






	layered monolith	microkernel	microservices	service-based	event-driven	space-based
agility	★	★★★	★★★★★	★★★★	★★★★	★★★
deployment	★	★★★	★★★★★	★★★★	★★★	★★★
testability	★★	★★★	★★★★★	★★★★	★★	★
performance	★★★	★★★	★★	★★★	★★★★★	★★★★★
scalability	★	★	★★★★★	★★★	★★★★	★★★★★
elasticity	★	★	★★★★	★★	★★★	★★★★★
simplicity	★★★★★	★★★★	★	★★★	★	★
fault-tolerance	★	★	★★★★★	★★★★	★★★★★	★★★
evolvability	★	★★★	★★★★★	★★★★	★★★★★	★★★
total cost	★★★★★	★★★★★	★	★★★★	★★★	★★

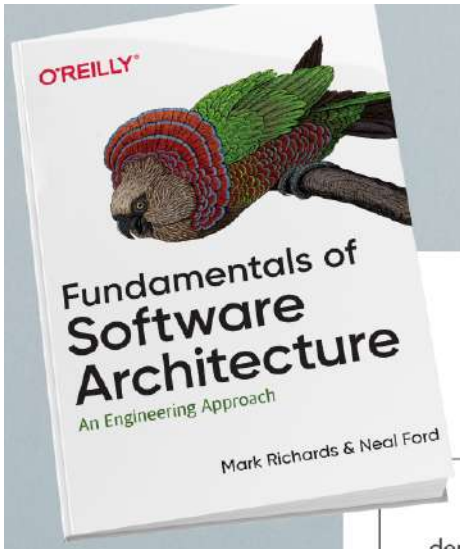


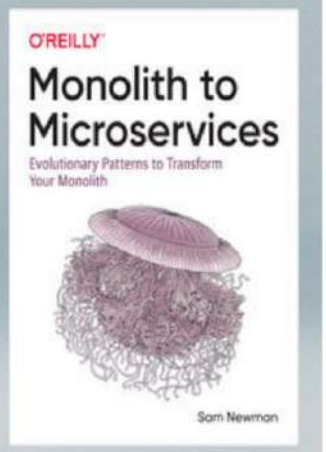
Mark Richards

Neal Ford

space-based architecture

	layered monolith	microkernel	microservices	service-based	event-driven	space-based
agility	★	★★★	★★★★★	★★★★	★★★★	★★★
deployment	★	★★★	★★★★★	★★★★	★★★	★★★
testability	★★	★★★	★★★★★	★★★★	★★	★
performance	★★★	★★★	★★	★★★	★★★★★	★★★★★
scalability	★	★	★★★★★	★★★	★★★★	★★★★★
elasticity	★	★	★★★★	★★	★★★	★★★★★
simplicity	★★★★★	★★★★	★	★★★	★	★
fault-tolerance	★	★	★★★★★	★★★★	★★★★★	★★★
evolvability	★	★★★	★★★★★	★★★★	★★★★★	★★★
total cost	★★★★★	★★★★★	★	★★★★	★★★	★★





Mark Richards

Neal Ford

space-based architecture

	layered monolith	microkernel	microservices	service-based	event-driven	space-based
agility	★	★★★	★★★★★	★★★★	★★★★	★★★
deployment	★	★★★	★★★★★	★★★★	★★★	★★★
testability	★★	★★★	★★★★★	★★★★	★★	★
performance	★★★	★★★	★★	★★★	★★★★★	★★★★★
scalability	★	★	★★★★★	★★★	★★★★	★★★★★
elasticity	★	★	★★★★	★★	★★★	★★★★★
simplicity	★★★★★	★★★★	★	★★★	★	★
fault-tolerance	★	★	★★★★★	★★★★	★★★★★	★★★
evolvability	★	★★★	★★★★★	★★★★	★★★★★	★★★
total cost	★★★★★	★★★★★	★	★★★★	★★★	★★


Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community

kubernétés: a steersman, pilot

Original Word: κυβερνήτης, ου, ὁ

Part of Speech: Noun, Masculine

Transliteration: kubernétés

Phonetic Spelling: (koo-ber-nay'-tace)

Short Definition: a steersman, pilot

Definition: a steersman, pilot; met: a guide, governor.

7 sided heptagon

7 of 9

Project Seven

The nicer borg

https://tinyurl.com/ybpktsno

Founders: Joe Beda, Brendan Burns, Craig McLuckie

Growing Ecosystem

Defacto container manager

- April 2015, Google published paper
- Borg 10-15 years of development, ~2003
- Omega 2013, an offspring of Borg
- Kubernetes released 2014
- Google donated V1 to Cloud Native Computing Foundation (CNCF) in 2015

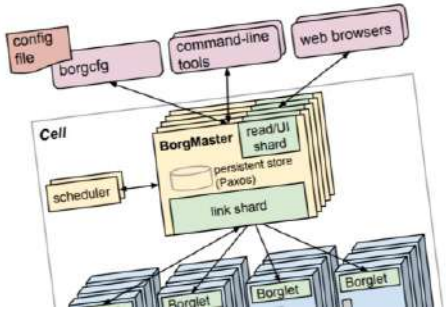
Large-scale cluster management at Google with Borg

Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports



- April 2015, Google published paper
- Borg 10-15 years of development, ~2003
- Omega 2013, an offspring of Borg
- Kubernetes released 2014
- Google donated V1 to Cloud Native Computing Foundation (CNCF) in 2015

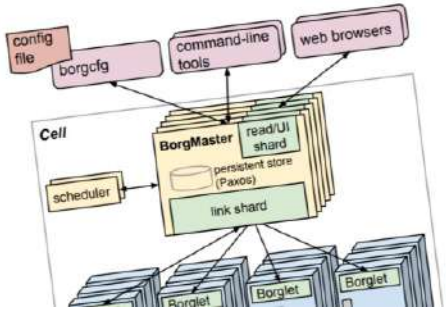
Large-scale cluster management at Google with Borg

Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports



Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative ex-

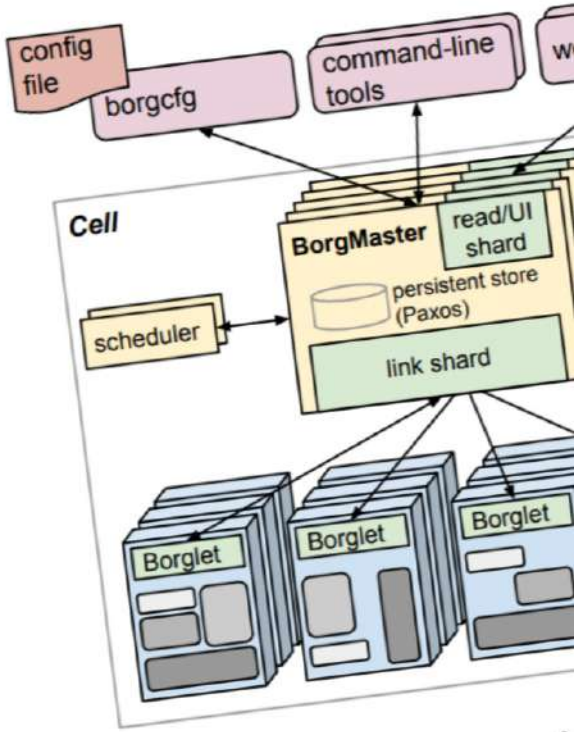
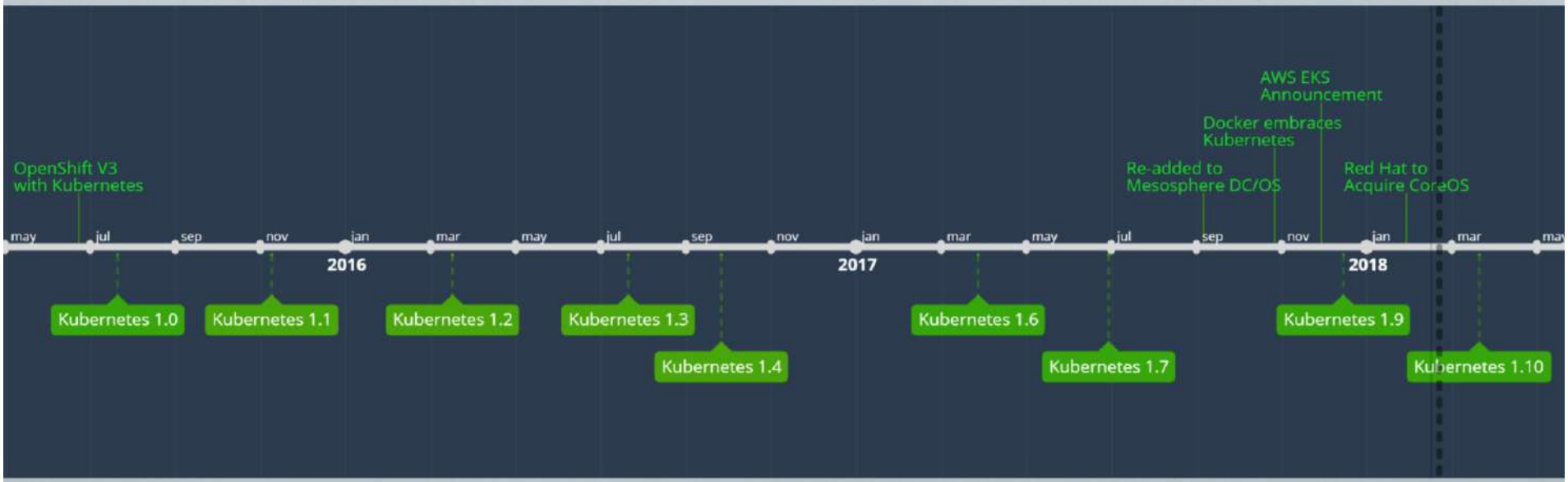


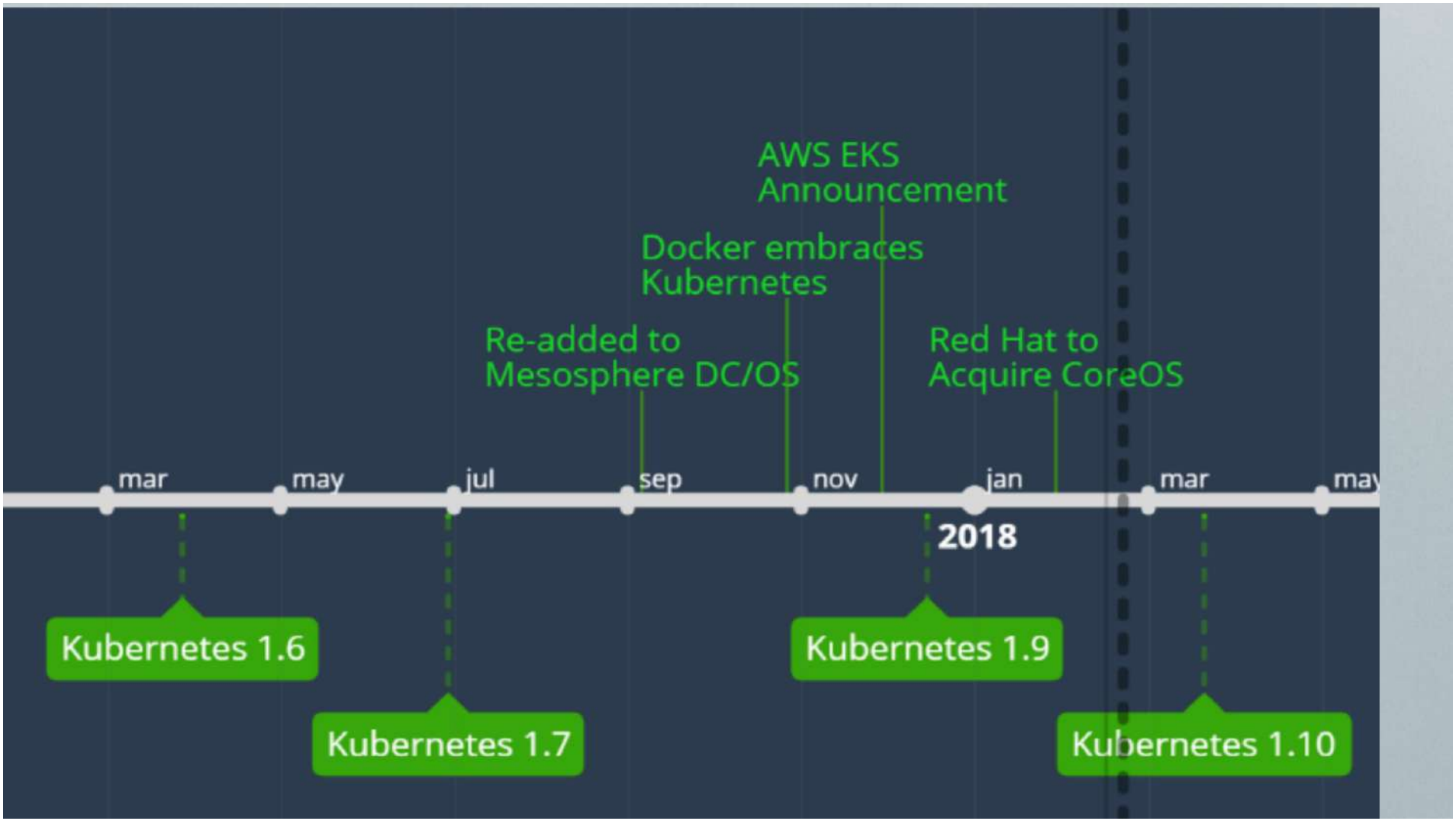
Figure 1: The high-level architecture of
of the thousands of worker nodes are sh

cluding with a set of qualitative obs
Borg in production

1.19 in August 2020



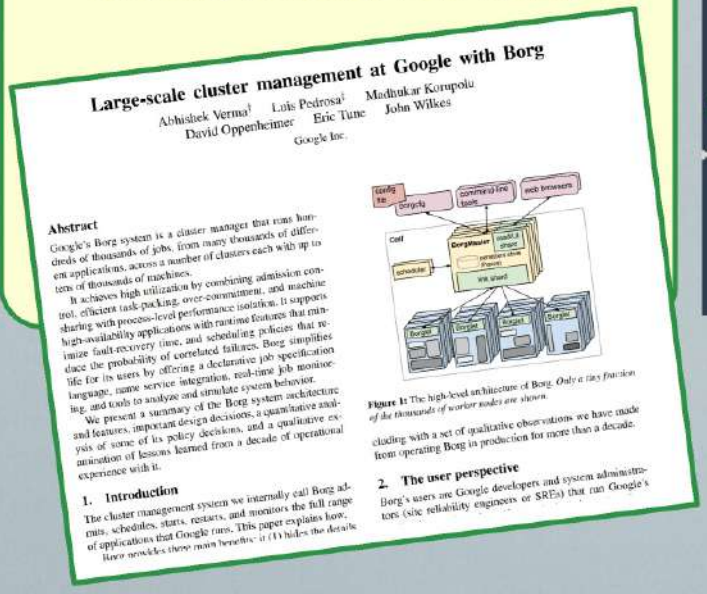
Detailed timeline: https://blog.risingstack.com/the-history-of-kubernetes


AWS EKS
Announcement
Docker embraces
Kubernetes
Re-added to
Mesosphere DC/OS
Red Hat to
Acquire CoreOS
mar
may
jul
sep
nov
jan
mar
2018
Kubernetes 1.6
Kubernetes 1.7
Kubernetes 1.9
Kubernetes 1.10

Growing Ecosystem

Defacto container manager

- April 2015, Google published paper
- Borg 10-15 years of development, ~2003
- Omega 2013, an offspring of Borg
- Kubernetes released 2014
- Google donated V1 to Cloud Native Computing Foundation (CNCF) in 2015



Large-scale cluster management at Google with Borg

Abhishek Verma[1] Luis Pedrosa[2] Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes

Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

1. Introduction

The cluster management system we internally call Borg admits, schedules, starts, restarts, and monitors the full range of applications that Google runs. This paper explains how.

Borg provides three main benefits: it (1) hides the details

Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

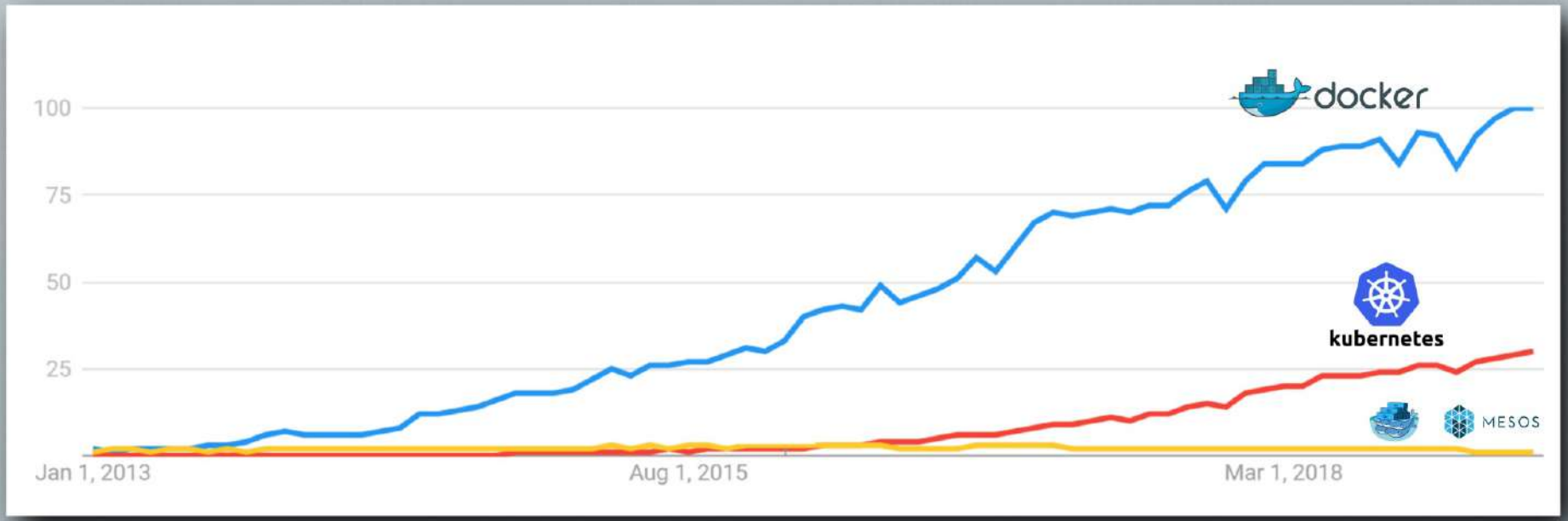
2. The user perspective

Borg's users are Google developers and system administrators (site reliability engineers or SREs) that run Google's

1.19 in August 2020

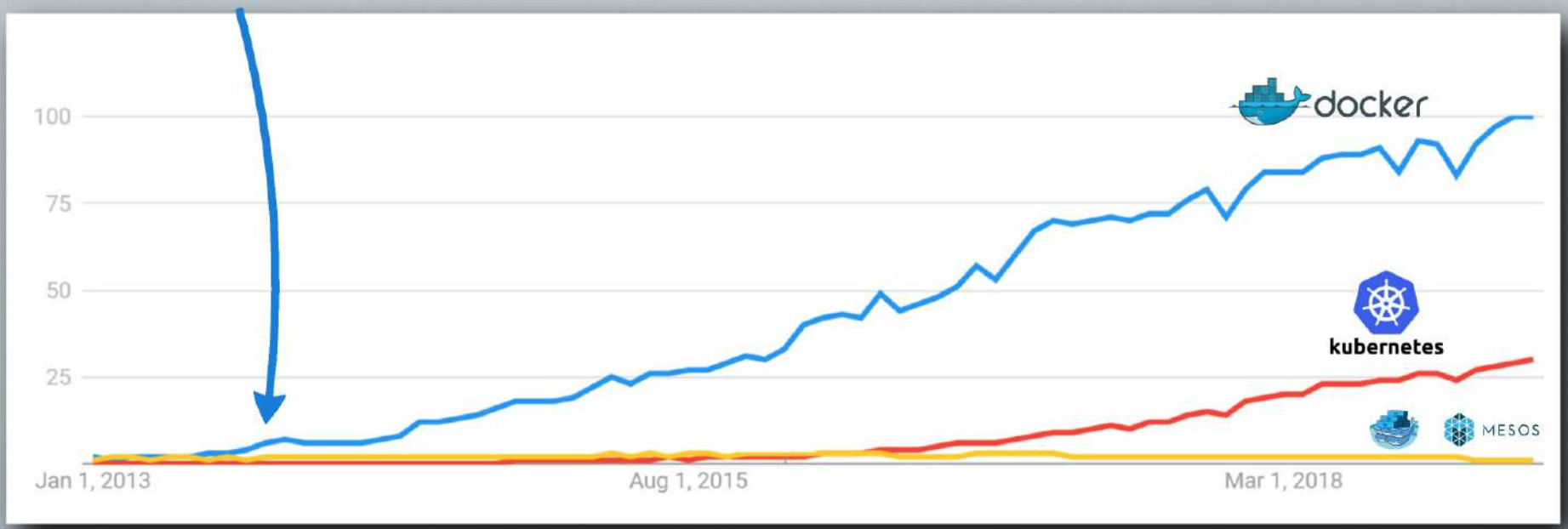
Detailed timeline: https://blog.risingstack.com/the-history-of-kubernetes

Trends
By Google



Trends
By Google

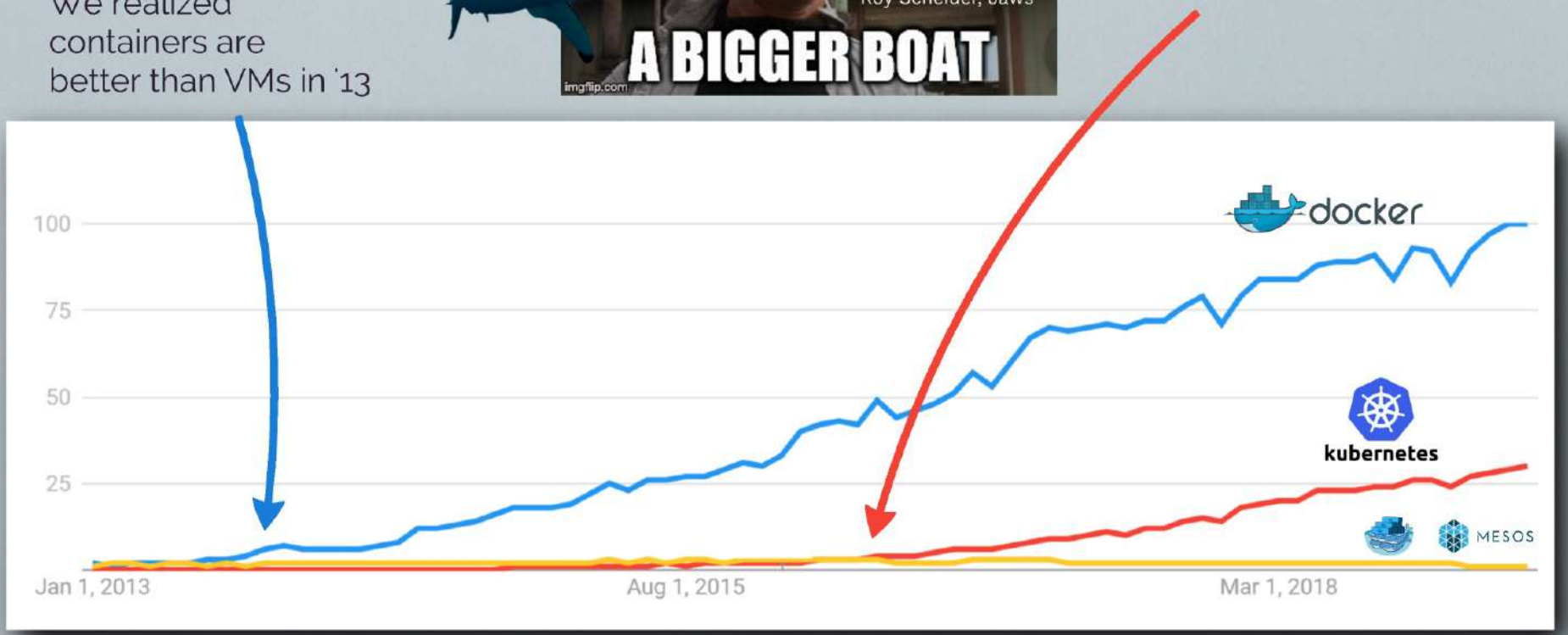
We realized
containers are
better than VMs in '13



Trends
By Google

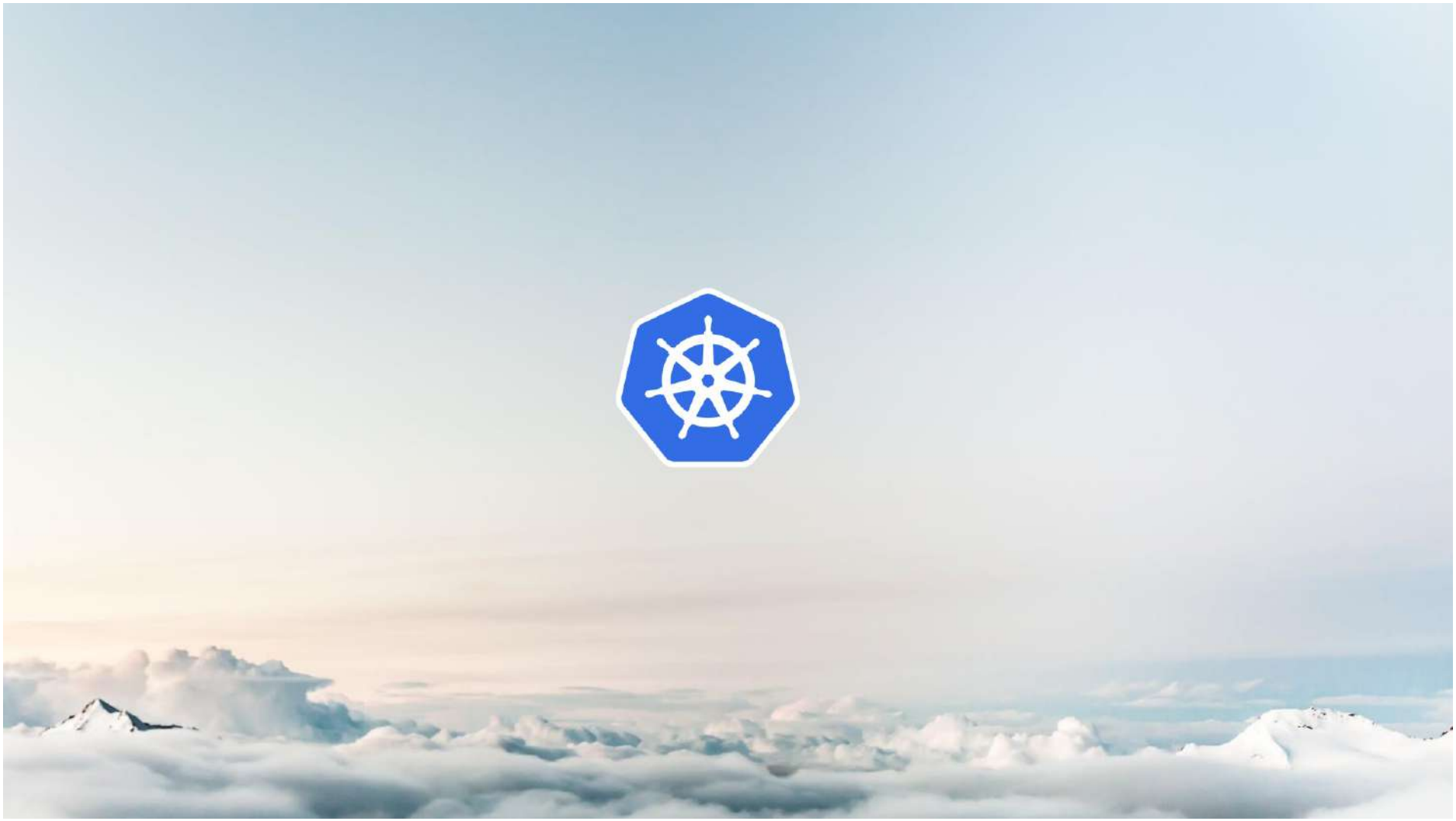
We realized containers are better than VMs in '13

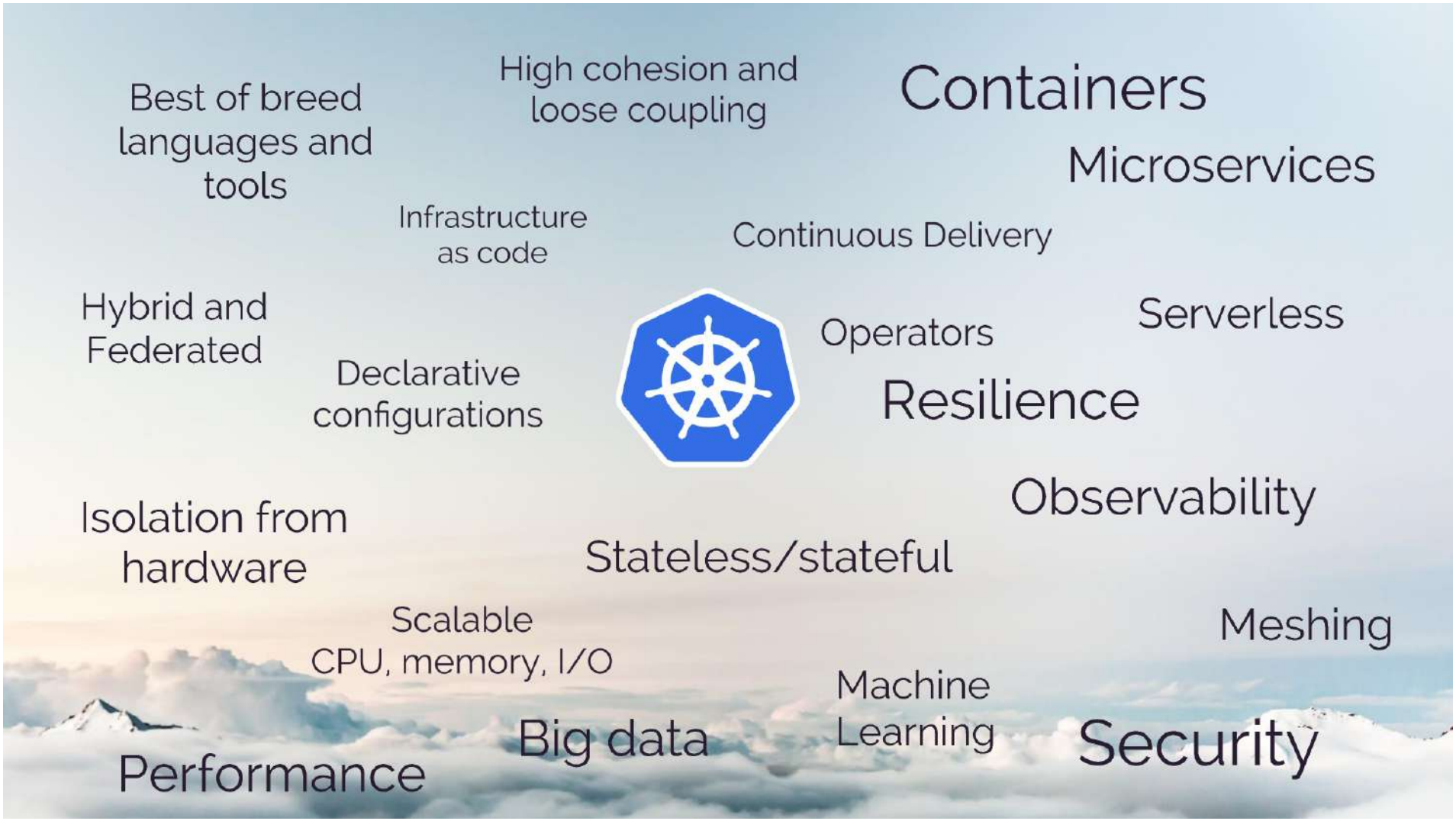
We realized something needs to manage all these containers in '16



CLOUD NATIVE
COMPUTING
FOUNDATION

Cloud native computing uses
an open source software stack
to deploy applications as microservices,
packaging each part into its own container,
and dynamically orchestrating those containers
to optimize resource utilization.




Best of breed languages and tools
High cohesion and loose coupling
Containers
Microservices
Infrastructure as code
Continuous Delivery
Hybrid and Federated
Declarative configurations
Operators
Serverless
Resilience
Observability
Isolation from hardware
Stateless/stateful
Scalable CPU, memory, I/O
Meshing
Machine Learning
Big data
Security
Performance

Healthy community

Healthy community

THE LINUX FOUNDATION

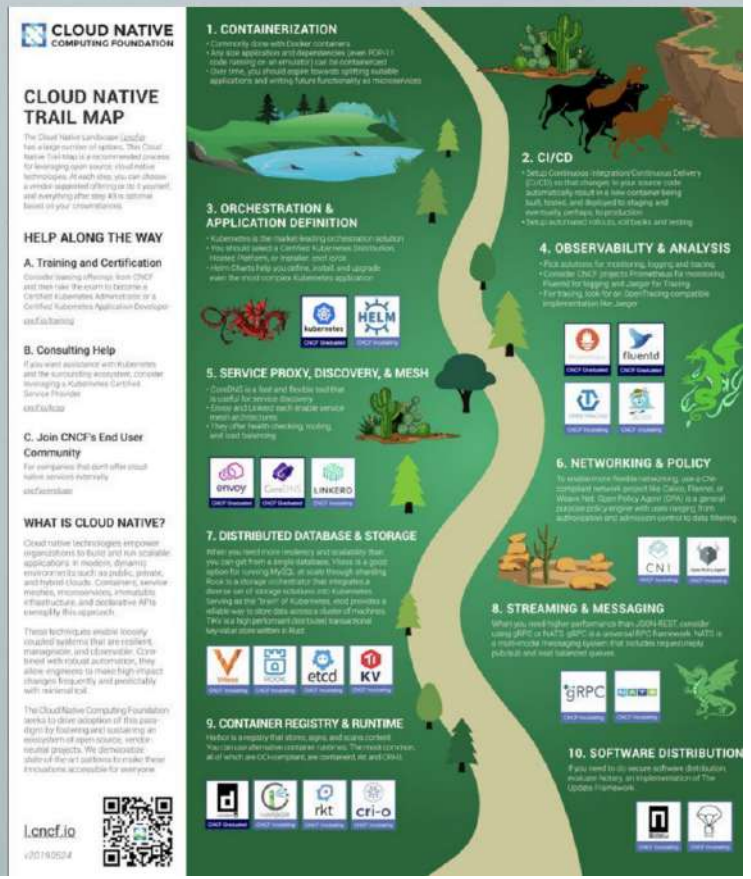
KubeCon
A CNCF EVENT

CLOUD NATIVE COMPUTING FOUNDATION

OPEN CONTAINER INITIATIVE

CD.FOUNDATION

https://landscape.cncf.io



Healthy community

THE LINUX FOUNDATION

KubeCon
A CNCF EVENT

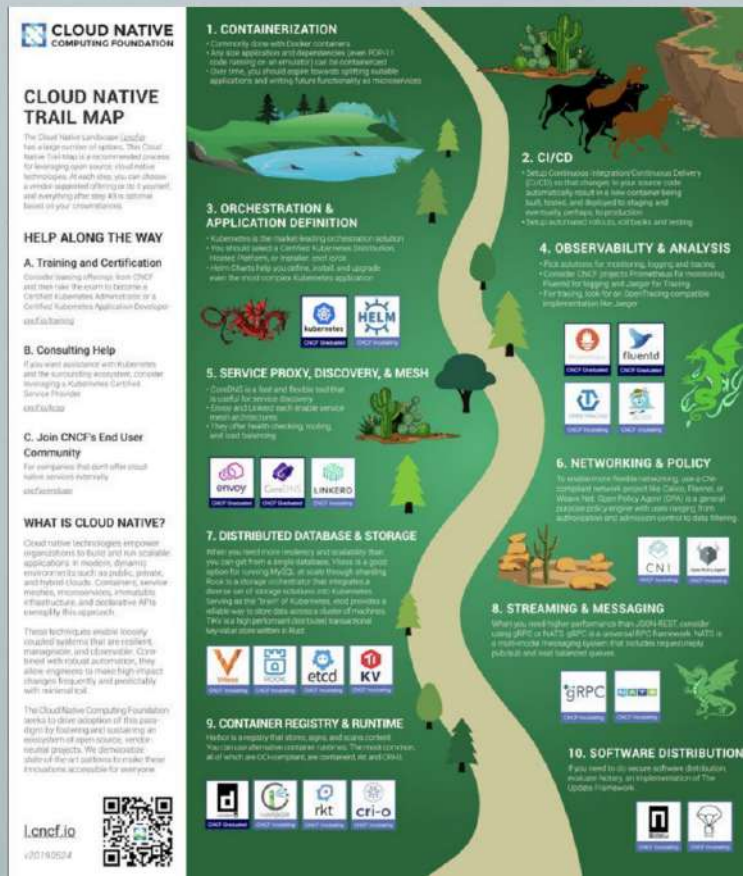
CLOUD NATIVE COMPUTING FOUNDATION

Google Kubernetes PodCast
Craig Box · Adam Glick

OPEN CONTAINER INITIATIVE

CD.FOUNDATION

https://landscape.cncf.io



Healthy community

THE LINUX FOUNDATION

KubeCon
A CNCF EVENT

CLOUD NATIVE COMPUTING FOUNDATION

Google Kubernetes PodCast
Craig Box · Adam Glick

OPEN CONTAINER INITIATIVE

CD.FOUNDATION

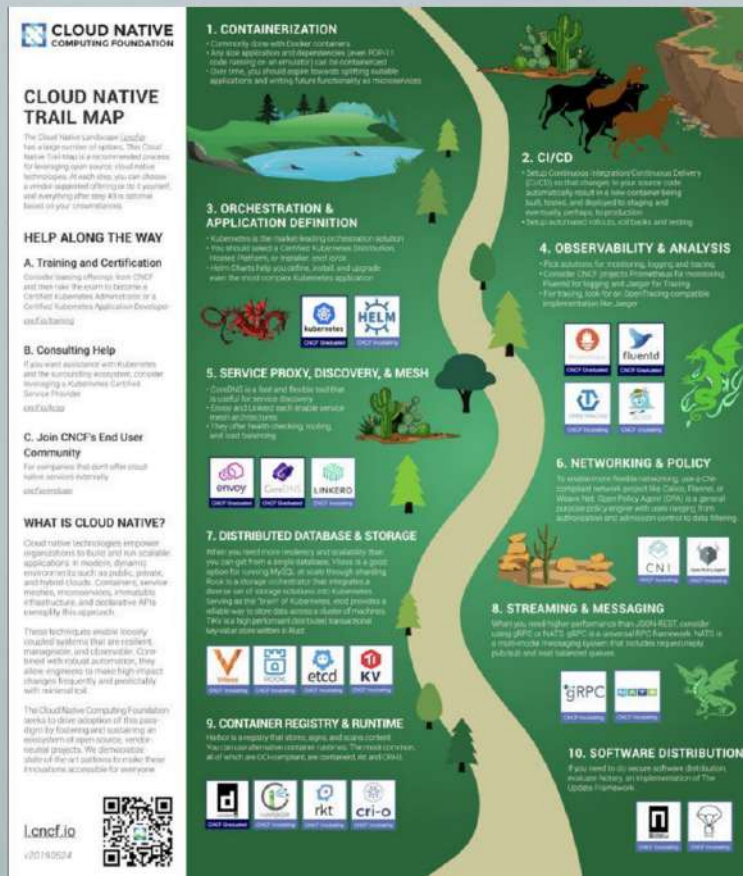
100+ Vendors

https://www.cncf.io/certification/software-conformance/

https://tinyurl.com/jw7mqt8

certified kubernetes

https://landscape.cncf.io



Healthy community

THE LINUX FOUNDATION

KubeCon
A CNCF EVENT

CLOUD NATIVE COMPUTING FOUNDATION

Google Kubernetes PodCast
Craig Box · Adam Glick

OPEN CONTAINER INITIATIVE

CD.FOUNDATION

100+ Vendors

https://www.cncf.io/certification/software-conformance/

https://tinyurl.com/jw7mqt8

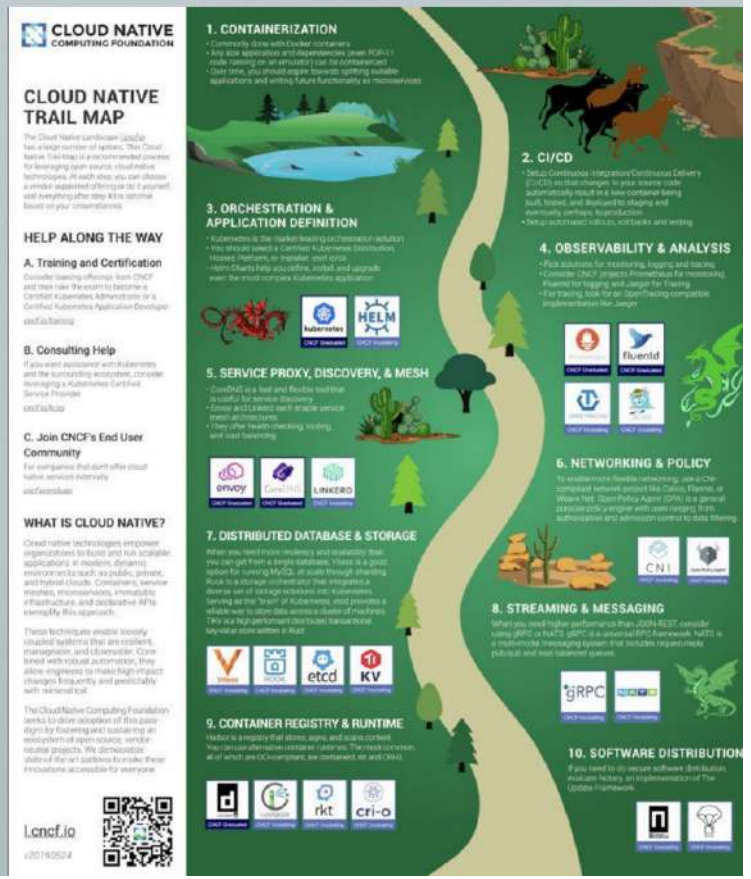
certified kubernetes

CERTIFIED kubernetes APPLICATION DEVELOPER

CERTIFIED kubernetes ADMINISTRATOR

CERTIFIED kubernetes SECURITY SPECIALIST

https://landscape.cncf.io



1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

Commercial Managed Solutions

5+ years of wide-scale adoption by 100+ Kubernetes providers (KaaS)

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

AKS	Microsoft, Azure Container Service (Brendan Burns)
EKS	Amazon, Elastic Container Service
GKE	Google Kubernetes Engine
IKS	IBM, Cloud Kubernetes Service

Commercial Managed Solutions

5+ years of wide-scale adoption by 100+ Kubernetes providers (KaaS)

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

AKS	Microsoft, Azure Container Service (Brendan Burns)
EKS	Amazon, Elastic Container Service
GKE	Google Kubernetes Engine
IKS	IBM, Cloud Kubernetes Service
ACK	Alibaba
DOKS	DigitalOcean
OKE	Oracle
PKE	Banzai
MKE	D2iQ, Day Two iQ
OKD	Red Hat OpenShift, Origin Community Distribution
PKS	VMWare Tanzu, Pivotal, Heptio (Joe Beda, Craig McLuckie)
RKE	Rancher
	Canonical

100+ vendors offering certified Kubernetes

https://kubernetes.io/docs/setup/#production-environment

Commercial Managed Solutions

5+ years of wide-scale adoption by 100+ Kubernetes providers (KaaS)

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**

AKS Microsoft, Azure Container Service (Brendan Burns)
EKS Amazon, Elastic Container Service
GKE Google Kubernetes Engine
IKS IBM, Cloud Kubernetes Service 

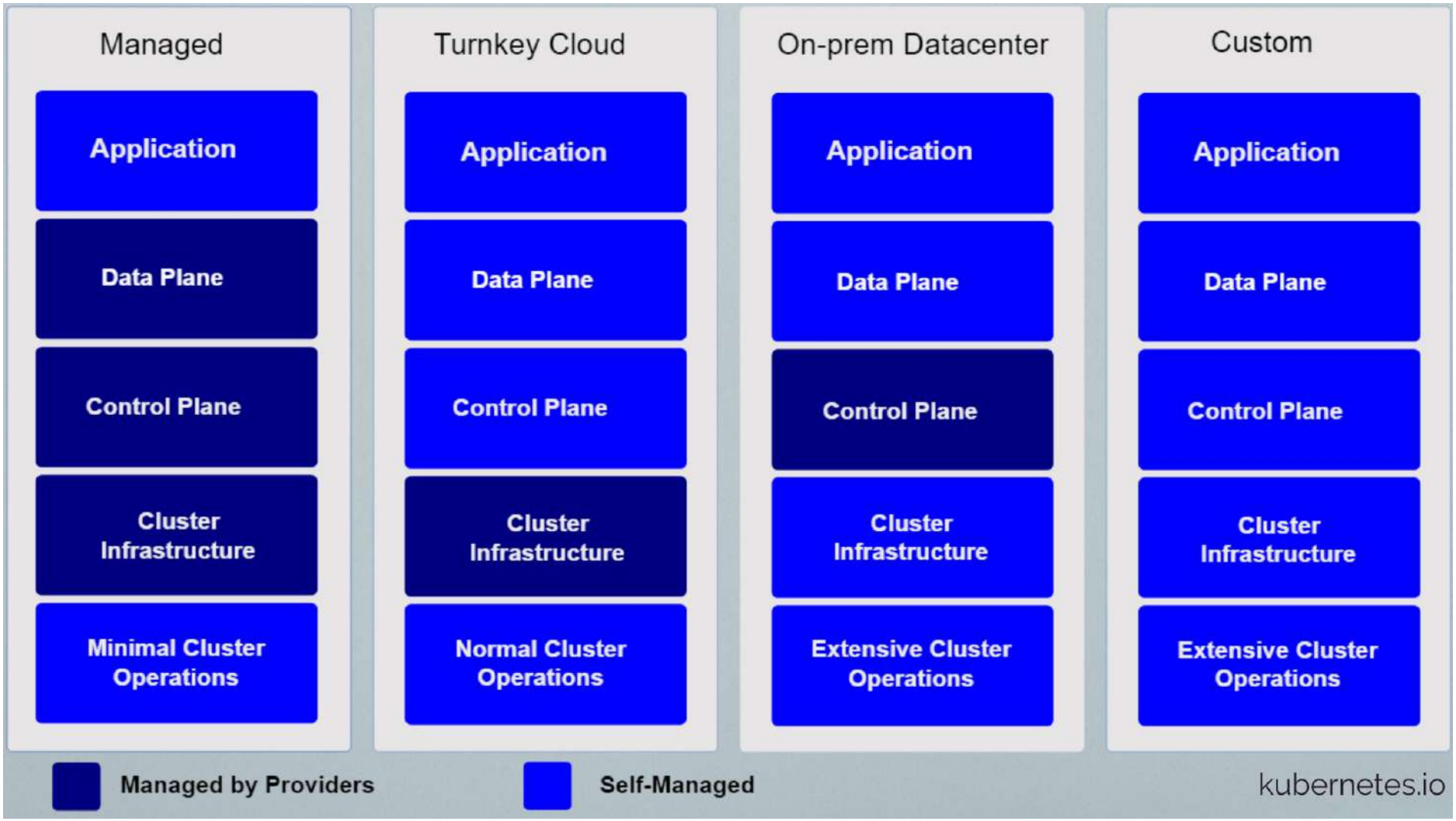
ACK Alibaba
DOKS DigitalOcean
OKE Oracle
PKE Banzai

MKE D2iQ, Day Two iQ
OKD Red Hat OpenShift, Origin Community Distribution
PKS VMWare Tanzu, Pivotal, Heptio (Joe Beda, Craig McLuckie)
RKE Rancher
Canonical

100+ vendors offering certified Kubernetes
https://kubernetes.io/docs/setup/#production-environment


IBM Cloud
Kubernetes Service
IBM Cloud Container Service
renamed to
IBM Cloud Kubernetes Service
May 2018

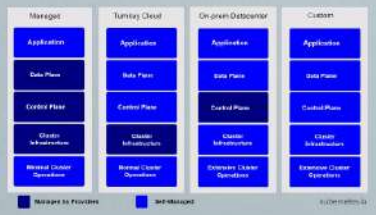

kubernetes.io


Managed
Application
Data Plane
Control Plane
Cluster Infrastructure
Minimal Cluster Operations
Turnkey Cloud
Application
Data Plane
Control Plane
Cluster Infrastructure
Normal Cluster Operations
On-prem Datacenter
Application
Data Plane
Control Plane
Cluster Infrastructure
Extensive Cluster Operations
Custom
Application
Data Plane
Control Plane
Cluster Infrastructure
Extensive Cluster Operations
Managed by Providers
Self-Managed
kubernetes.io

Commercial Managed Solutions

5+ years of wide-scale adoption by 100+ Kubernetes providers (KaaS)

1470, Evrard d'Espinques's painting of King Arthur presiding at the **Round Table with his Knights**



AKS Microsoft, Azure Container Service (Brendan Burns)
EKS Amazon, Elastic Container Service
GKE Google Kubernetes Engine
IKS IBM, Cloud Kubernetes Service

ACK Alibaba
DOKS DigitalOcean
OKE Oracle
PKE Banzai

MKE D2iQ, Day Two iQ
OKD Red Hat OpenShift, Origin Community Distribution
PKS VMWare Tanzu, Pivotal, Heptio (Joe Beda, Craig McLuckie)
RKE Rancher
Canonical

100+ vendors offering certified Kubernetes
https://kubernetes.io/docs/setup/#production-environment

Other Container Orchestrators

D2 IQ

- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

Other Container Orchestrators

Other Container Orchestrators

- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

- Pivotal subsidiary of VMWare 2020
- Heptio (Joe Beda, Craig McLuckie) VMWare 2018
- PCS morphing to PKS

Other Container Orchestrators

Other Container Orchestrators

D2 IQ
- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

Docker Swarm
- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

Pivotal
vmware
- Pivotal subsidiary of VMWare 2020
- Heptio (Joe Beda, Craig McLuckie) VMWare 2018
- PCS morphing to PKS

Amazon ECS
- ECS, proprietary, less expensive.
- EKS increases openness, features, cost

Other Container Orchestrators

- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

- Pivotal subsidiary of VMWare 2020
- Heptio (Joe Beda, Craig McLuckie) VMWare 2018
- PCS morphing to PKS

- ECS, proprietary, less expensive.
- EKS increases openness, features, cost

- Rancher 2.0, 2018, killed Cattle to Kubernetes
- Kubernetes solution provider

Other Container Orchestrators

D2 IQ
- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

Docker Swarm
- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

Pivotal / vmware
- Pivotal subsidiary of VMWare 2020
- Heptio (Joe Beda, Craig McLuckie) VMWare 2018
- PCS morphing to PKS

Amazon ECS
- ECS, proprietary, less expensive.
- EKS increases openness, features, cost

RANCHER
- Rancher 2.0, 2018, killed Cattle to Kubernetes
- Kubernetes solution provider

HashiCorp Nomad
- Lighter container orchestrator
- Not just containers
- Smaller community

Other Container Orchestrators

- Mesos with Marathon, DC/OS
- Rebrand, focus on meta clusters

- 700 enterprise customer sold to Mirantis 2019
- Only supported by Docker enterprise, RIP 2021

- Pivotal subsidiary of VMWare 2020
- Heptio (Joe Beda, Craig McLuckie) VMWare 2018
- PCS morphing to PKS

- ECS, proprietary, less expensive.
- EKS increases openness, features, cost

- Rancher 2.0, 2018, killed Cattle to Kubernetes
- Kubernetes solution provider

- Lighter container orchestrator
- Not just containers
- Smaller community

- Big Data with MapReduce
- 50% of fortune 500 in 2013
- Similar Spark, Presto, Kafka can do on Kubernetes

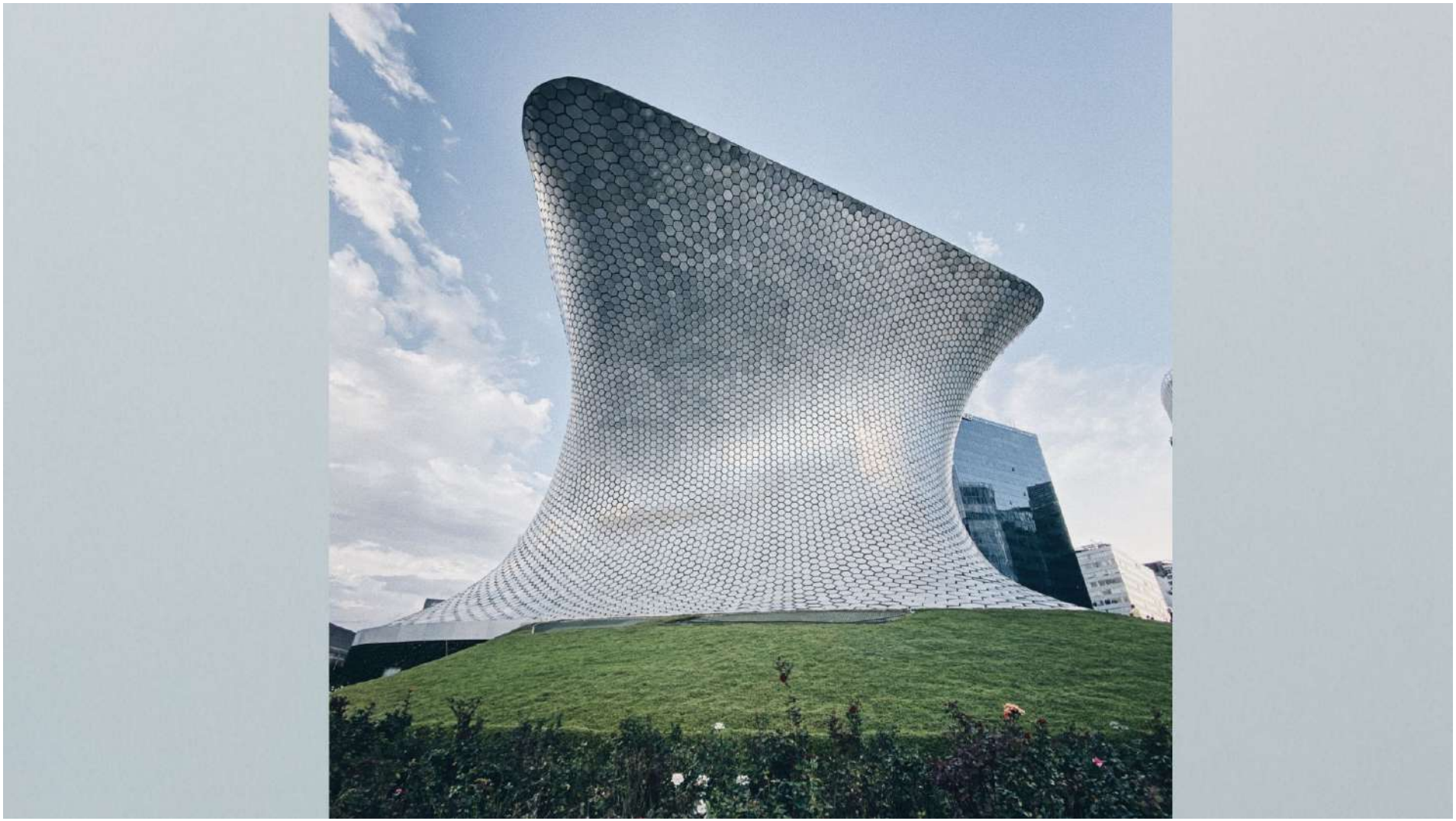

Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community

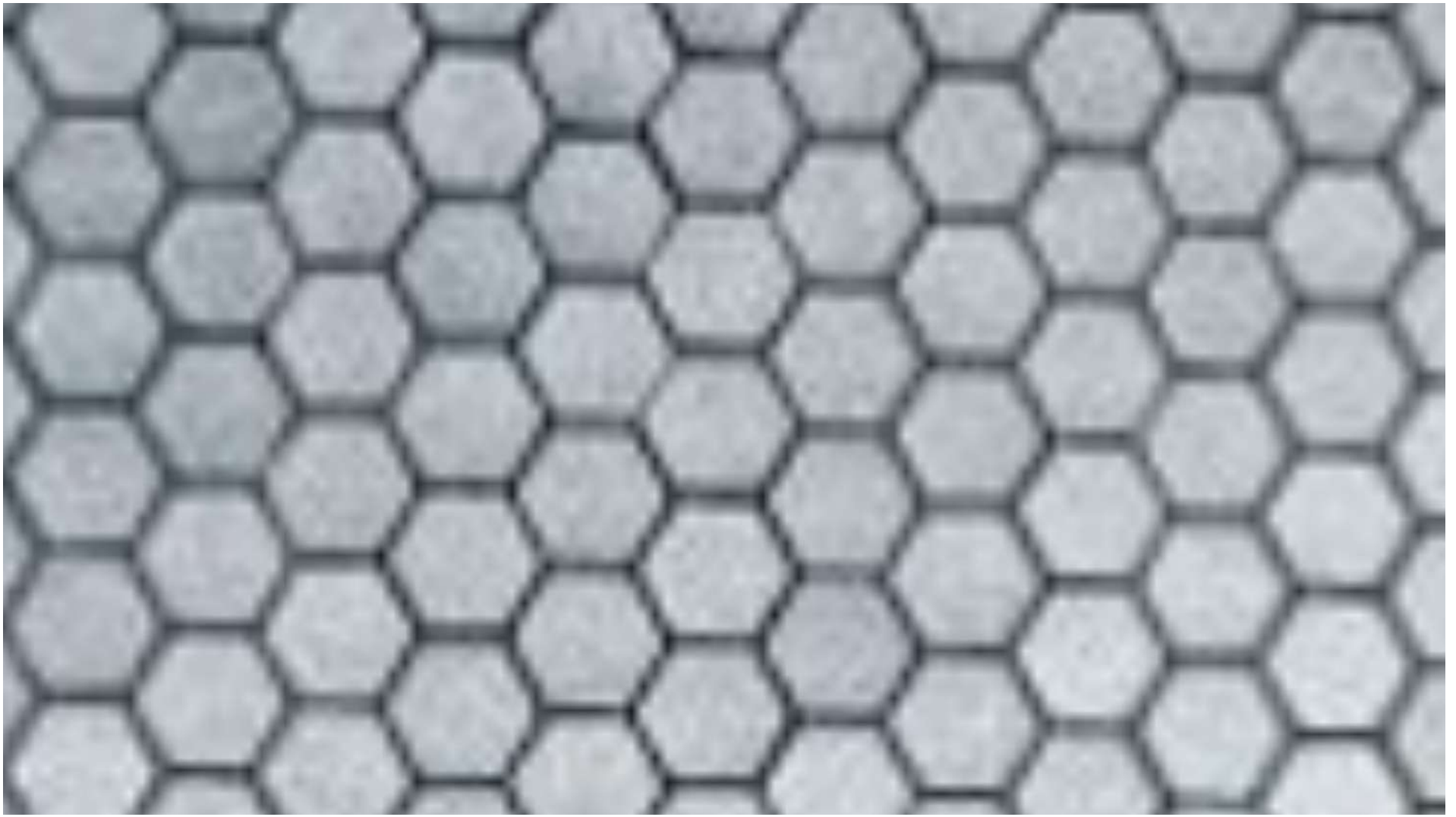
Kubernetes Architecture

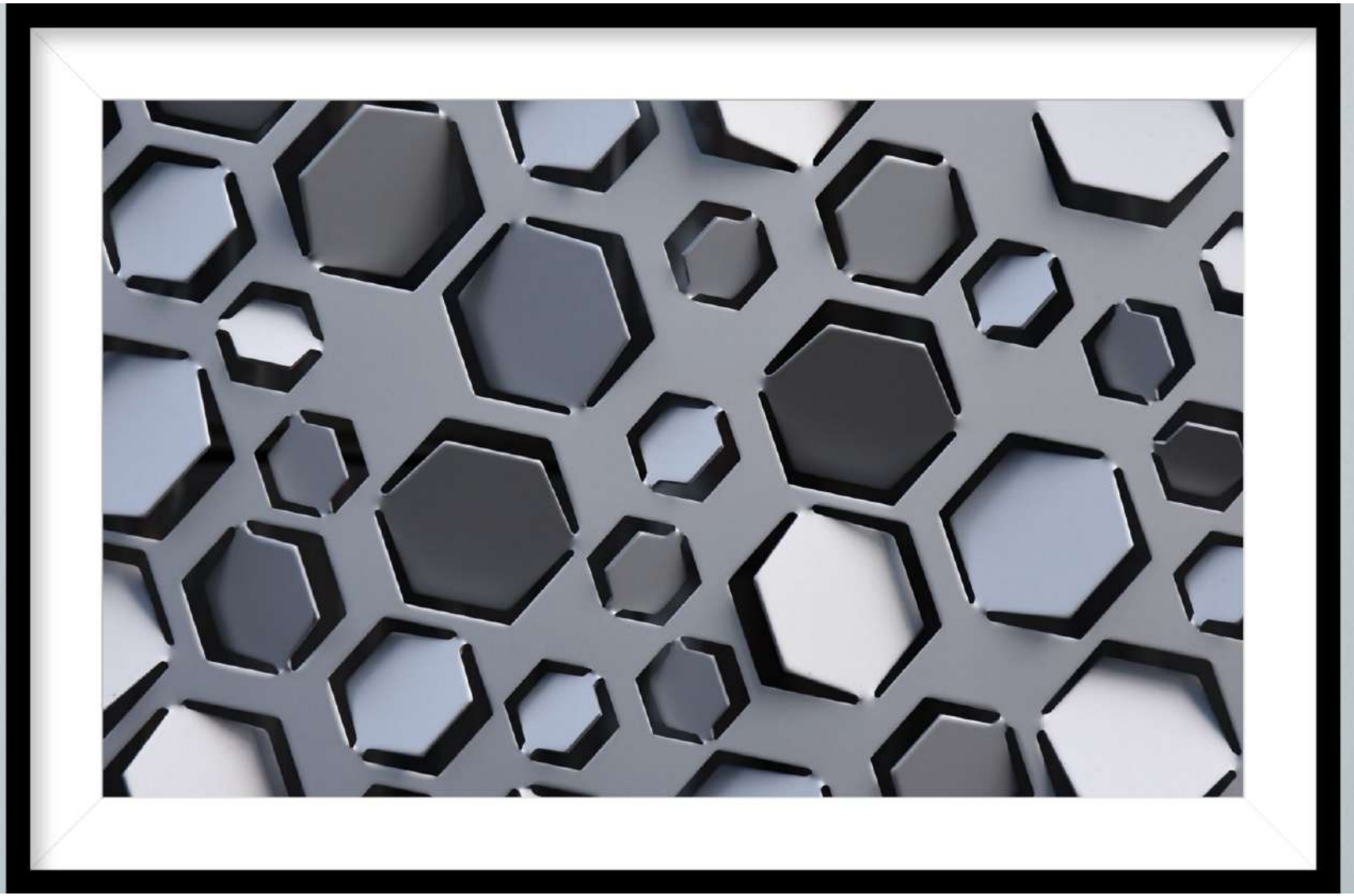
Key Features

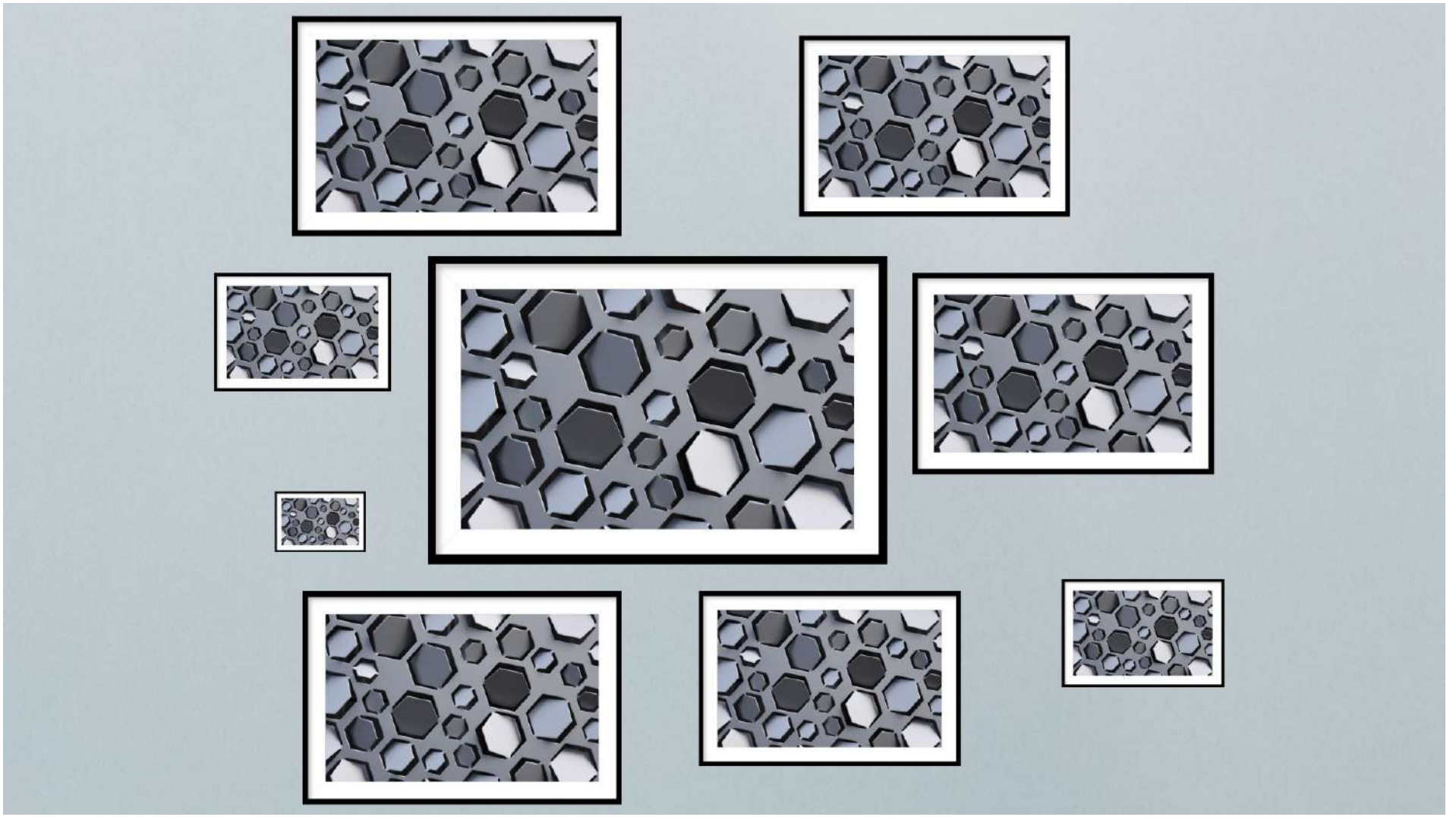
Components

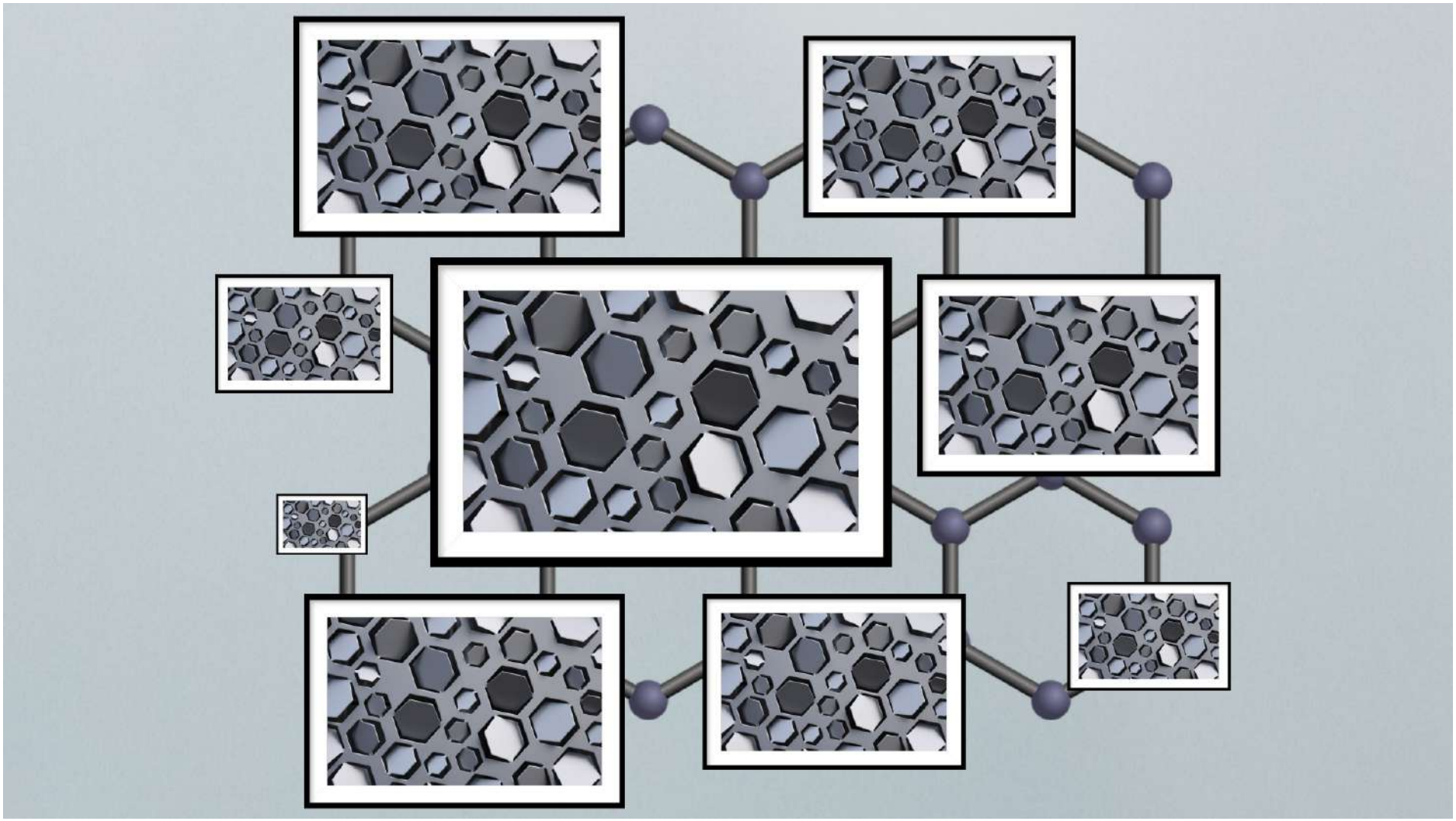
Declarative





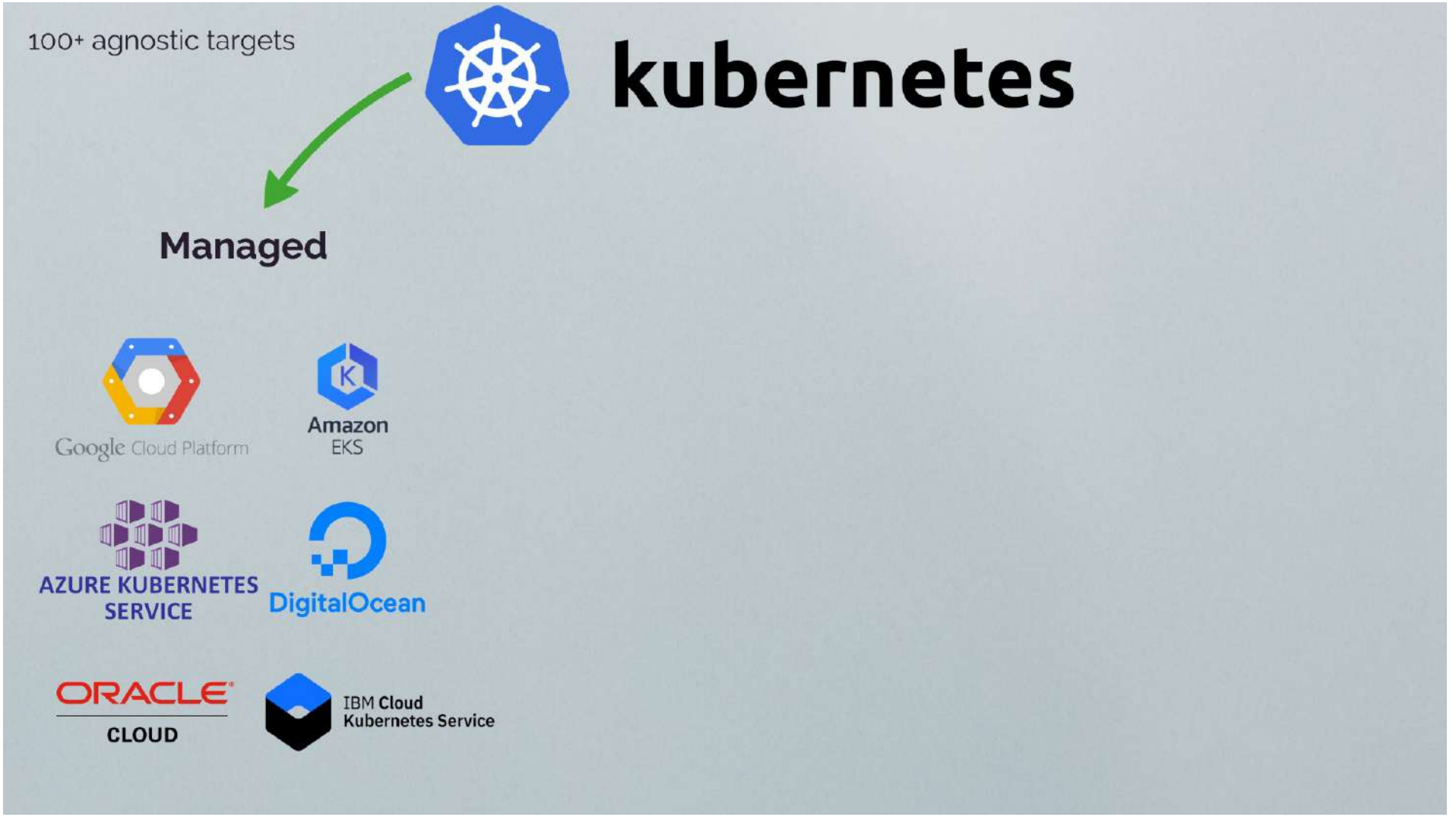






100+ agnostic targets

kubernetes


100+ agnostic targets
kubernetes
Managed
Google Cloud Platform
Amazon
EKS
AZURE KUBERNETES
SERVICE
DigitalOcean
ORACLE
CLOUD
IBM Cloud
Kubernetes Service

100+ agnostic targets

kubernetes

Managed

Google Cloud Platform

Amazon EKS

AZURE KUBERNETES SERVICE

DigitalOcean

ORACLE CLOUD

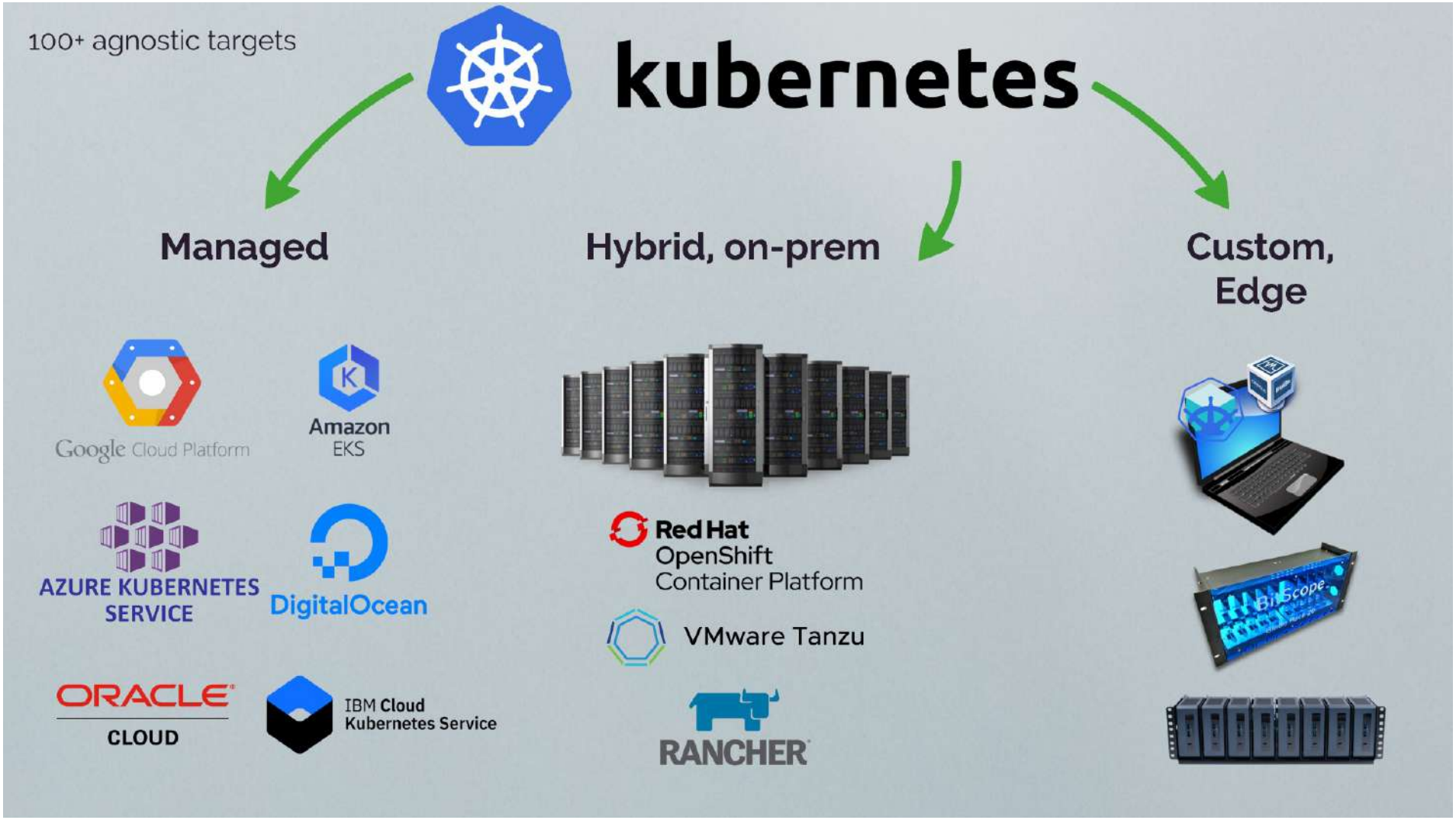
IBM Cloud Kubernetes Service

Hybrid, on-prem

Red Hat OpenShift Container Platform

VMware Tanzu

RANCHER


100+ agnostic targets
kubernetes
Managed
Hybrid, on-prem
Custom, Edge
Google Cloud Platform
Amazon EKS
AZURE KUBERNETES SERVICE
DigitalOcean
ORACLE CLOUD
IBM Cloud Kubernetes Service
Red Hat OpenShift Container Platform
VMware Tanzu
RANCHER


Cloud 2.0
Operating system
Kubernetes
16 Essentials
Open source and
cloud agnostic

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Security

Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Volumes

Security

Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Balancing
Volumes
Security
Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Portability
Balancing
Volumes
Security
Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

Resource$
Portability
Balancing
Volumes
Security
Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Self-healing
- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- Observability
- Self-healing
- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes 16 Essentials

Open source and
cloud agnostic

- Namespacing
- Observability
- Self-healing
- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- High Availability
- Namespacing
- Observability
- Self-healing
- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cloud 2.0
Operating system

Kubernetes
16 Essentials

Open source and
cloud agnostic

- High Availability
- Namespacing
- Observability
- Self-healing
- Abstraction
- Extensibility
- Networking
- Distributed
- Declarative
- Scheduling
- Resource$
- Portability
- Balancing
- Volumes
- Security
- Scalability & Elasticity

Cluster Operating System

Coordinates machines into cluster using shared network to communicate between each server

Each physical or virtual machine is called a **Node**

Cluster Operating System

Coordinates machines into cluster using shared network to communicate between each server

Each physical or virtual machine is called a **Node**

Master nodes run processes that manage cluster

Cluster Operating System

Coordinates machines into cluster using shared network to communicate between each server

Each physical or virtual machine is called a **Node**

Master nodes run processes that manage cluster

Worker nodes run your processes

Cluster Operating System

Coordinates machines into cluster using shared network to communicate between each server

Each physical or virtual machine is called a **Node**

Master nodes run processes that manage cluster

Worker nodes run your processes

Pods are groupings of one or more containers

Cluster Operating System

Coordinates machines into cluster using shared network to communicate between each server

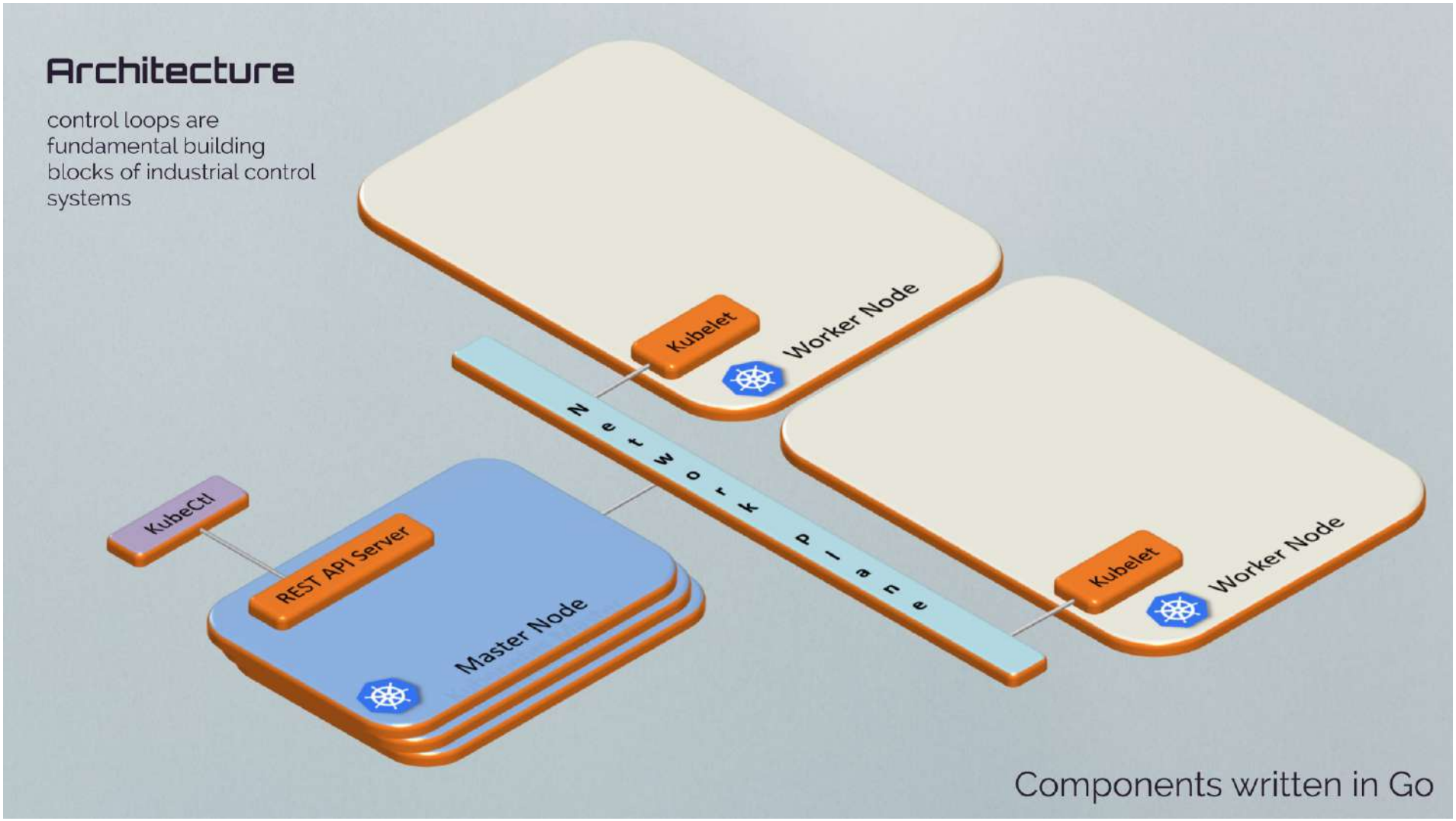
Each physical or virtual machine is called a **Node**

Master nodes run processes that manage cluster

Worker nodes run your processes

Pods are groupings of one or more containers

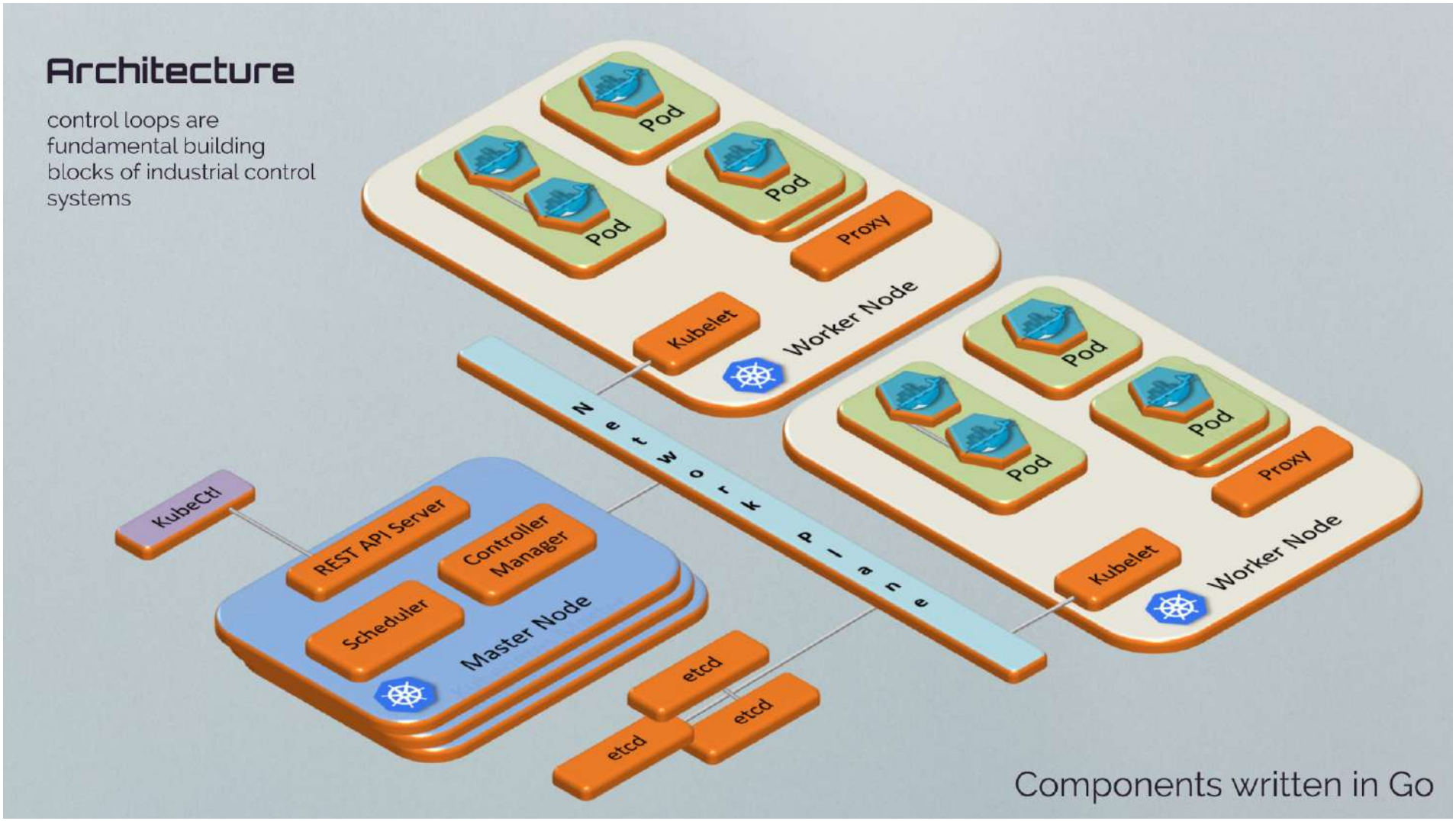
Services load balance between replicated pods


Architecture
control loops are fundamental building blocks of industrial control systems
Kubelet
Worker Node
Network Plane
KubeCtl
REST API Server
Master Node
Kubelet
Worker Node
Components written in Go

Architecture

control loops are fundamental building blocks of industrial control systems

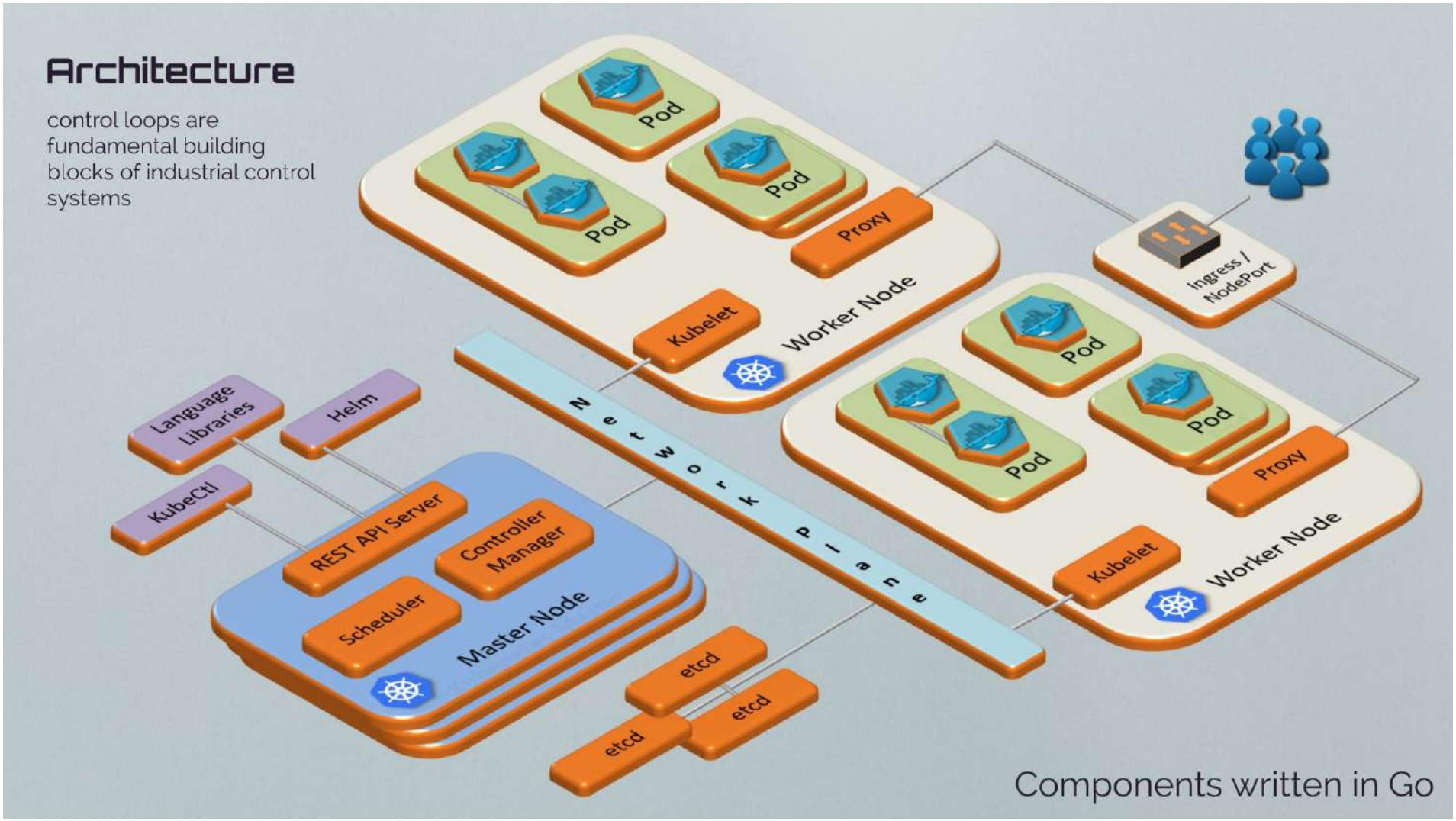
Components written in Go


Architecture
control loops are fundamental building blocks of industrial control systems
Pod
Pod
Pod
Proxy
Kubelet
Worker Node
Pod
Pod
Pod
Proxy
Kubelet
Worker Node
Network Plane
KubeCtl
REST API Server
Controller Manager
Scheduler
Master Node
etcd
etcd
etcd
Components written in Go

Architecture

control loops are fundamental building blocks of industrial control systems

Components written in Go


Architecture
control loops are fundamental building blocks of industrial control systems
Pod
Pod
Pod
Proxy
Kubelet
Worker Node
Ingress / NodePort
Pod
Pod
Pod
Proxy
Kubelet
Worker Node
Network plane
Language Libraries
Helm
KubeCtl
REST API Server
Controller Manager
Scheduler
Master Node
etcd
etcd
etcd
Components written in Go

Kubernetes truths stored here

Distributed key-value store

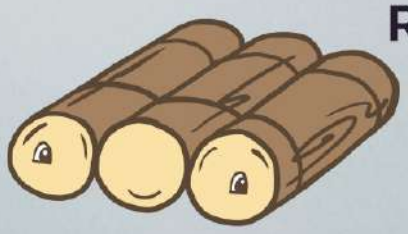
Partition tolerant consensus algorithm

Readable once consensus reached

High available solution at minimum 3 nodes

Fast reads, slower writes

etcd



Raft Consensus Algorithm

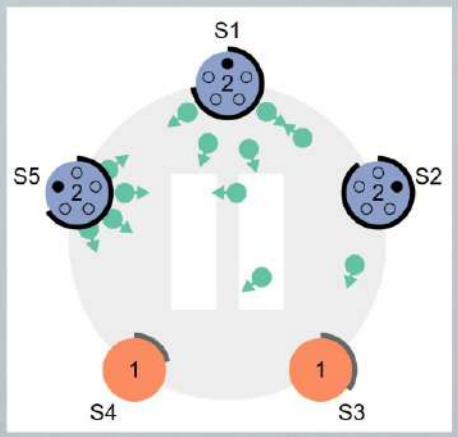
Interactive tutorials:
https://raft.github.io
http://thesecretlivesofdata.com/raft

Kubernetes truths stored here

Distributed key-value store

Partition tolerant consensus algorithm

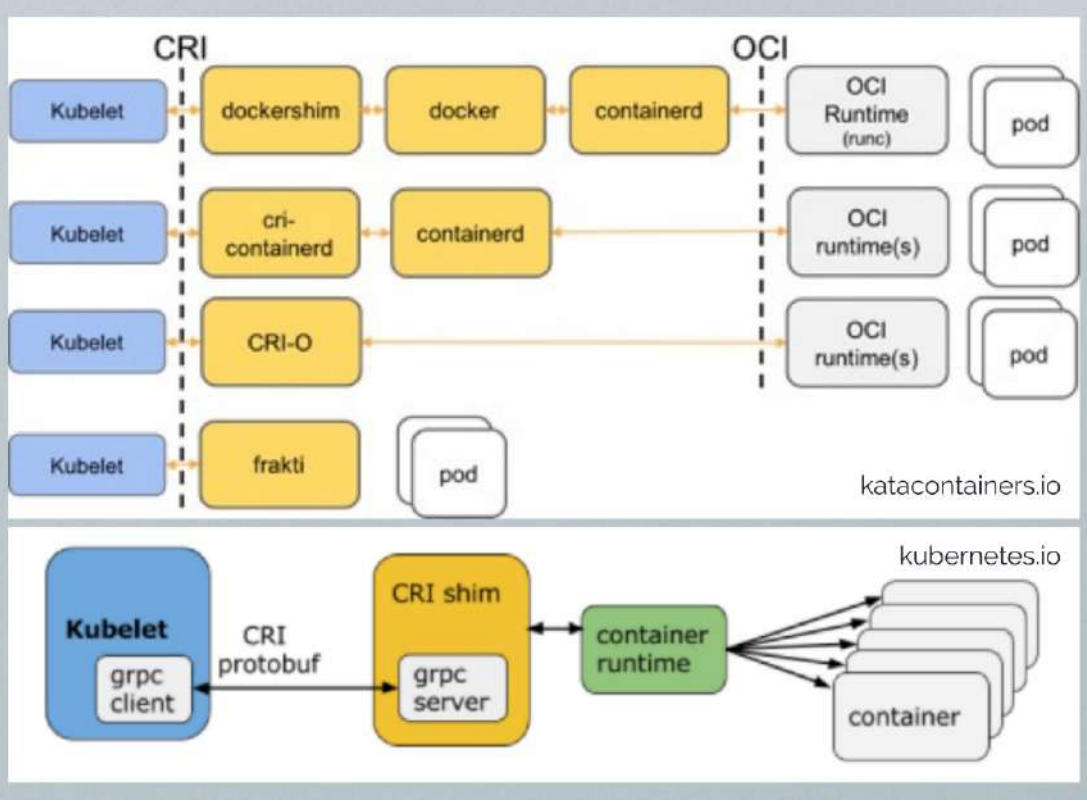
Readable once consensus reached



High available solution at minimum 3 nodes

Fast reads, slower writes

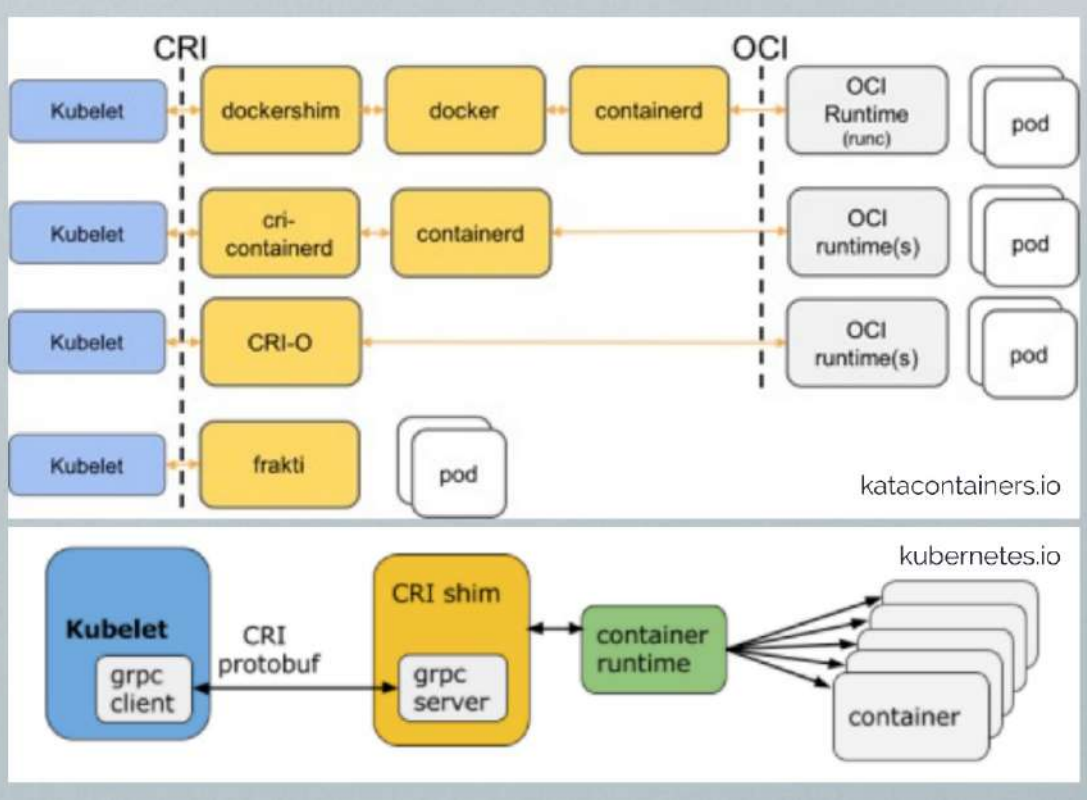
CRI and OCI


CRI
OCI
Kubelet
dockershim
docker
containerd
OCI Runtime (runc)
pod
Kubelet
cri-containerd
containerd
OCI runtime(s)
pod
Kubelet
CRI-O
OCI runtime(s)
pod
Kubelet
frakti
pod
katacontainers.io
kubernetes.io
Kubelet
grpc client
CRI protobuf
CRI shim
grpc server
container runtime
container

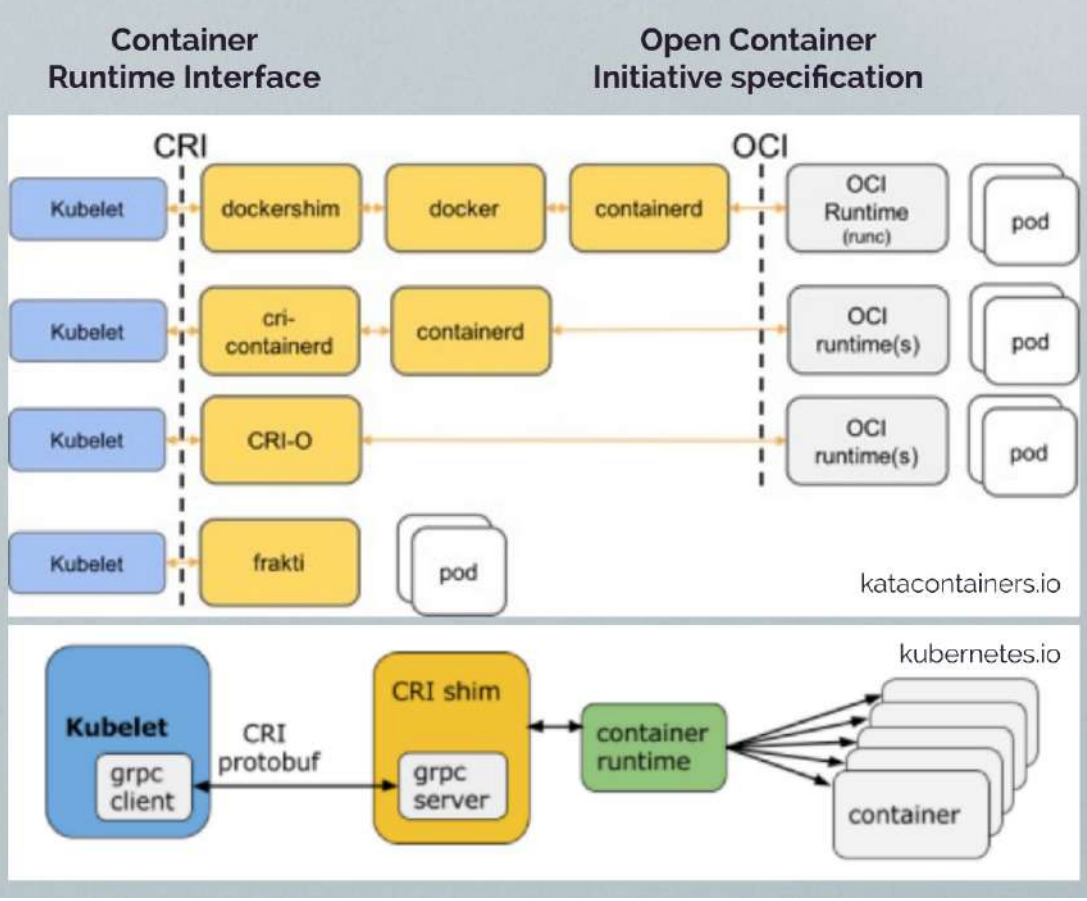
CRI and OCI

OPEN CONTAINER INITIATIVE

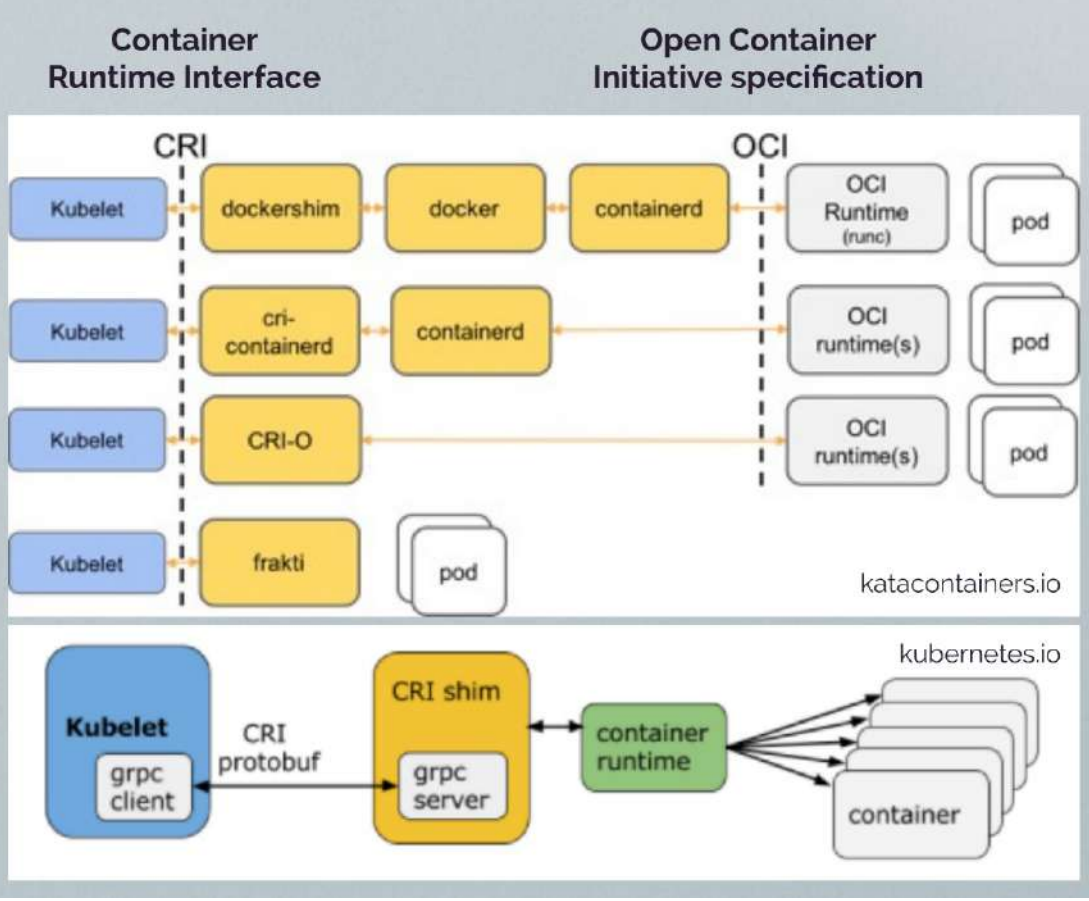
Container Runtime Interface



CRI and OCI


Container
Runtime Interface
Open Container
Initiative specification
CRI
OCI
Kubelet
dockershim
docker
containerd
OCI
Runtime
(runc)
pod
Kubelet
cri-
containerd
containerd
OCI
runtime(s)
pod
Kubelet
CRI-O
OCI
runtime(s)
pod
Kubelet
frakti
pod
katacontainers.io
kubernetes.io
Kubelet
grpc
client
CRI
protobuf
CRI shim
grpc
server
container
runtime
container

CRI and OCI



OPEN CONTAINER INITIATIVE

OCI container runtimes with Kubelet CRI

- Docker
- cri-o (OpenShift)
- cri-containerd
  - **runc**
  - gVisor
  - Kata containers
  - Firecracker
  - Nabla containers
- crun (c based)
- Rktlet (CRI for Rkt)
- rktnetes
- Frakti
- Singularity
- * Virtual Kubelet

https://landscape.cncf.io/category=container-runtime&format=card-mode&grouping=category

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Object classification

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Object classification

Associated data

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Object classification

Associated data

Specific object details

Declarative

Instructions for a state machine

- You declare, not script
- Modeling Resources with YAMLs and Charts

YAML Human friendly data serialization standard

HELM Package manager for Kubernetes

VSCode and **Intellij** extensions for writing YAMLs

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace apps-collection -o YAML
```

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Object classification

Associated data

Specific object details

Declaring Pods

Declaring Pods

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

Declaring Pods

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

Declaring Pods

Pod
Container(s)

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.19.2
    ports:
    - containerPort: 80
```

Declaring Pods

Pod
Container(s)

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

Declaring Pods

ReplicaSet

Pod
Container(s)

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.2
        ports:
        - containerPort: 80
```

Declaring Pods

ReplicaSet

Pod
Container(s)

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

Declaring Pods

Deployment

ReplicaSet

Pod
Container(s)

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

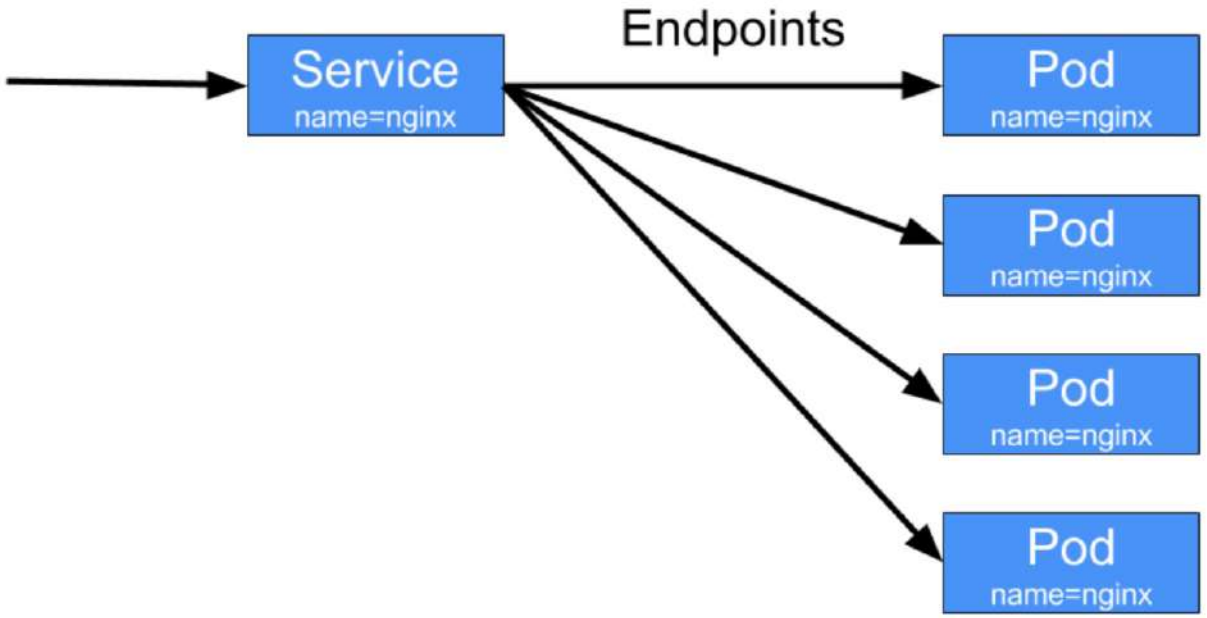
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.2
        ports:
        - containerPort: 80
```

```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ kubectl run my-nginx --image=nginx:1.19.2 --port 80
```

Declaring Service Associated to Pods

Labels used to declare associations



Declaring Service Associated to Pods

Labels used to declare associations

Declaring Service Associated to Pods

Labels used to declare associations

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.1
        ports:
        - containerPort: 80
          name: nginx-pod-port
```

Declaring Service Associated to Pods

Labels used to declare associations

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webserver
  labels:
    app: webserver
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
  - port: 31000
    targetPort: nginx-pod-port
    protocol: TCP
    name: web
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.1
        ports:
        - containerPort: 80
          name: nginx-pod-port
```

Declaring Service Associated to Pods

Labels used to declare associations

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webserver
  labels:
    app: webserver
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
  - port: 31000
    targetPort: nginx-pod-port
    protocol: TCP
    name: web
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.1
        ports:
        - containerPort: 80
          name: nginx-pod-port
```

Declaring ConfigMaps and Secrets

Data can be linked to:

- Environment variables
- Command line parameters
- File mounts (read-only)

```
apiVersion: v1
kind: Pod
metadata:
  name: passable
spec:
  containers:
    - name: question-app
      image: grail-seeker
      env:
        - name: swallow-bird-type
          value: "African"
        - name: Q1
          valueFrom:
            configMapKeyRef:
              name: questions
              key: q1
        - name: Q2
          valueFrom:
            configMapKeyRef:
              name: questions
              key: q2
```

Declaring ConfigMaps and Secrets

Data can be linked to:

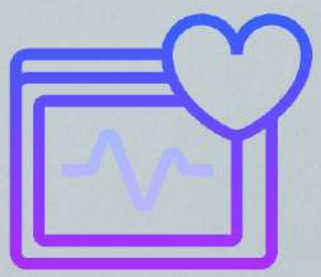
- Environment variables
- Command line parameters
- File mounts (read-only)

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: questions
data:
  q1: "What is your quest?"
  q2: "What is your name?"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: passable
spec:
  containers:
    - name: question-app
      image: grail-seeker
      env:
        - name: swallow-bird-type
          value: "African"
        - name: Q1
          valueFrom:
            configMapKeyRef:
              name: questions
              key: q1
        - name: Q2
          valueFrom:
            configMapKeyRef:
              name: questions
              key: q2
```

Declarative

Probes



```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

```yaml
readinessProbe:
  exec:
    command:
    - cat
    - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

```go
http.HandleFunc("/healthz", func(w http.ResponseWriter, r *http.Request) {
    duration := time.Now().Sub(started)
    if duration.Seconds() > 10 {
        w.WriteHeader(500)
        w.Write([]byte(fmt.Sprintf("error: %v", duration.Seconds())))
    } else {
        w.WriteHeader(200)
        w.Write([]byte("ok"))
    }
})
```

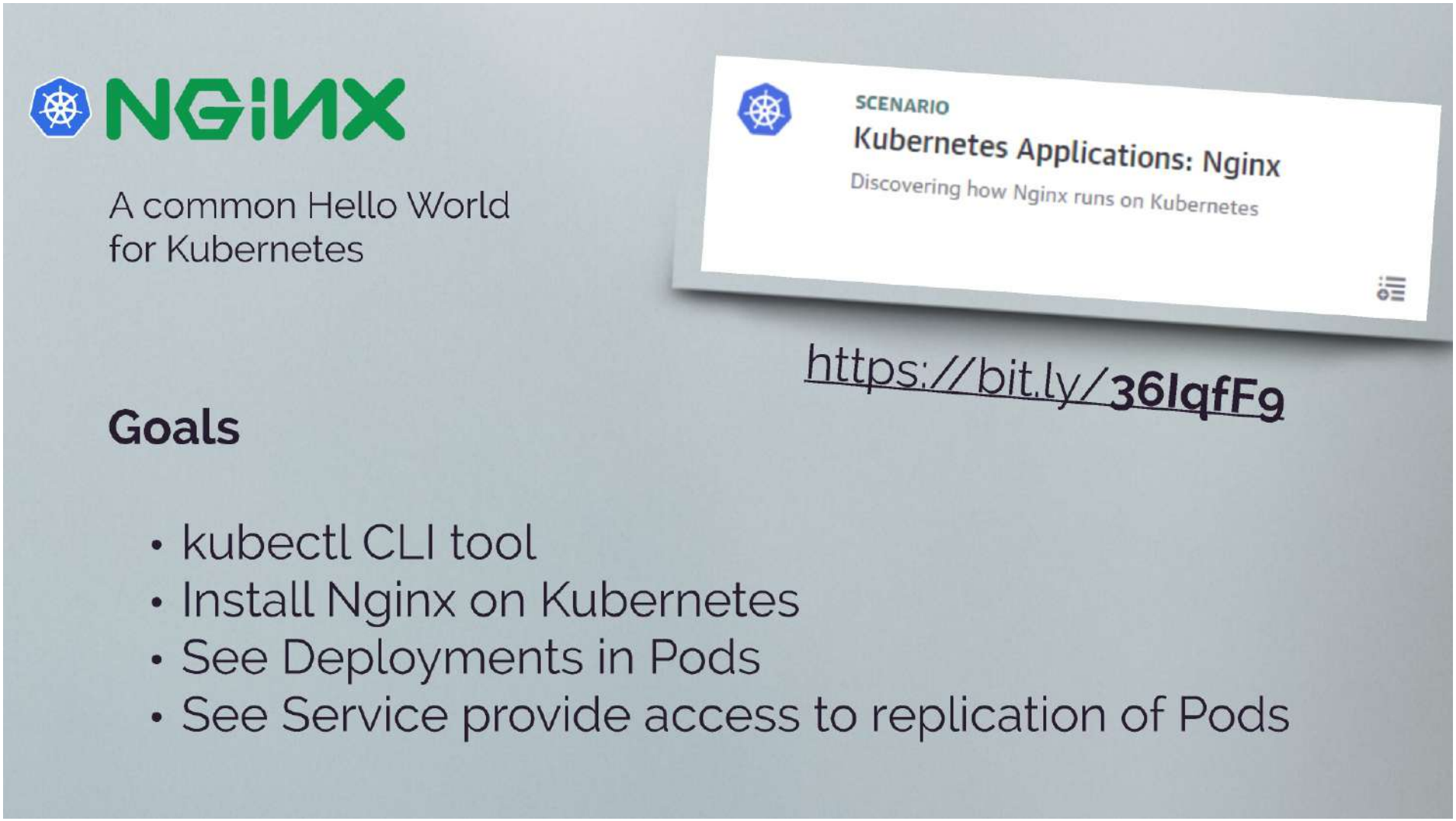

Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community

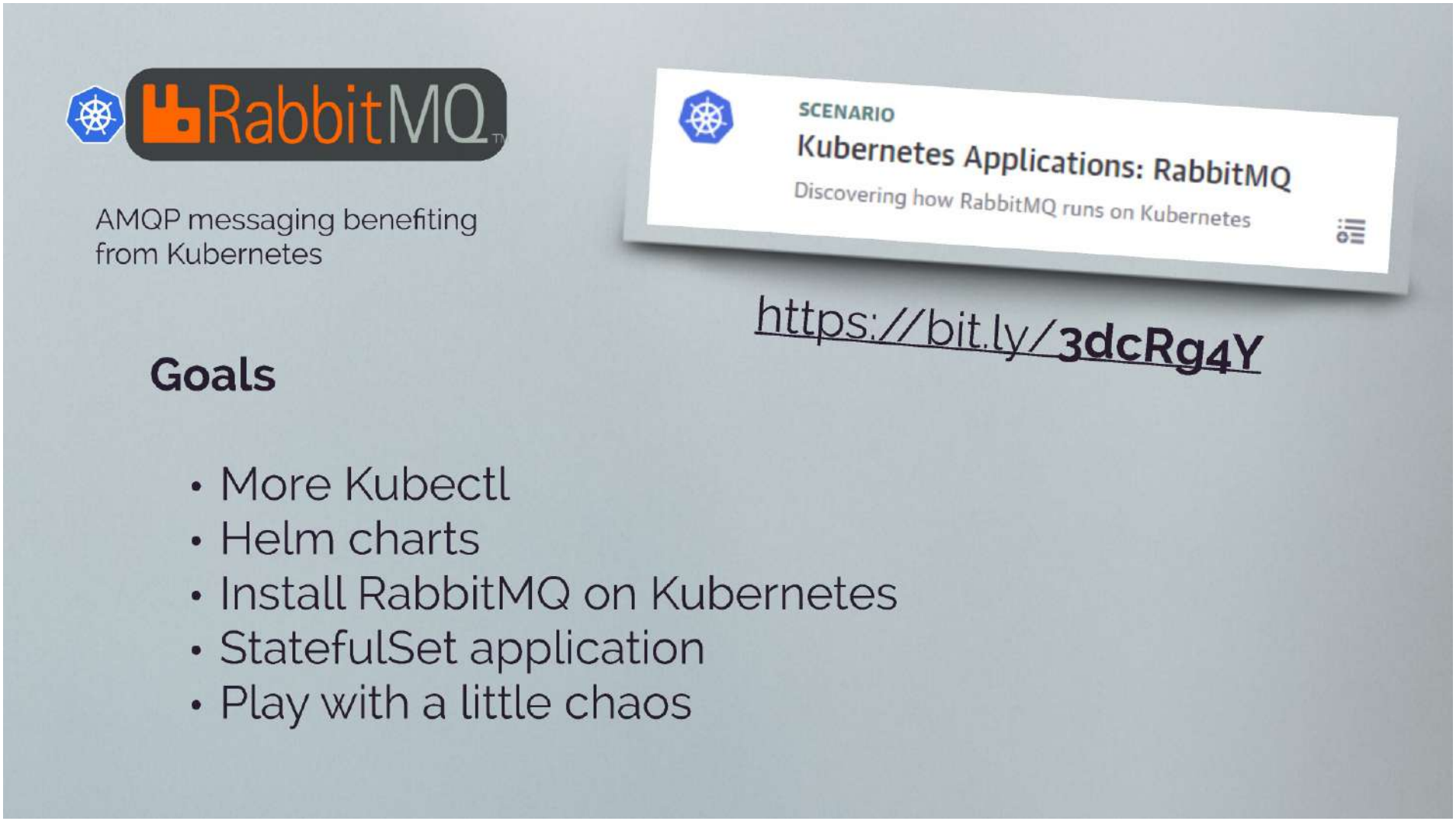
First App

https://bit.ly/33Cd4Uw

Goals

- Command line tool
- Dashboard
- Run app
- Scale app
- Load balance
- Test resilience
- Roll out new version of app


NGINX
A common Hello World
for Kubernetes
SCENARIO
Kubernetes Applications: Nginx
Discovering how Nginx runs on Kubernetes
https://bit.ly/36lqfF9
Goals
• kubectl CLI tool
• Install Nginx on Kubernetes
• See Deployments in Pods
• See Service provide access to replication of Pods


RabbitMQ
AMQP messaging benefiting from Kubernetes
SCENARIO
Kubernetes Applications: RabbitMQ
Discovering how RabbitMQ runs on Kubernetes
https://bit.ly/3dcRg4Y
Goals
• More Kubectl
• Helm charts
• Install RabbitMQ on Kubernetes
• StatefulSet application
• Play with a little chaos

R Language

https://bit.ly/3jEK5EP

Goals

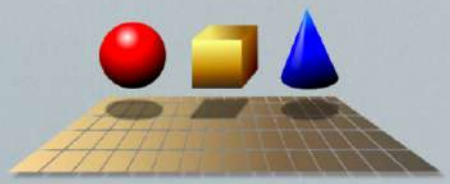
- Basics of kubectl CLI tool
- Install Shiny R applications on Kubernetes
- Containers are deployed as Deployments in Pods
- Service can provide access to a Pod


Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community

Kubernetes Resources and Objects

Declaring State

Kubernetes resources
represent state of cluster.

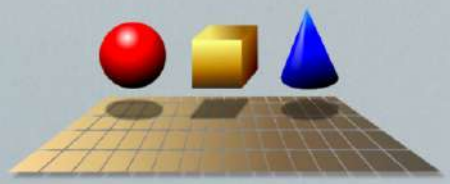


Kubernetes Resources and Objects

Declaring State

Kubernetes resources represent state of cluster.

Resources declared in Kubernetes manifests (YAMLs) as "Kinds".

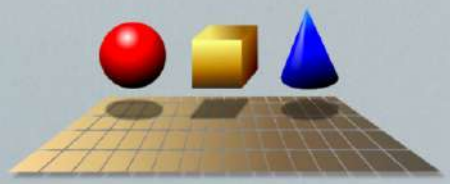


Kubernetes Resources and Objects

Declaring State

Kubernetes resources represent state of cluster.

Resources declared in Kubernetes manifests (YAMLs) as "Kinds".



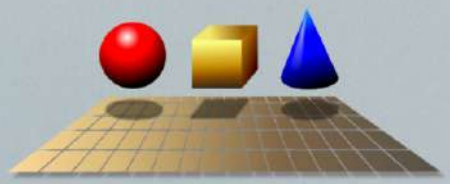
Instantiated resources are persisted objects across the cluster

Kubernetes Resources and Objects

Declaring State

Kubernetes resources represent state of cluster.

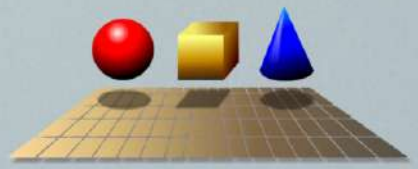
Resources declared in Kubernetes manifests (YAMLs) as "Kinds".



Instantiated resources are persisted objects across the cluster

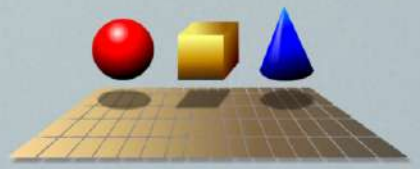
Objects are managed via API and often through kubectl.

Kubernetes Objects



They will be
Kind to you

Kubernetes Objects



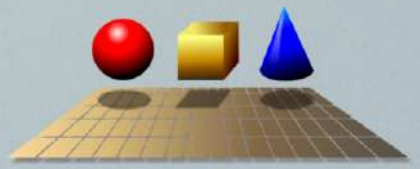
They will be **Kind** to you

Pod Contructs
- Pod
- ReplicaSet
- Deployments
- StatefulSet
- DaemonSet
- Job (batch)
- CronJob (batch)

Context / Data
- ConfigMap
- Secrets

Networking
- Ingress
- Service
- NetworkPolicy
- Endpoints

Kubernetes Objects



They will be **Kind** to you

Pod Contructs
- Pod
- ReplicaSet
- Deployments
- StatefulSet
- DaemonSet
- Job (batch)
- CronJob (batch)

Context / Data
- ConfigMap
- Secrets

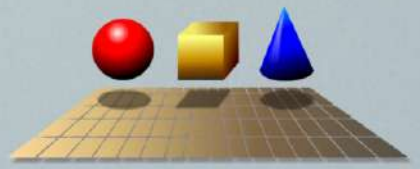
Networking
- Ingress
- Service
- NetworkPolicy
- Endpoints

Persistence
- Volumes
- Persistent Volumes
- StorageClass
- CSIDrivers
- CSINodes

RBAC
- ServiceAccount
- Role
- RoleBinding

Cluster Scope
- Node
- Namespace
- CustomResourceDefinition

Kubernetes Objects



They will be **Kind** to you

Pod Contructs
- Pod
- ReplicaSet
- Deployments
- StatefulSet
- DaemonSet
- Job (batch)
- CronJob (batch)

Context / Data
- ConfigMap
- Secrets

Networking
- Ingress
- Service
- NetworkPolicy
- Endpoints

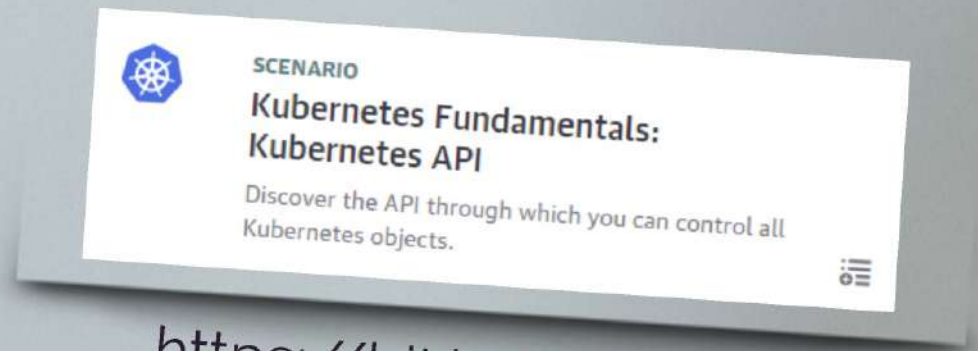
Persistence
- Volumes
- Persistent Volumes
- StorageClass
- CSIDrivers
- CSINodes

RBAC
- ServiceAccount
- Role
- RoleBinding

Cluster Scope
- Node
- Namespace
- CustomResourceDefinition

Other
- Resource Quotas
- Horizontal Pod Auto Scalar
- Pod Disruption Budgets
- Leases
- Events
- ...and more

API and Objects

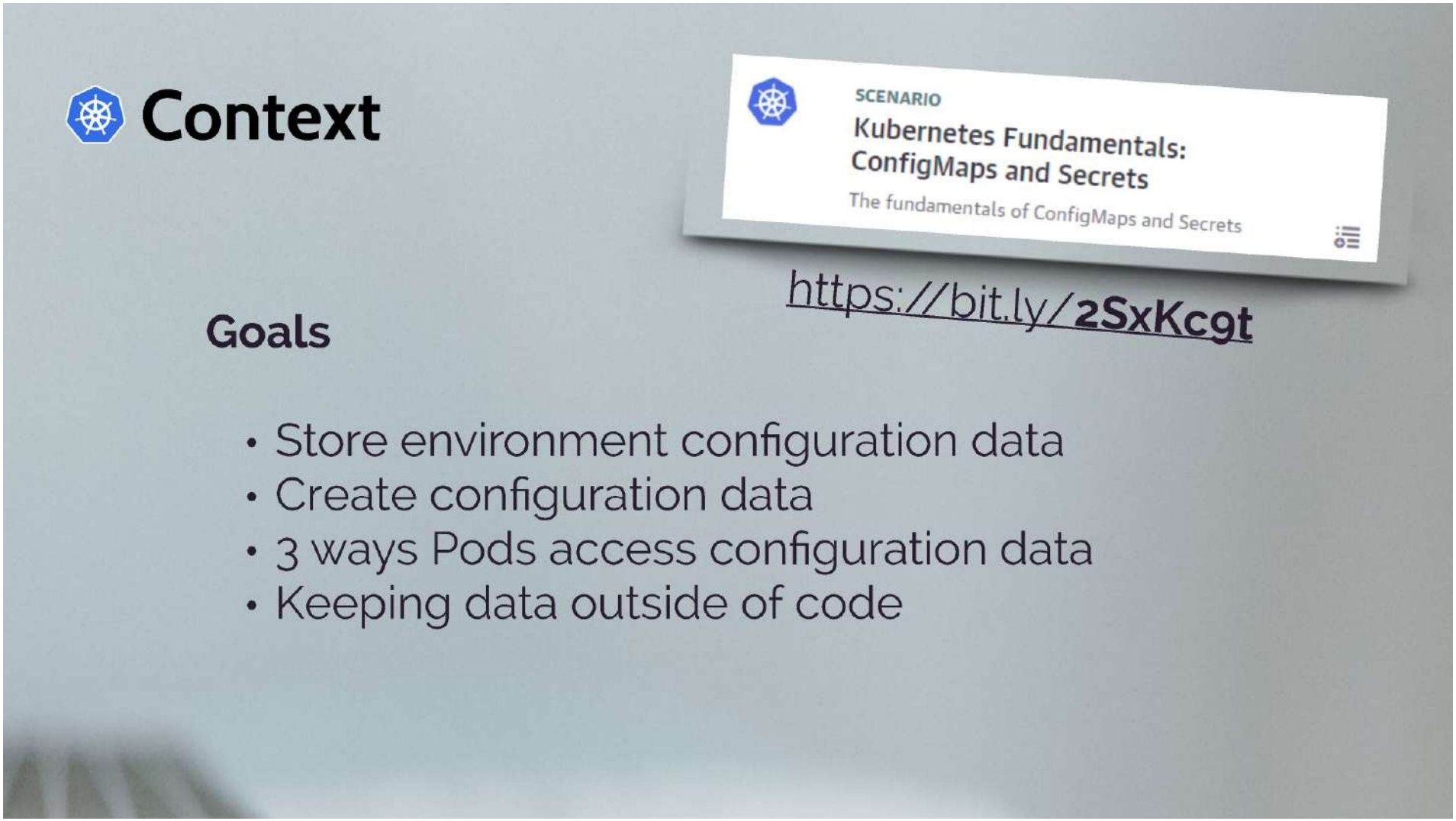


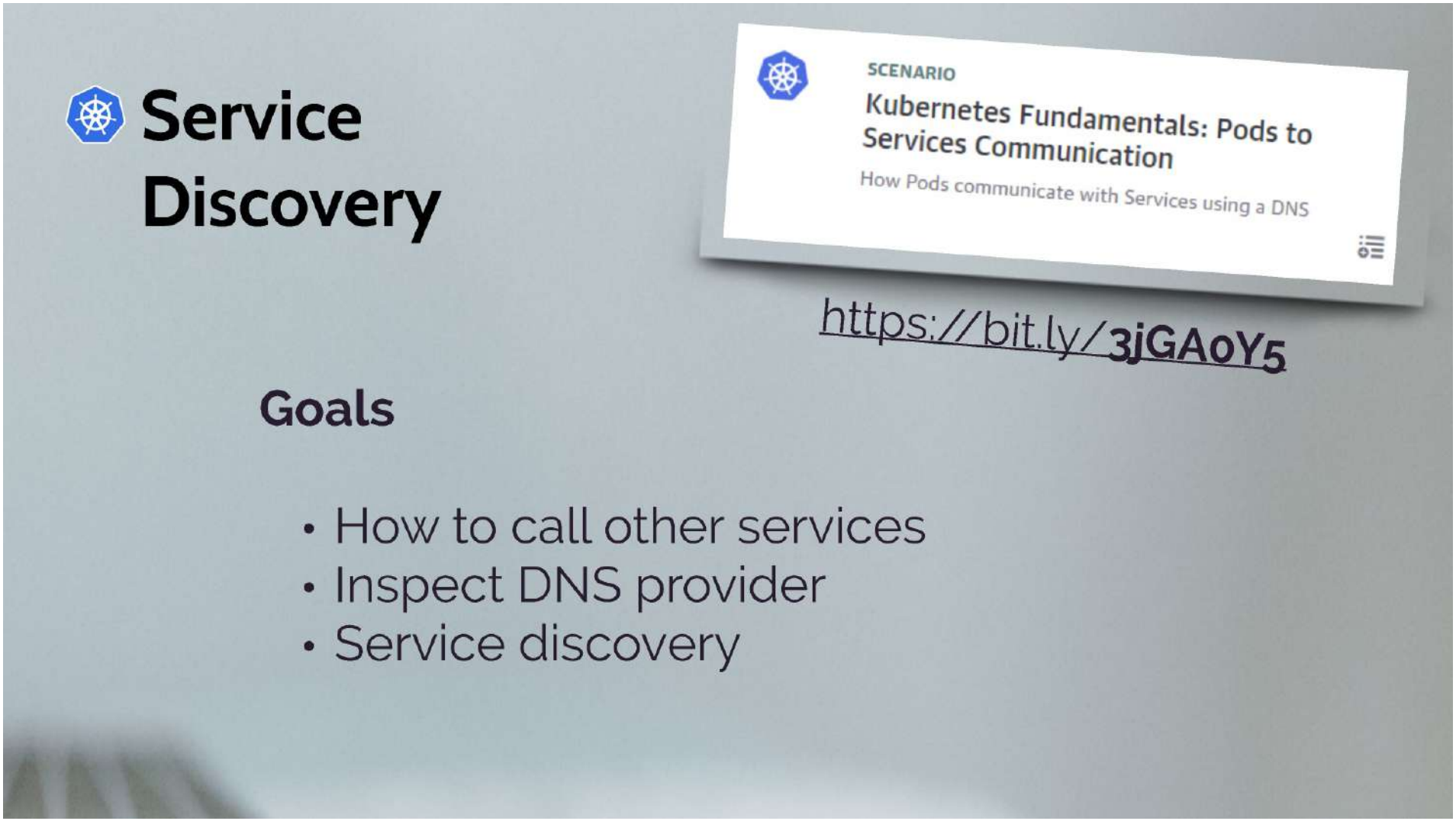
https://bit.ly/3nxcnUr

Goals

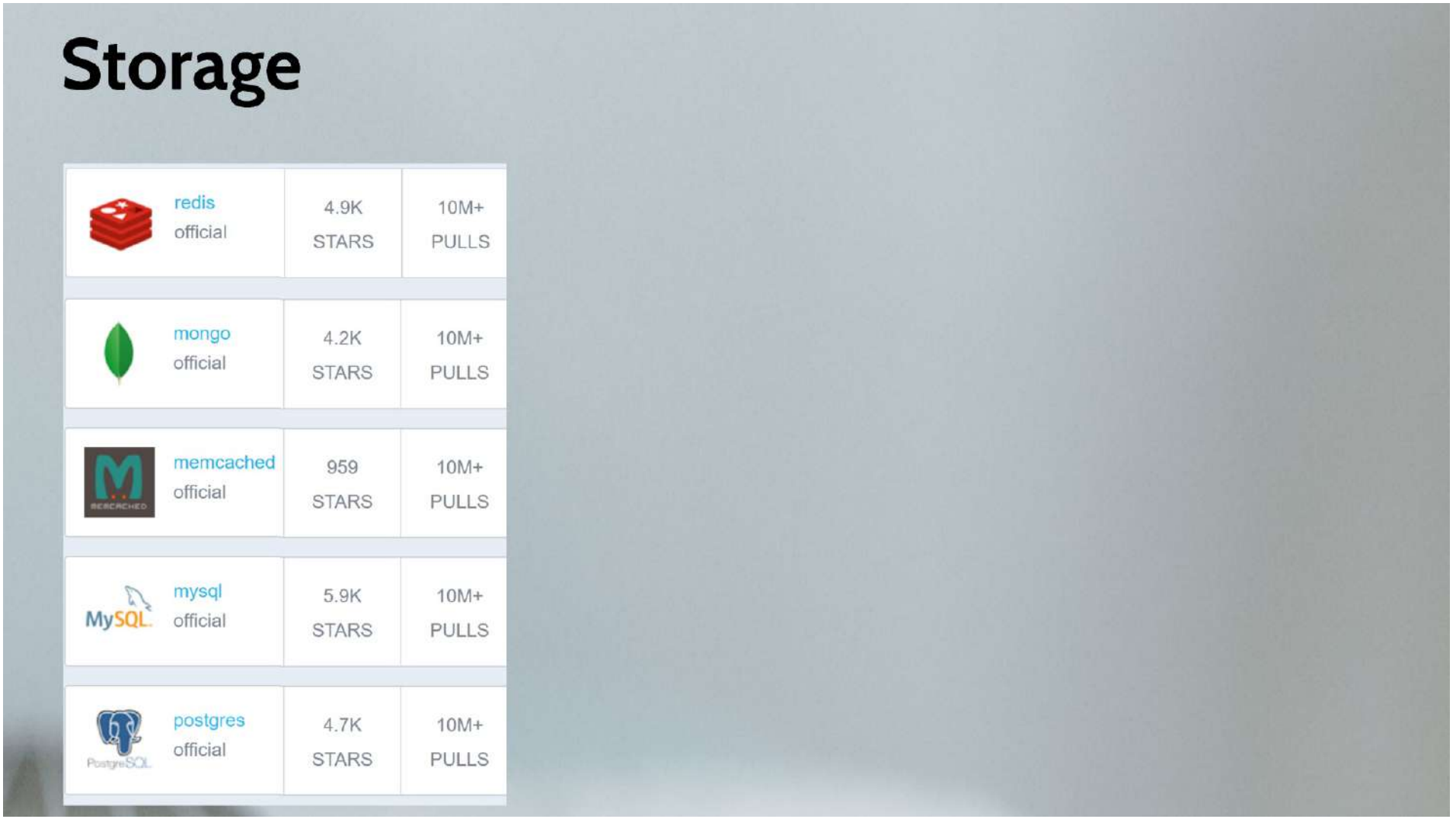
- Access API via kubectl
- Introspect objects in cluster via API
- Access cluster API locally through a Proxy
- Discover api-resources and api-versions
- Discover Explain and Describe commands

```
$ kubectl get --raw
```


Context
SCENARIO
Kubernetes Fundamentals:
ConfigMaps and Secrets
The fundamentals of ConfigMaps and Secrets
https://bit.ly/2SxKc9t
Goals
• Store environment configuration data
• Create configuration data
• 3 ways Pods access configuration data
• Keeping data outside of code


Service Discovery
SCENARIO
Kubernetes Fundamentals: Pods to Services Communication
How Pods communicate with Services using a DNS
https://bit.ly/3jGAoY5
Goals
• How to call other services
• Inspect DNS provider
• Service discovery

Storage


redis
official
4.9K
STARS
10M+
PULLS
mongo
official
4.2K
STARS
10M+
PULLS
memcached
official
959
STARS
10M+
PULLS
mysql
official
5.9K
STARS
10M+
PULLS
postgres
official
4.7K
STARS
10M+
PULLS

Storage

	redis official	4.9K STARS	10M+ PULLS
	mongo official	4.2K STARS	10M+ PULLS
	memcached official	959 STARS	10M+ PULLS
	mysql official	5.9K STARS	10M+ PULLS
	postgres official	4.7K STARS	10M+ PULLS

Databases
Relational, NoSQL, Graph, Time series, +...

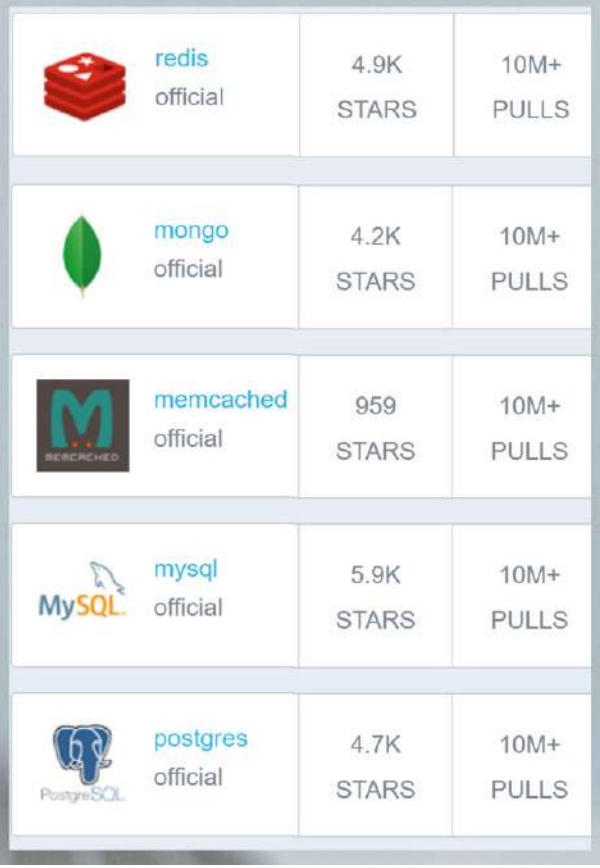
Storage

redis official	4.9K STARS	10M+ PULLS
mongo official	4.2K STARS	10M+ PULLS
memcached official	959 STARS	10M+ PULLS
mysql official	5.9K STARS	10M+ PULLS
postgres official	4.7K STARS	10M+ PULLS

Databases
Relational, NoSQL, Graph, Time series, +...

Memory Centralized
Redis, H2, eXtremeDB, memcached...
wikipedia.org/wiki/List_of_in-memory_databases

Storage



Databases
Relational, NoSQL, Graph, Time series, +...

Memory Centralized
Redis, H2, eXtremeDB, memcached...
wikipedia.org/wiki/List_of_in-memory_databases

Key-value Datastores, distributed consensus
etcd, Consul, Zookeeper, Hazlecast

Storage

redis	official	4.9K STARS	10M+ PULLS
mongo	official	4.2K STARS	10M+ PULLS
memcached	official	959 STARS	10M+ PULLS
mysql	official	5.9K STARS	10M+ PULLS
postgres	official	4.7K STARS	10M+ PULLS

CONTAINER STORAGE INTERFACE

CSI Production Drivers for Kubernetes (82)
https://kubernetes-csi.github.io/docs/drivers.html

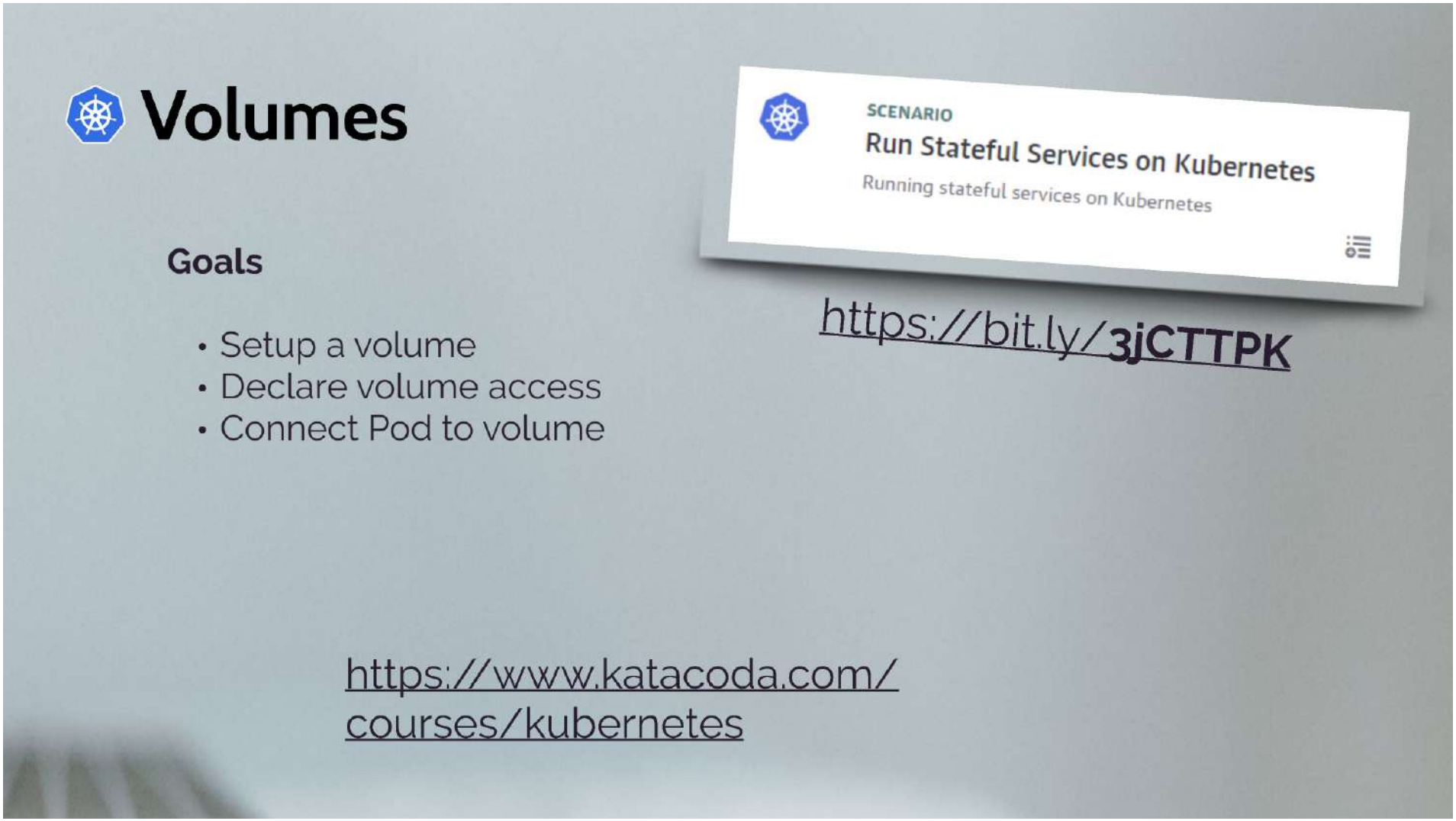
Databases
Relational, NoSQL, Graph, Time series, +...

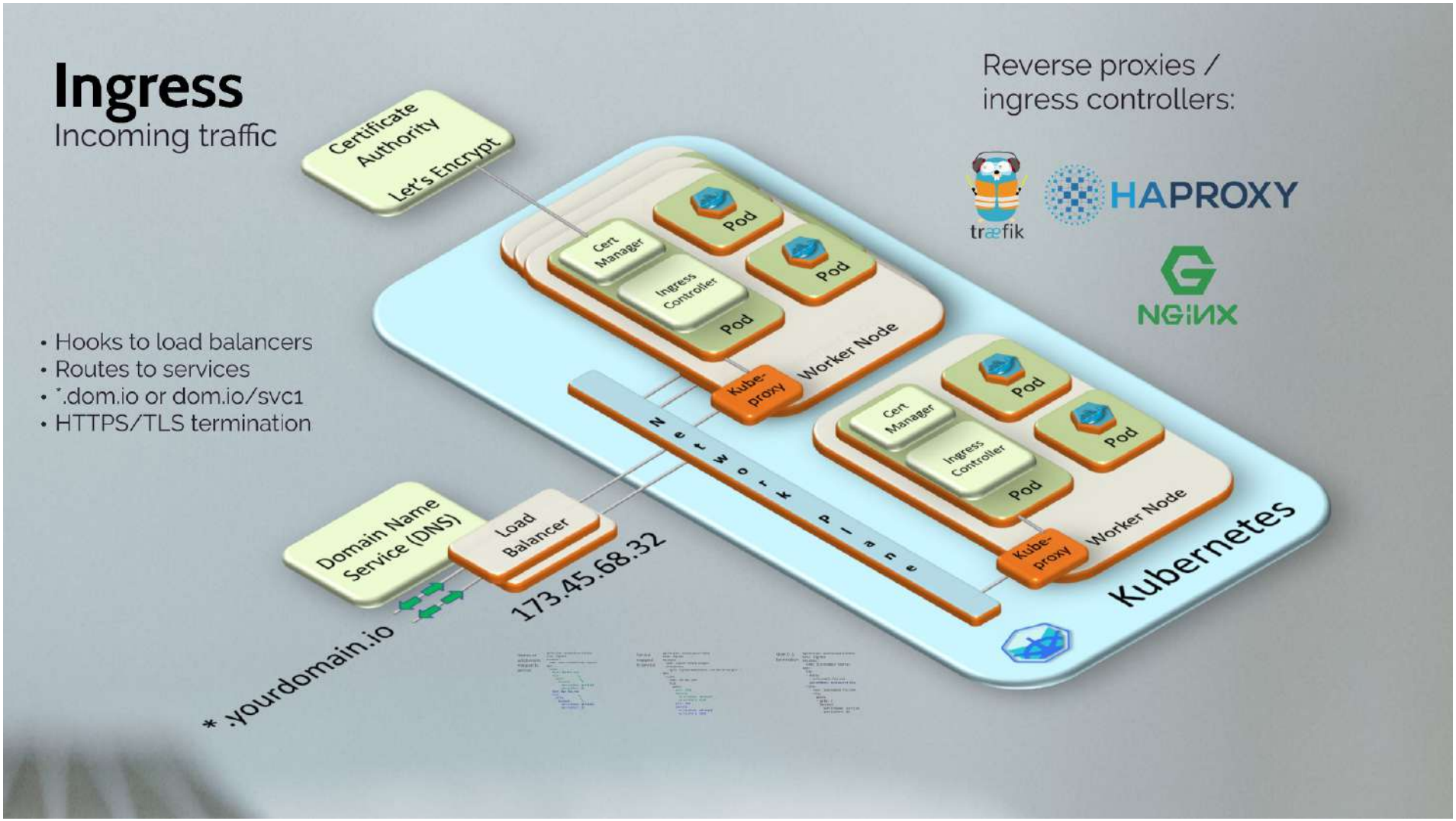
Memory Centralized
Redis, H2, eXtremeDB, memcached...
wikipedia.org/wiki/List_of_in-memory_databases

Key-value Datastores, distributed consensus
etcd, Consul, Zookeeper, Hazlecast

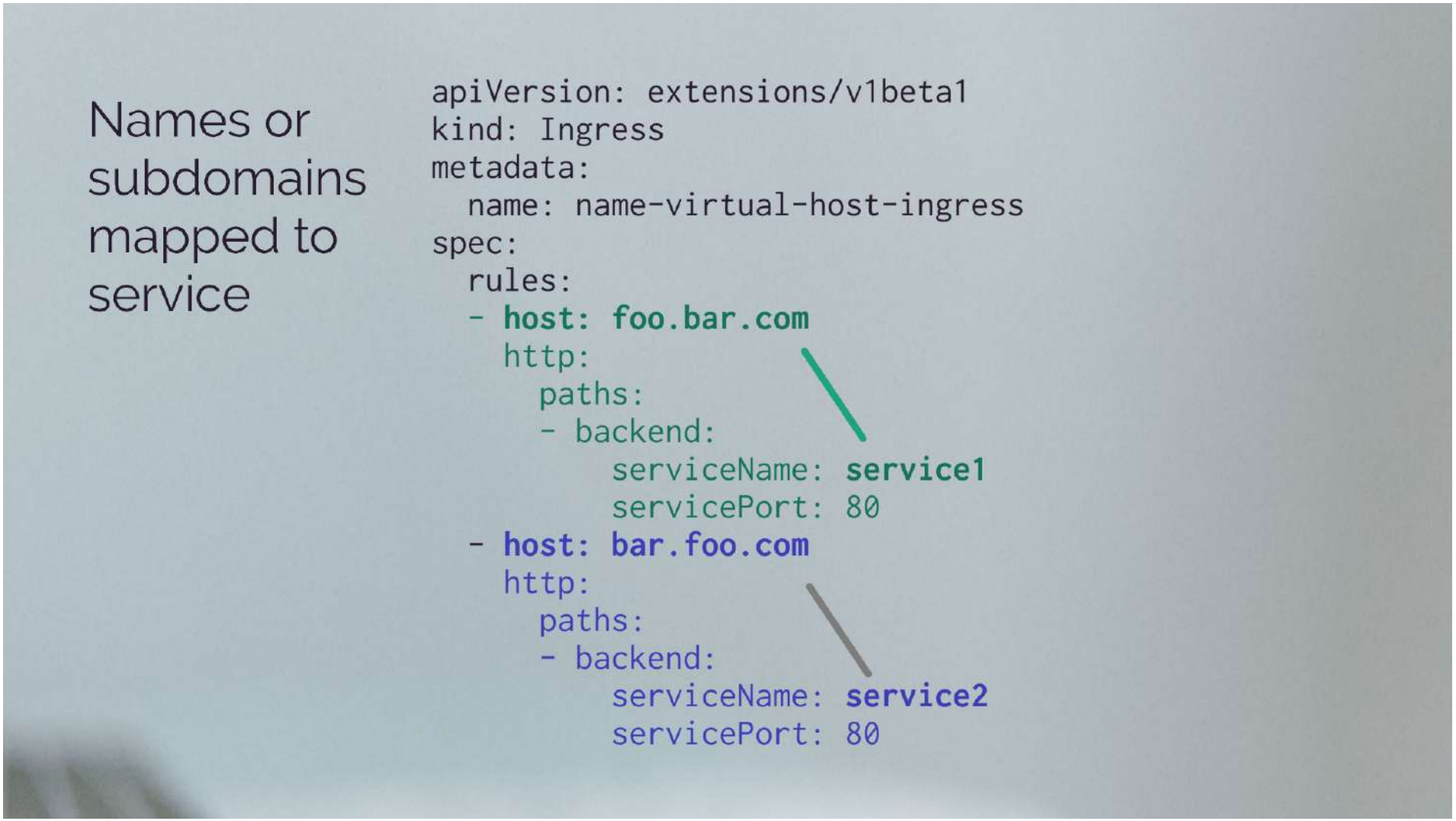
Kind: PersistentVolume
Mounts: NFS, iSCSI, CephFS, RDB, FC, Flocker...
+ Minio, S3, Github

Volume Plugin
AWSElasticBlockStore
AzureFile
AzureDisk
CephFS
Cinder
FC
FlexVolume
Flocker
GCEPersistentDisk
Glusterfs
HostPath
iSCSI
PhotonPersistentDisk
Quobyte
NFS
RBD
VsphereVolume
PortworxVolume
ScaleIO
StorageOS


Volumes
SCENARIO
Run Stateful Services on Kubernetes
Running stateful services on Kubernetes
Goals
• Setup a volume
• Declare volume access
• Connect Pod to volume
https://bit.ly/3jCTTPK
https://www.katacoda.com/courses/kubernetes


Ingress
Incoming traffic
Reverse proxies / ingress controllers:
Certificate Authority
Let's Encrypt
træfik
HAPROXY
NGINX
• Hooks to load balancers
• Routes to services
• *.dom.io or dom.io/svc1
• HTTPS/TLS termination
Cert Manager
Ingress Controller
Pod
Pod
Pod
Kube-proxy
Worker Node
Network Plane
Cert Manager
Ingress Controller
Pod
Pod
Pod
Kube-proxy
Worker Node
Kubernetes
Domain Name Service (DNS)
Load Balancer
173.45.68.32
* .yourdomain.io

Names or subdomains mapped to service



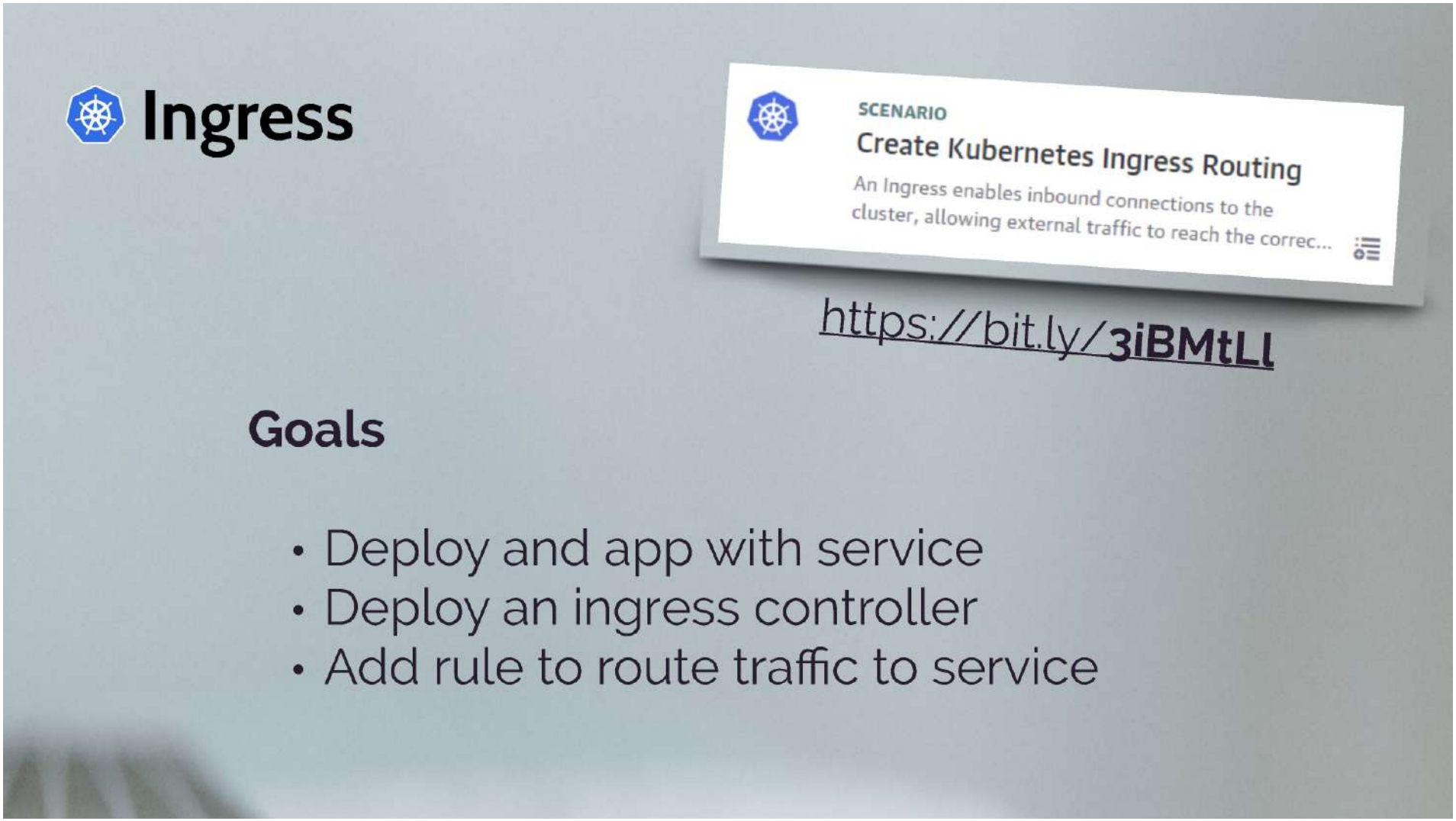
```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: service1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: service2
          servicePort: 80
```

Fan out mapped to service

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: service1
          servicePort: 4200
      - path: /bar
        backend:
          serviceName: service2
          servicePort: 8080
```

With TLS termination

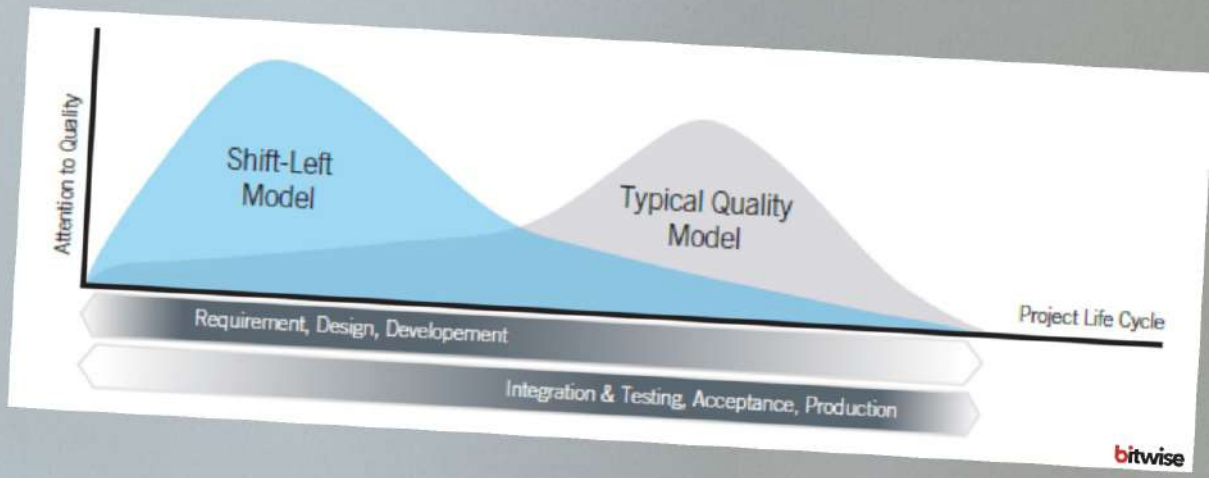
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
  - hosts:
    - sslexample.foo.com
    secretName: testsecret-tls
  rules:
    - host: sslexample.foo.com
      http:
        paths:
        - path: /
          backend:
            serviceName: service1
            servicePort: 80
```


Ingress
SCENARIO
Create Kubernetes Ingress Routing
An Ingress enables inbound connections to the cluster, allowing external traffic to reach the correc...
https://bit.ly/3iBMtLl
Goals
• Deploy and app with service
• Deploy an ingress controller
• Add rule to route traffic to service


Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community

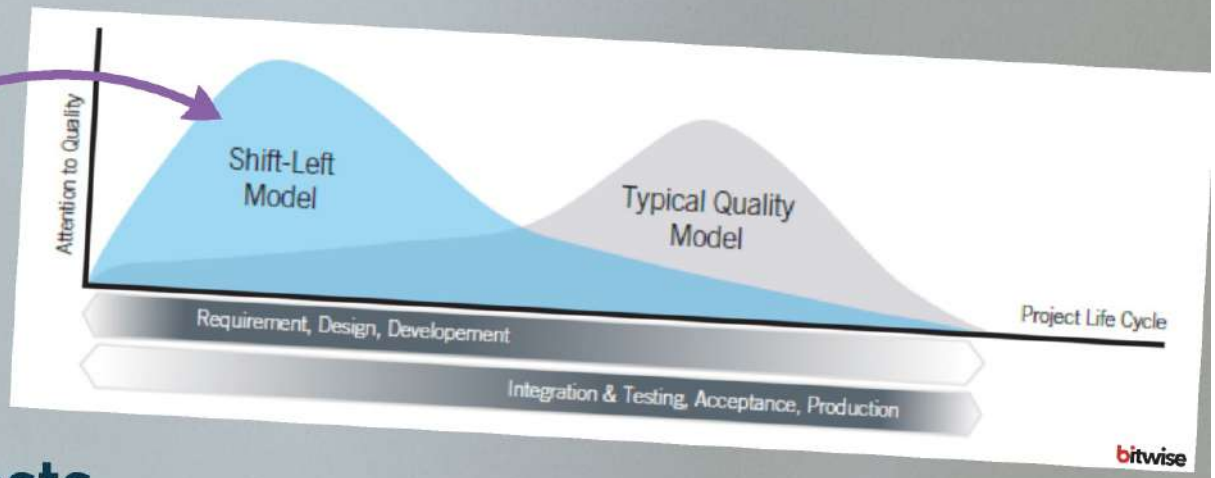
Infrastructure as Code

Drive for Shift-left



Infrastructure as Code

Drive for Shift-left

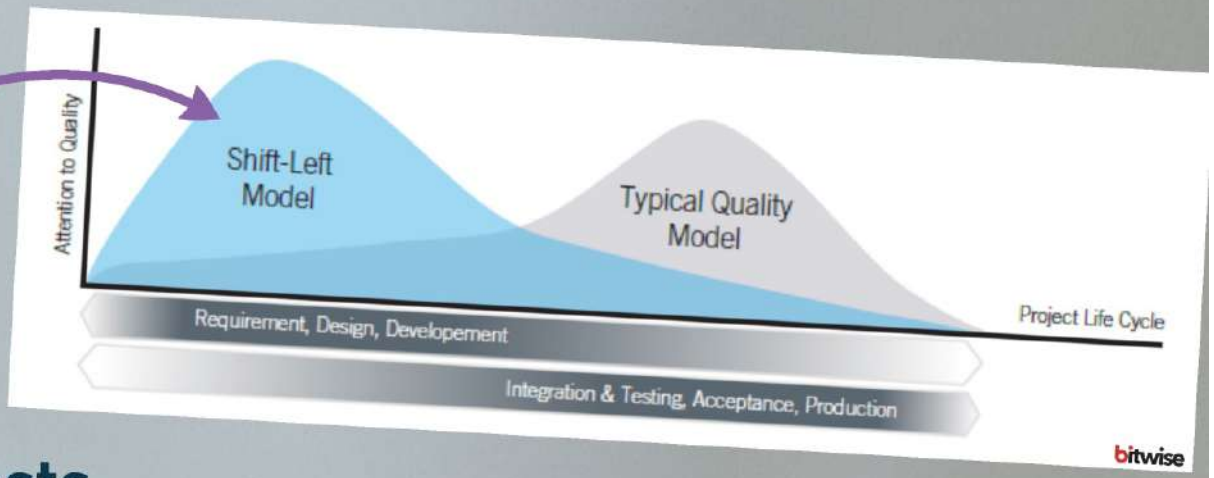


Maven, Gradle
Jenkinsfile
Dockerfile
Kubernetes Manifests
Helm Charts

Terraform
Ansible, Chef, Puppet

Infrastructure as Code

Drive for Shift-left



Maven, Gradle
Jenkinsfile
Dockerfile
Kubernetes Manifests
Helm Charts

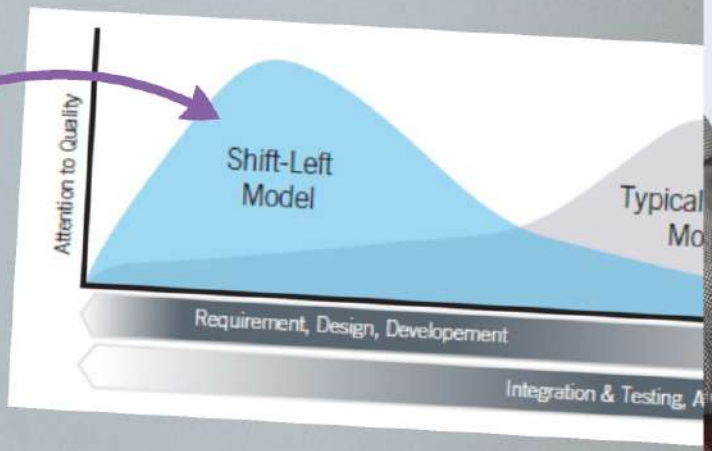
Terraform
Ansible, Chef, Puppet

1) Declarative rather than imperative

2) What data center managers control slides toward developers hands

Infrastructure as Code

Drive for Shift-left



Maven, Gradle
Jenkinsfile
Dockerfile
Kubernetes Manifests
Helm Charts

Terraform
Ansible, Chef, Puppet

1) Declarative rather than imperative

2) What data center managers control slides toward developers hands

Explore Kubernetes

Single-node ***on your laptop***

Install Minikube

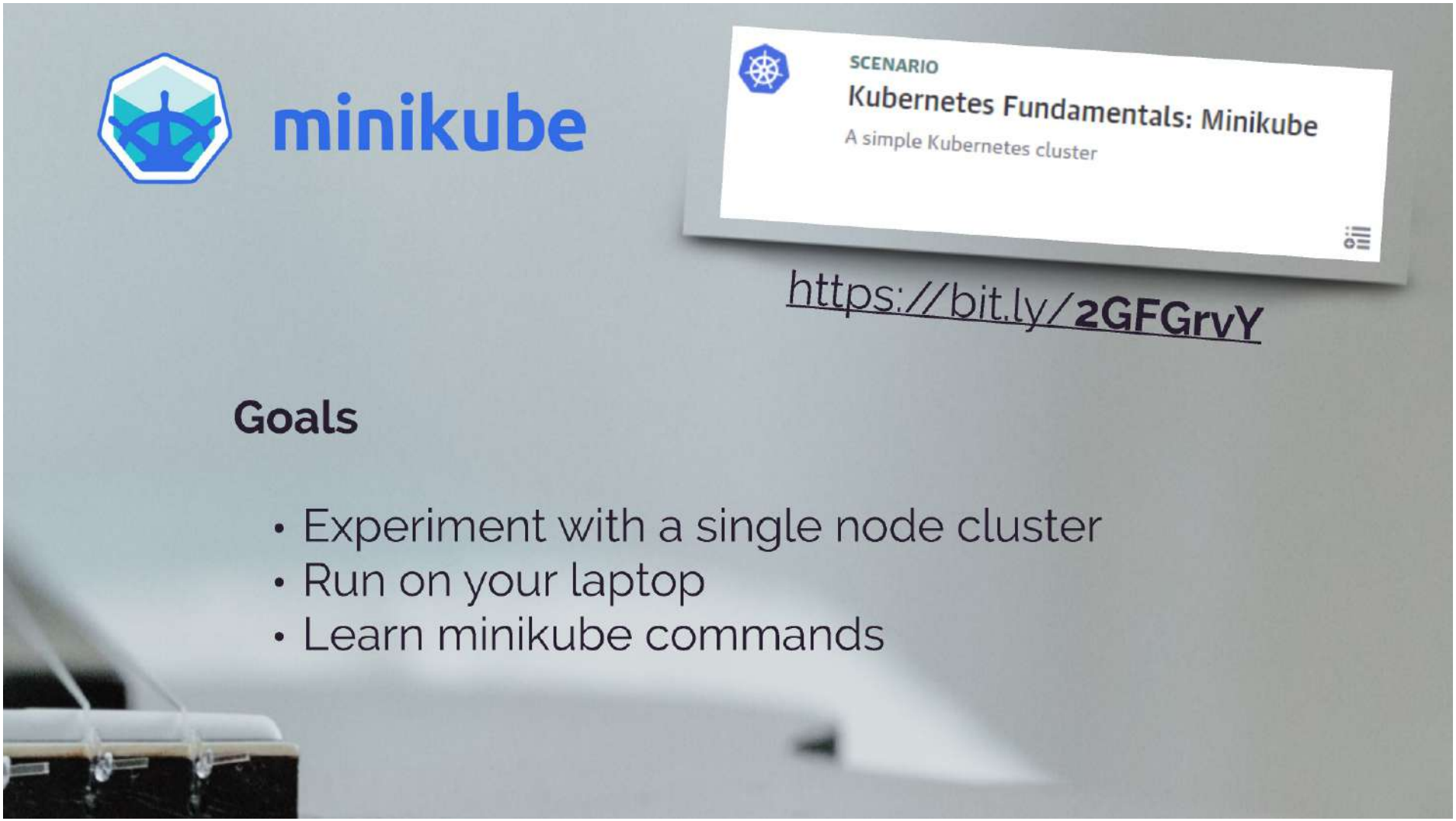
https://kubernetes.io/docs/tasks/tools/install-minikube

This page shows how to install Minikube.

- Before you begin
- Install a Hypervisor
- Install kubectl
- Install Minikube
- What's next

Docker Desktop For Mac or Windows

Other laptop based clusters: KinD, K3s, Microk8s, Minishift


minikube
SCENARIO
Kubernetes Fundamentals: Minikube
A simple Kubernetes cluster
https://bit.ly/2GFGrvY
Goals
• Experiment with a single node cluster
• Run on your laptop
• Learn minikube commands

Reading and Hands-on

https://learning.oreilly.com/scenarios/

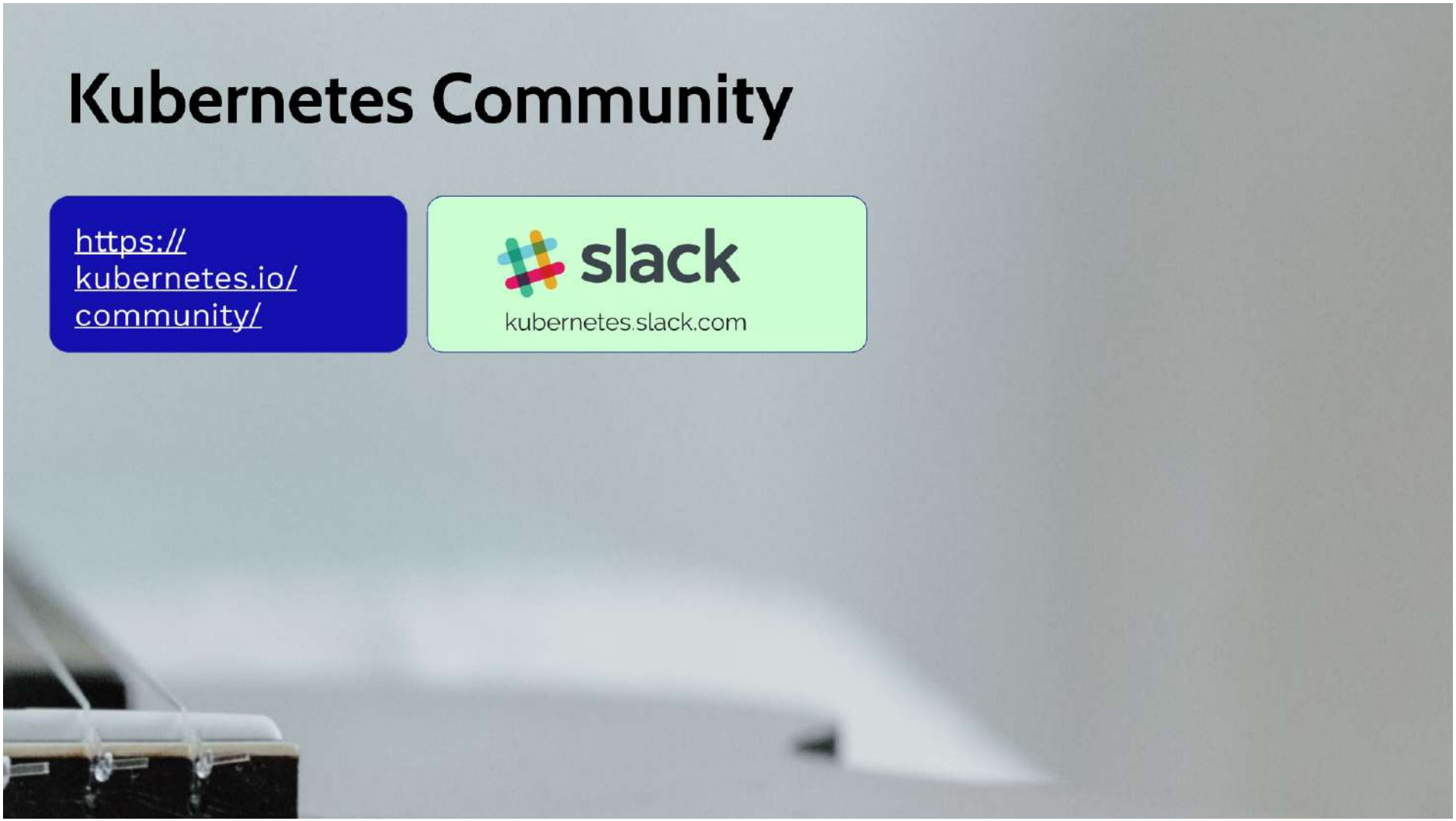
https://kubernetes.io/docs/tutorials/kubernetes-basics/

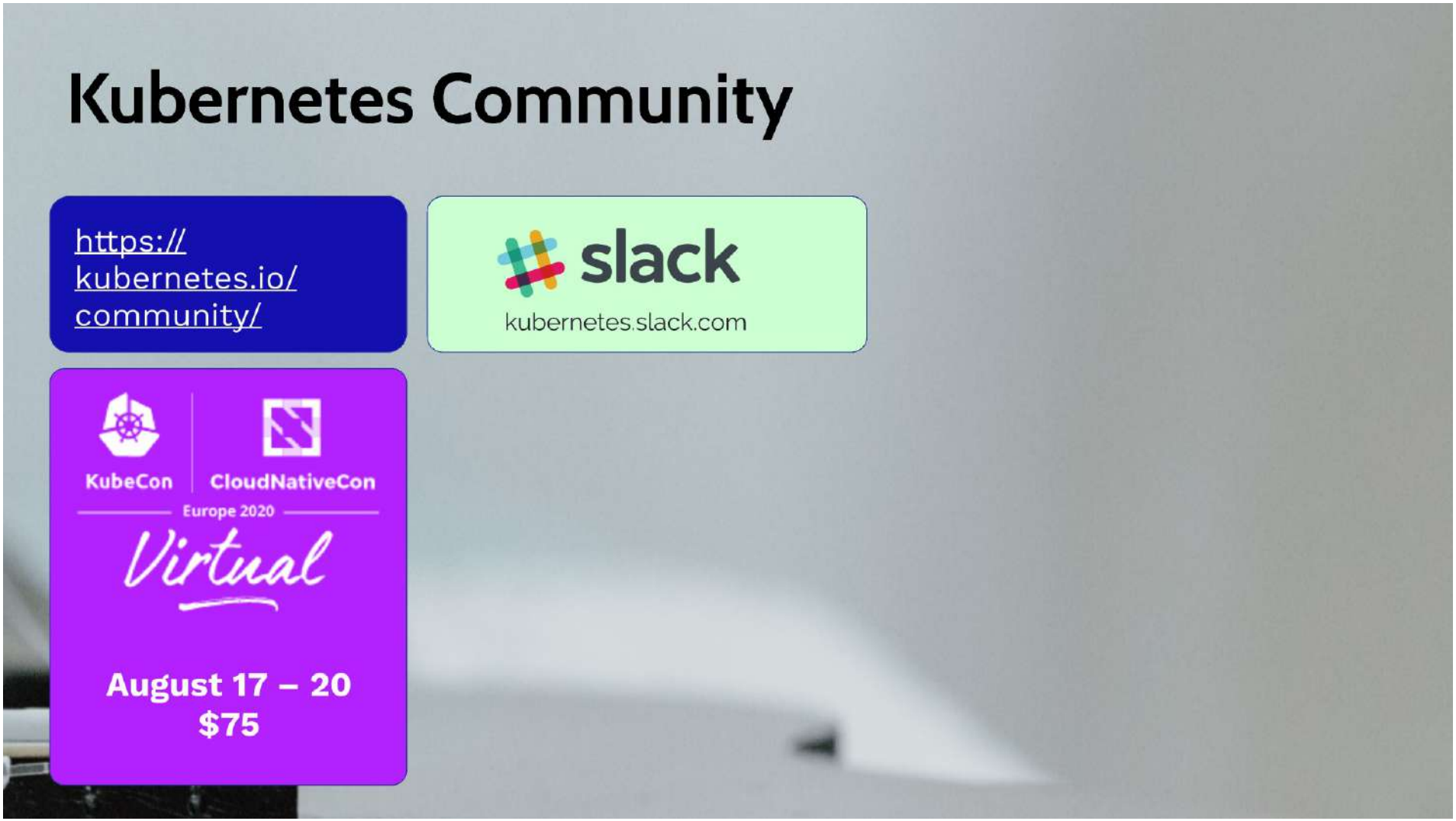
https://katacoda.com/courses/docker

https://katacoda.com/courses/kubernetes

https://learn.openshift.com

https://kubernetes.io/docs/tutorials/online-training/overview/


Kubernetes Community
https://
kubernetes.io/
community/
slack
kubernetes.slack.com


Kubernetes Community
https://
kubernetes.io/
community/
slack
kubernetes.slack.com
KubeCon
CloudNativeCon
Europe 2020
Virtual
August 17 – 20
$75

Kubernetes Community

https://
kubernetes.io/
community/

slack
kubernetes.slack.com

KubeCon | CloudNativeCon
Europe 2020
Virtual

August 17 – 20
$75

WEEKLY

- 218+ weekly newsletters
  - Bob Killen
  - Chris Short
  - Craig Box
  - Kim McMahon
  - Michael Hausenblas

https://kubeweekly.io

Kubernetes Community

https://
kubernetes.io/
community/

slack
kubernetes.slack.com

KubeCon | CloudNativeCon
Europe 2020
Virtual

August 17 – 20
$75

WEEKLY

- 218+ weekly newsletters
  - Bob Killen
  - Chris Short
  - Craig Box
  - Kim McMahon
  - Michael Hausenblas

https://kubeweekly.io


Craig
Box


Adam
Glick

Kubernetes Podcast
from Google

- 105+ weekly podcasts
- Experts interviewed

@KubernetesPod
https://kubernetespodcast.com

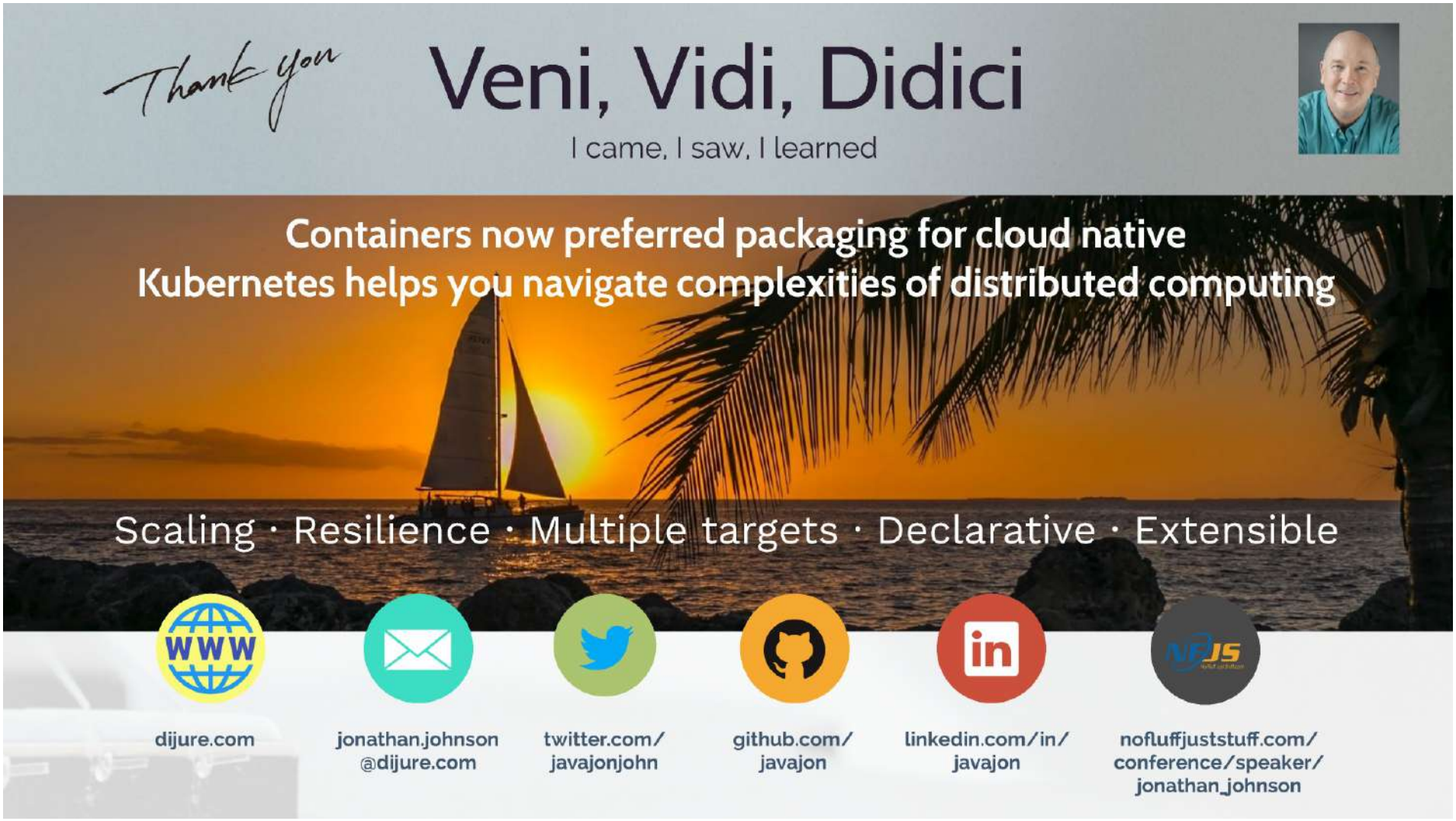
Many Interests

Talks and workshops

- Container patterns
- Java in containers
- Meshing and Istio
- Serverless
- Observability
- Pipelining and CI/CD
- Operator pattern
- Knative
- Extending Kubernetes

- Distributed computing
- Machine learning
- Cluster hardening
- Cloud KaaS offerings
- Scaling techniques
- Ingress and load balancing
- Delivery strategies
- GitOps, SecOps, SREs
- Federated and hybrid clusters
- Security, RBAC, Secrets and Vault
- Persistent volumes
- Certifications: CKAD, CKA

- Architecting: RabbitMQ, NGINX, Redis, KubeFlow, Tensorflow, Kafka, PostgreSQL, GraalVM, Spring...


Thank you
Veni, Vidi, Didici
I came, I saw, I learned
Containers now preferred packaging for cloud native
Kubernetes helps you navigate complexities of distributed computing
Scaling · Resilience · Multiple targets · Declarative · Extensible
WWW
in
NFJS
dijure.com
jonathan.johnson
@dijure.com
twitter.com/
javajonjohn
github.com/
javajon
linkedin.com/in/
javajon
nofluffjuststuff.com/
conference/speaker/
jonathan_johnson


Kubernetes
Getting Started from a Developer Perspective
Architecture
First Apps
Objects
Learning Resources
Distributed Computing
Community