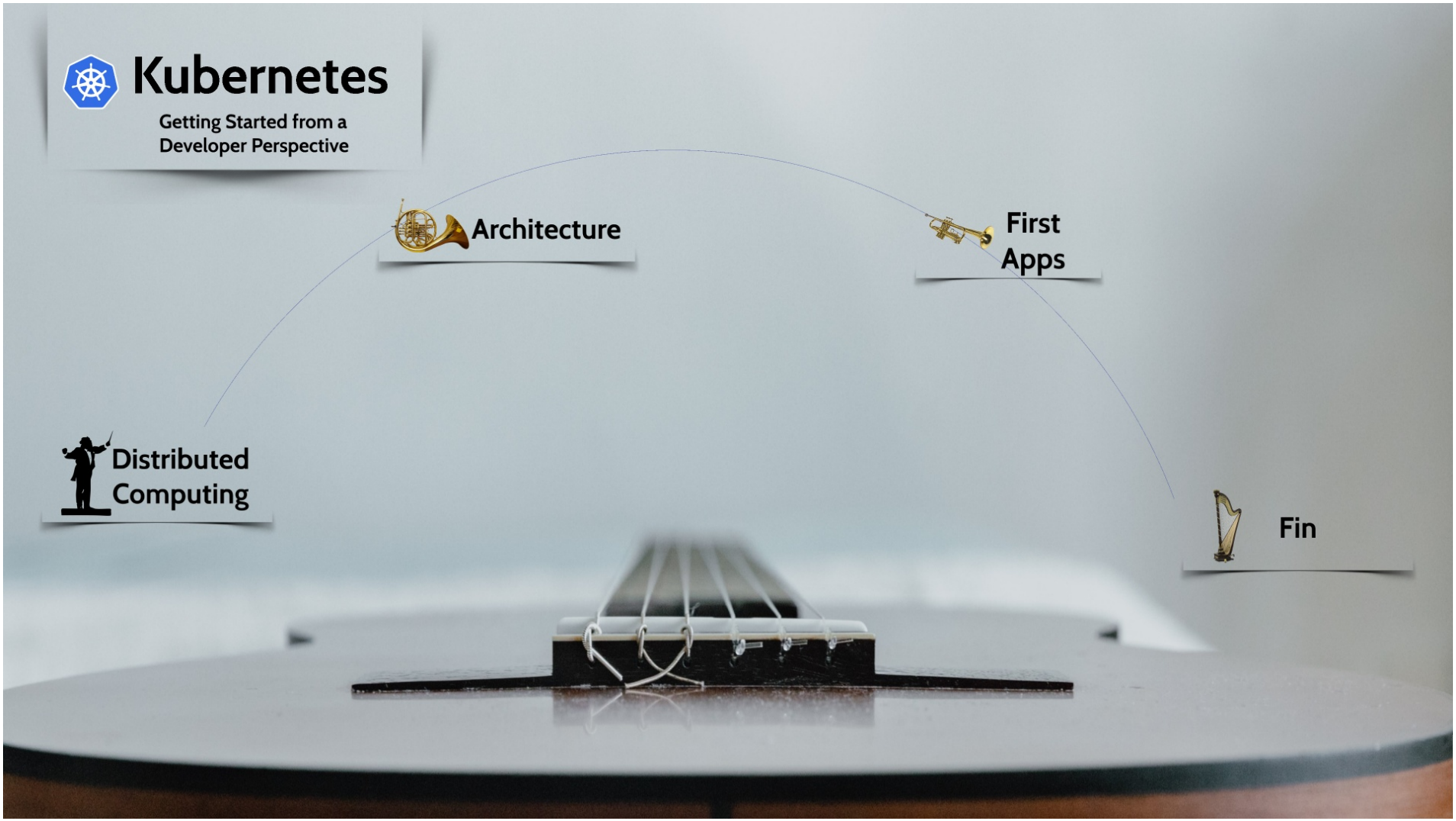
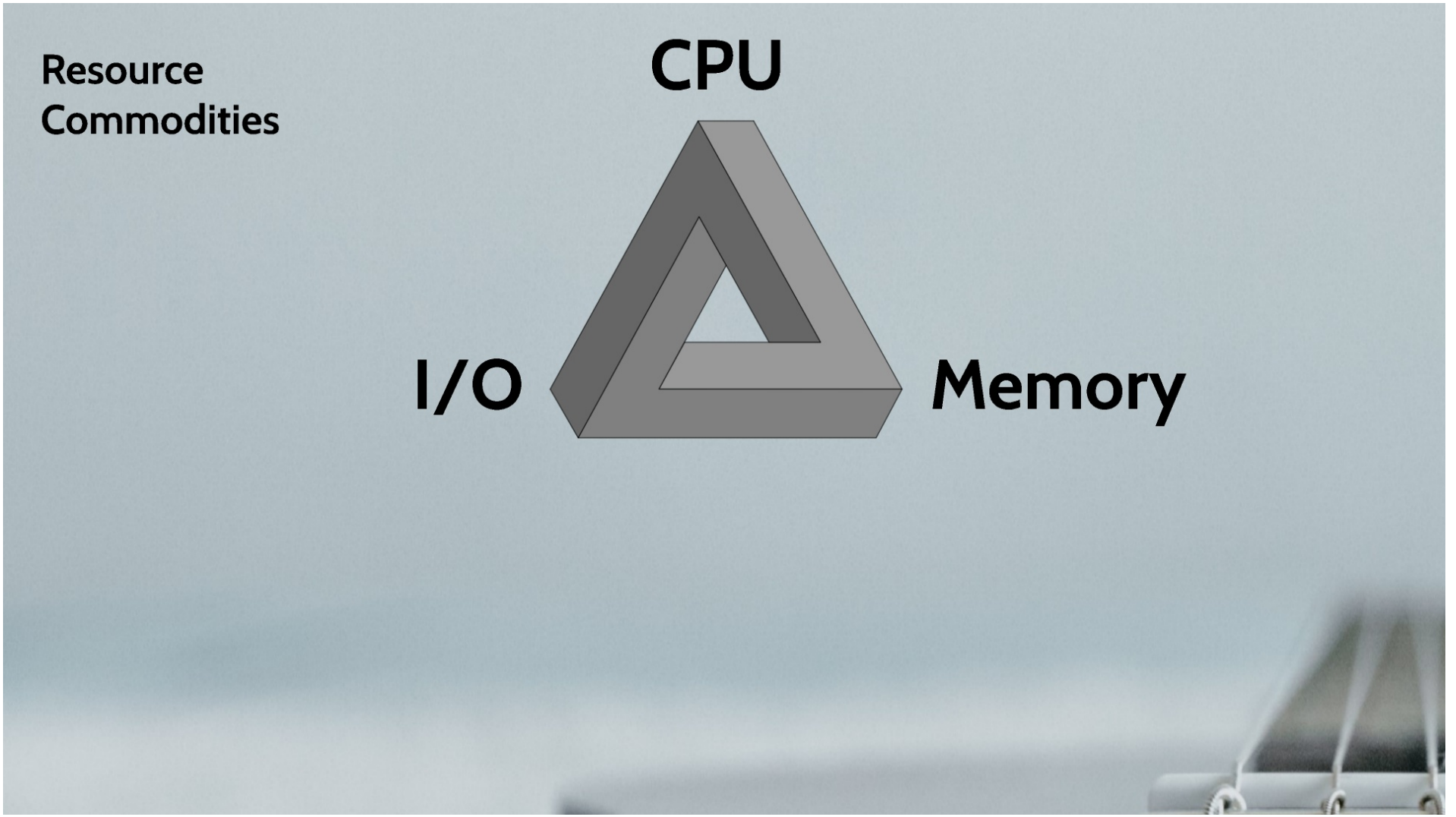
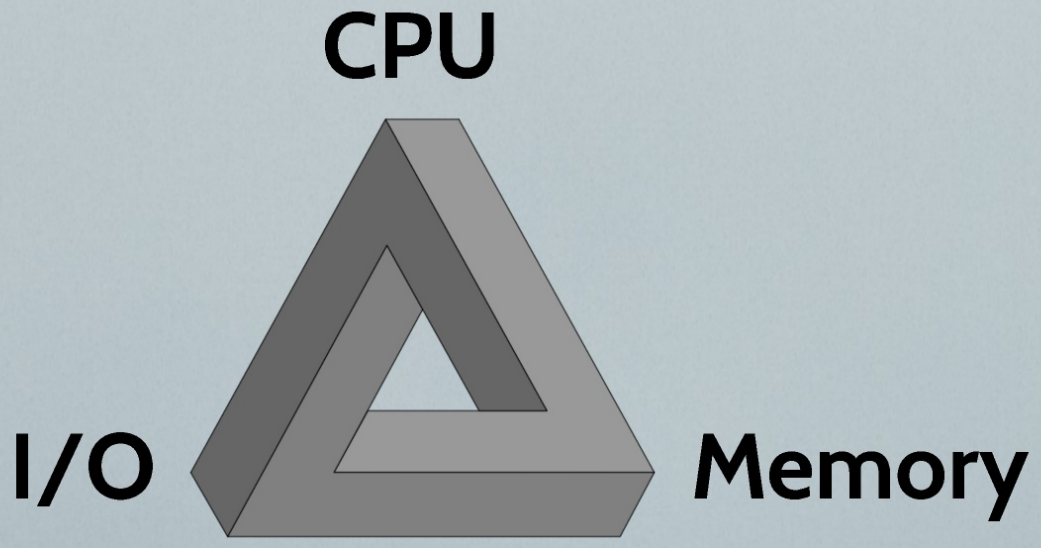

Kubernetes
Getting Started from a
Developer Perspective
Architecture
First
Apps
Distributed
Computing
Fin


Resource
Commodities
CPU
I/O
Memory

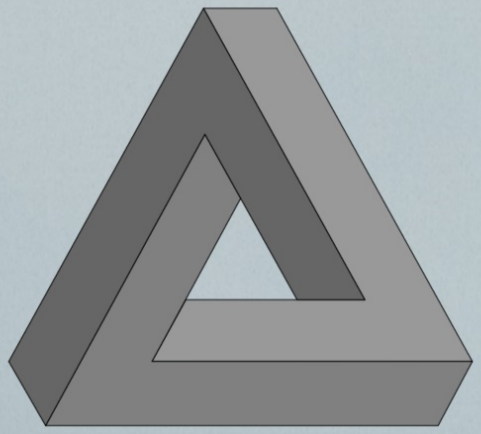
Resource
Commodities



Operating systems manage these resource

Resource Commodities

CPU



I/O

Memory

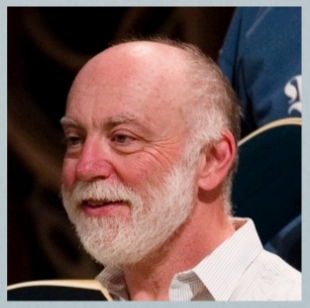
Operating systems manage these resource

Virtual machines abstract access to these resources

Containers are lighter technique for restricted resource access

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.

space-based architecture


layered monolith


microkernel

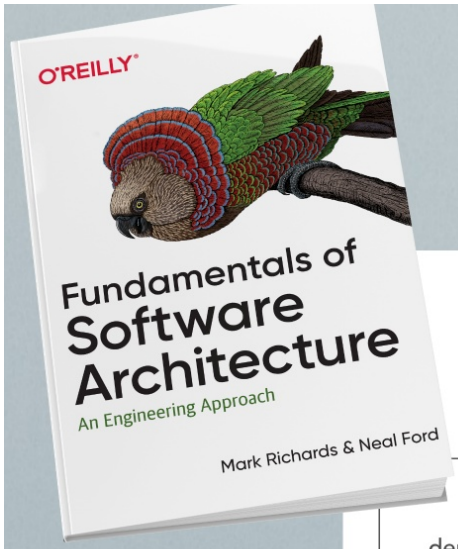

microservices


service-based

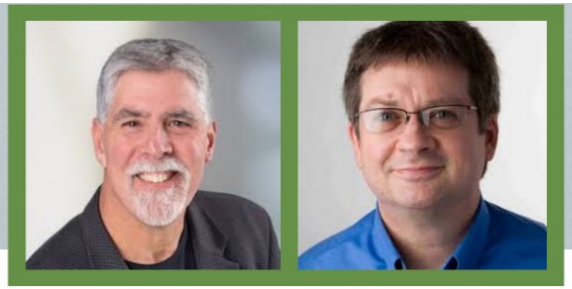

event-driven


space-based

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



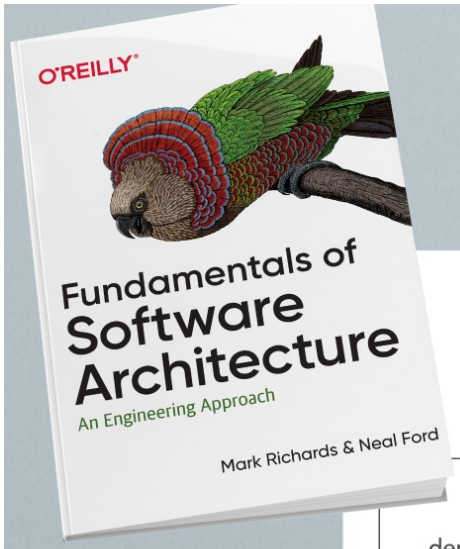
Mark Richards

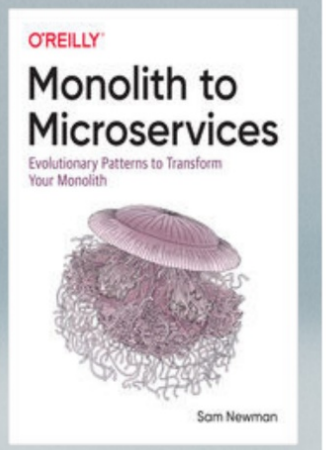


Neal Ford

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	★★★★★	★★★★★	★	★★★★	★★★	★★



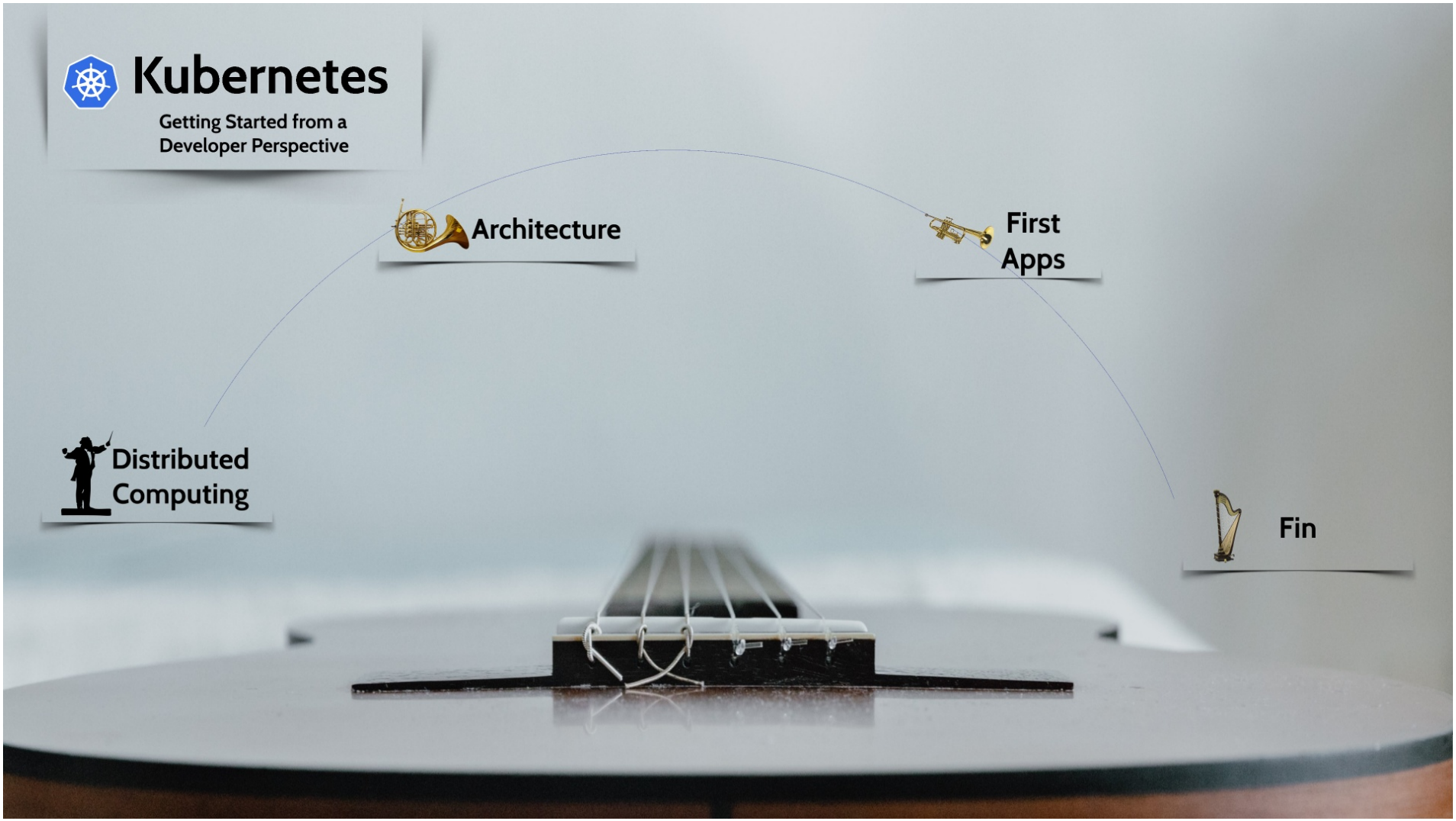


Mark Richards

Neal Ford

space-based architecture

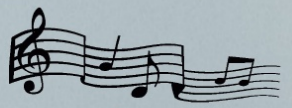
	layered monolith	microkernel	microservices	service-based	event-driven	space-based
agility	★	★★★	★★★★★	★★★★	★★★★	★★★
deployment	★	★★★	★★★★★	★★★★	★★★	★★★
testability	★★	★★★	★★★★★	★★★★	★★	★
performance	★★★	★★★	★★	★★★	★★★★★	★★★★★
scalability	★	★	★★★★★	★★★	★★★★	★★★★★
elasticity	★	★	★★★★	★★	★★★	★★★★★
simplicity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


Kubernetes
Getting Started from a
Developer Perspective
Architecture
First
Apps
Distributed
Computing
Fin

Kubernetes Architecture

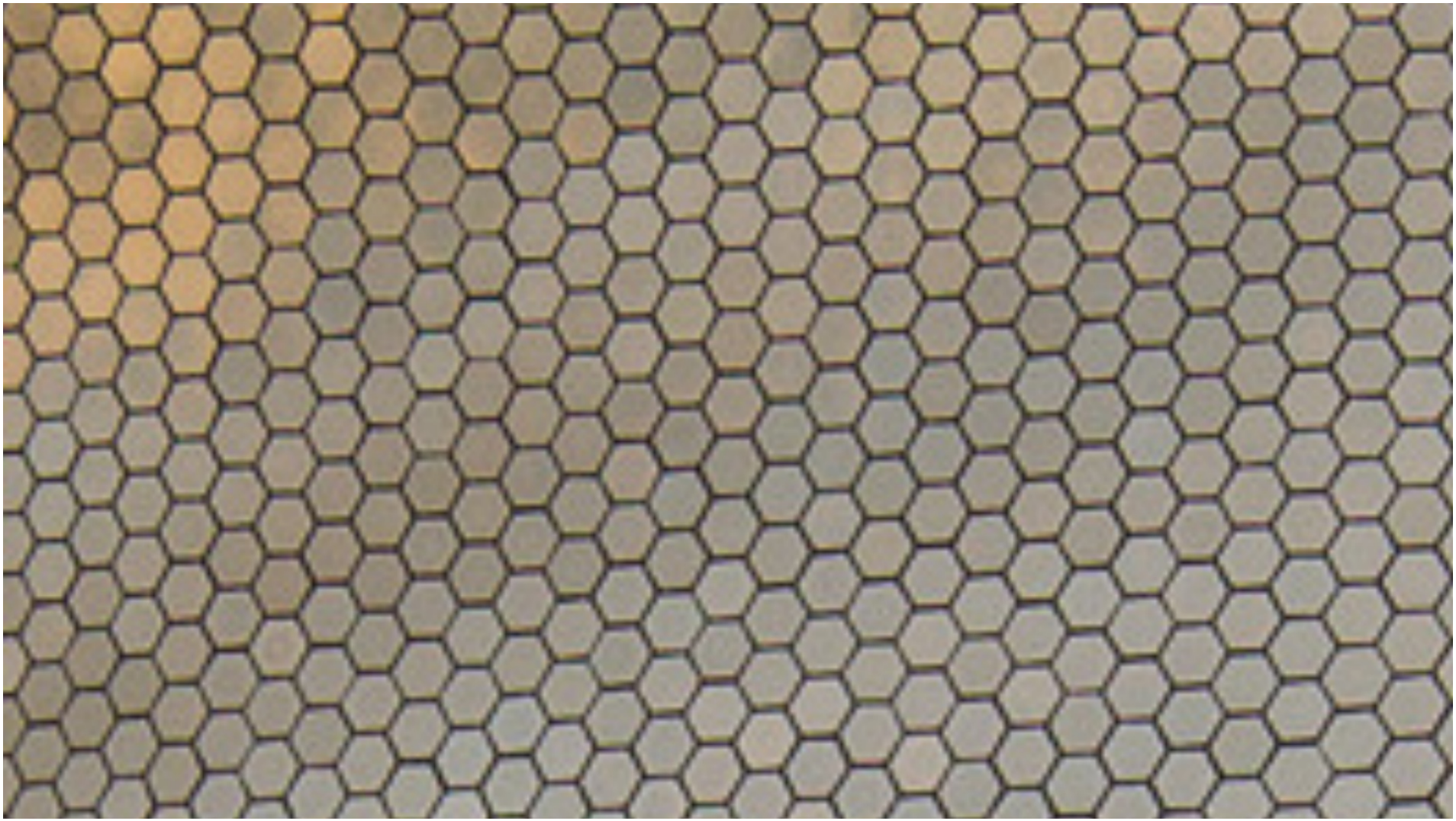
Key Features

Components



Declarative


Soumaya museum of art
Mexico City
Select

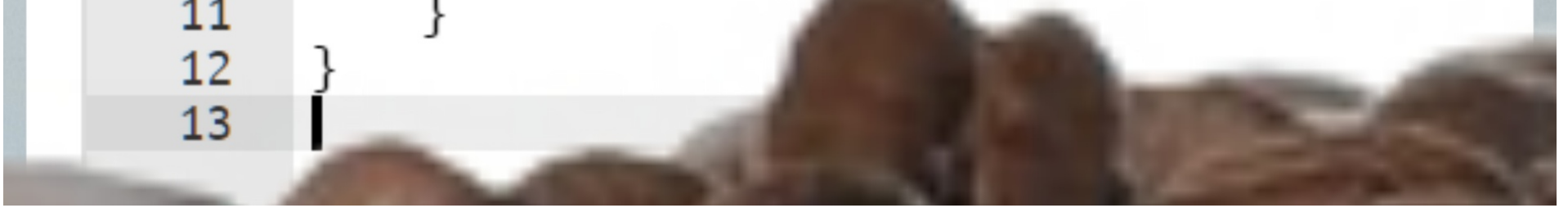


Your Code ...

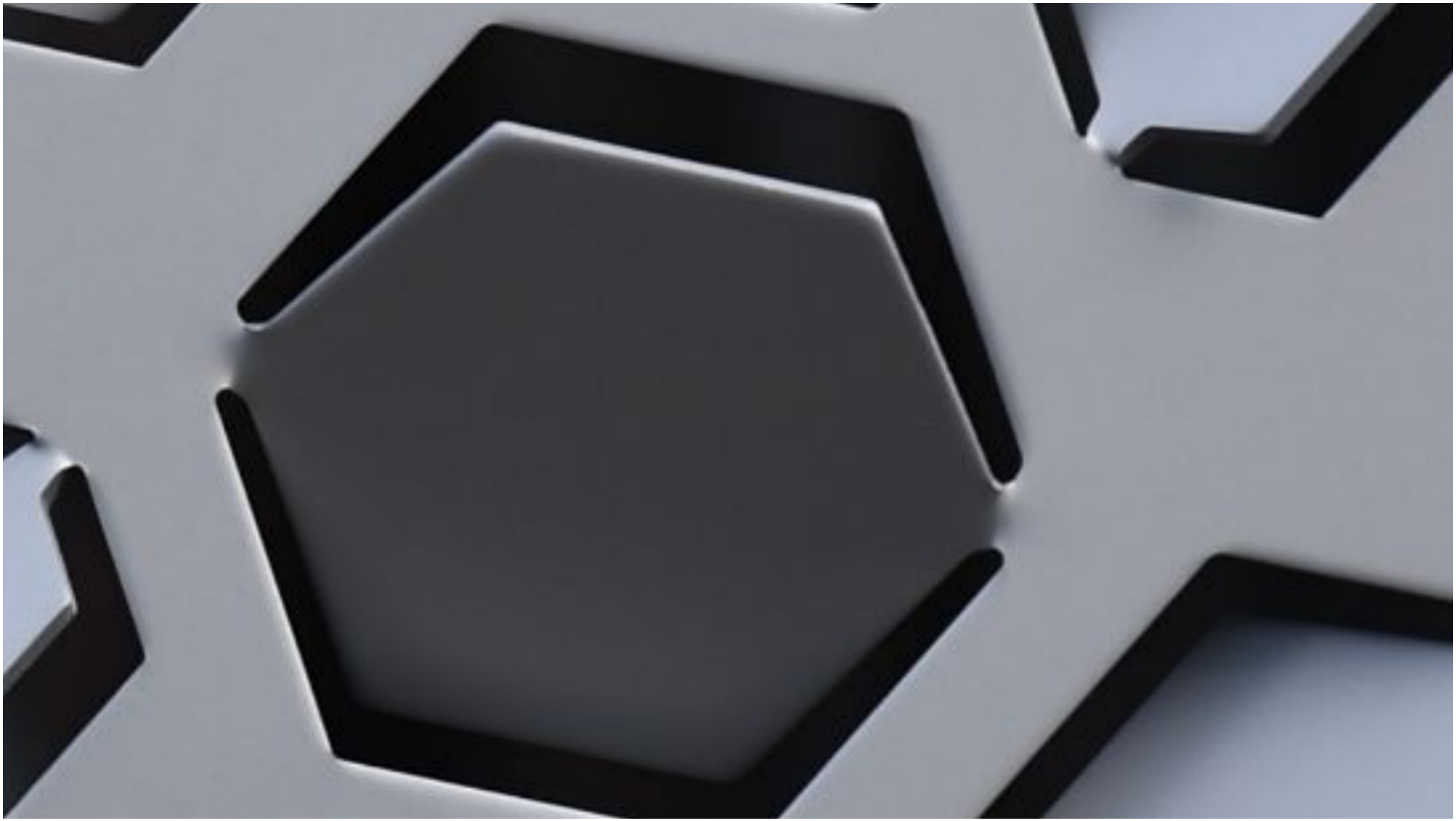
```
1  public class DeepThought {
2      public int worker() {
3
4          int x = getMeaningOfLife();
5
6          int y = getMeaningOfUniverse();
7
8          int z = getMeaningOfEverything();
9
10         return x + y + z;
11     }
12 }
13
```

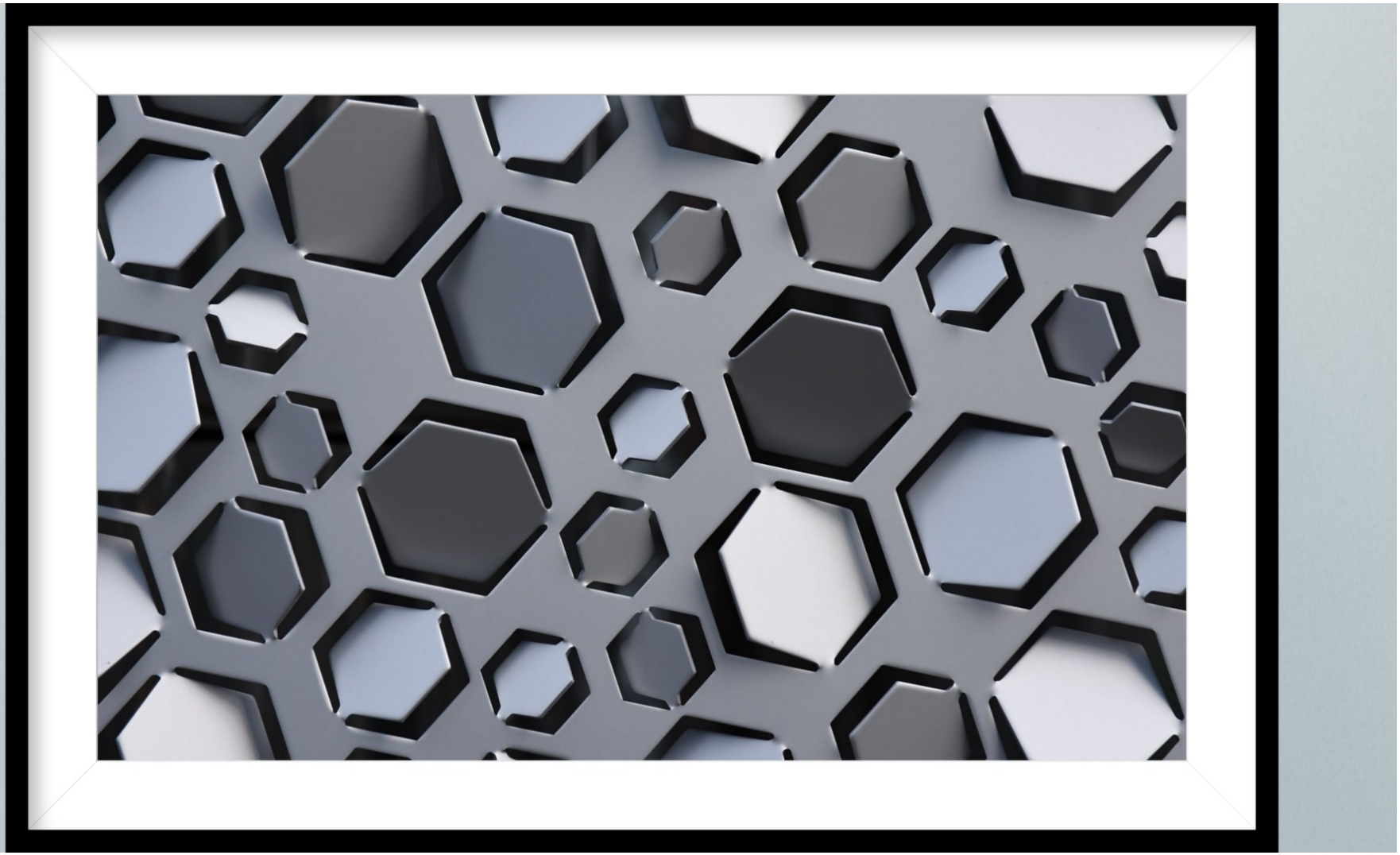
Your Code ...

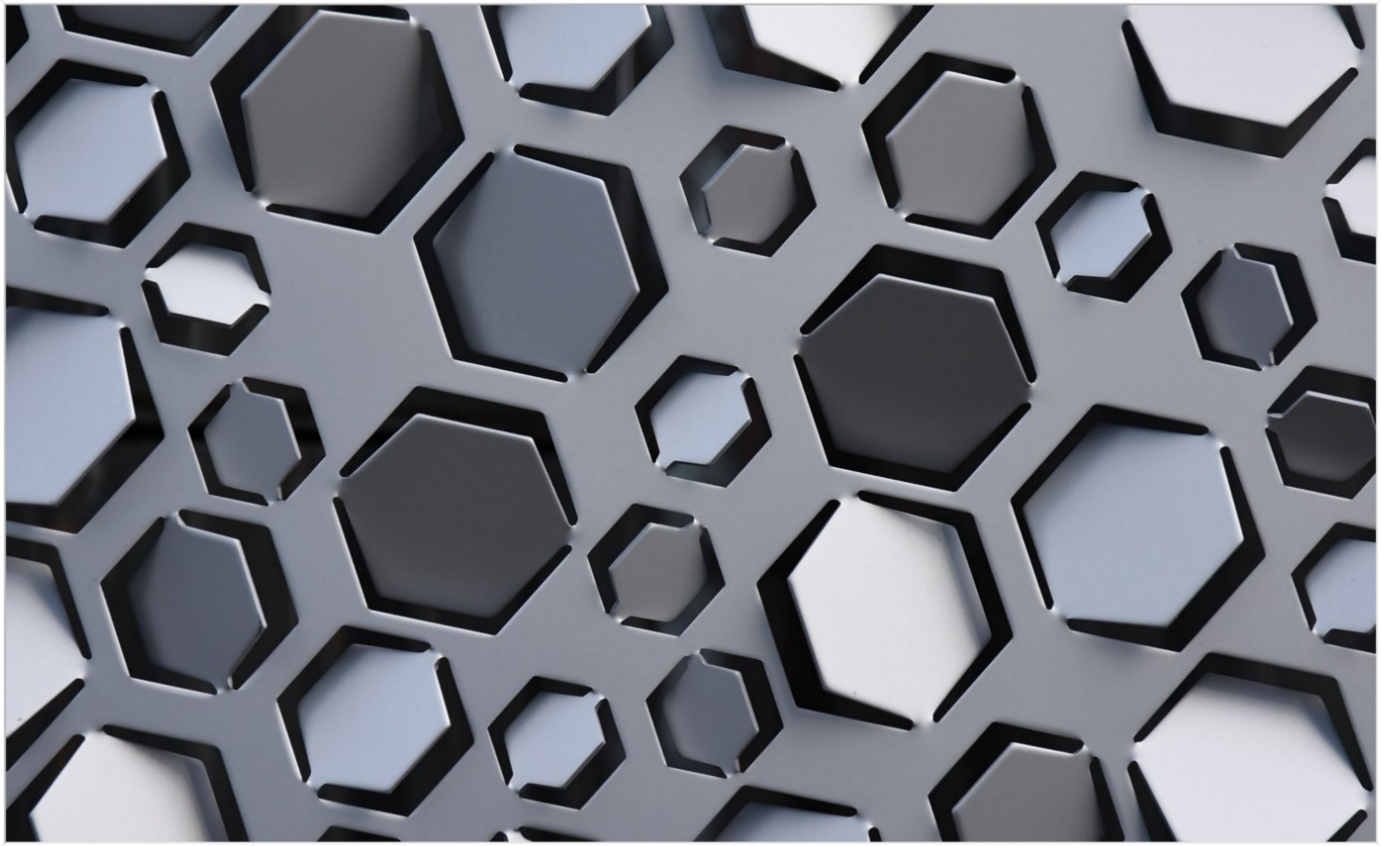
```
1  public class DeepThought {
2      public int worker() {
3
4          int x = getMeaningOfLife();
5
6          int y = getMeaningOfUniverse();
7
8          int z = getMeaningOfEverything();
9
10         return x + y + z;
11     }
12 }
13
```

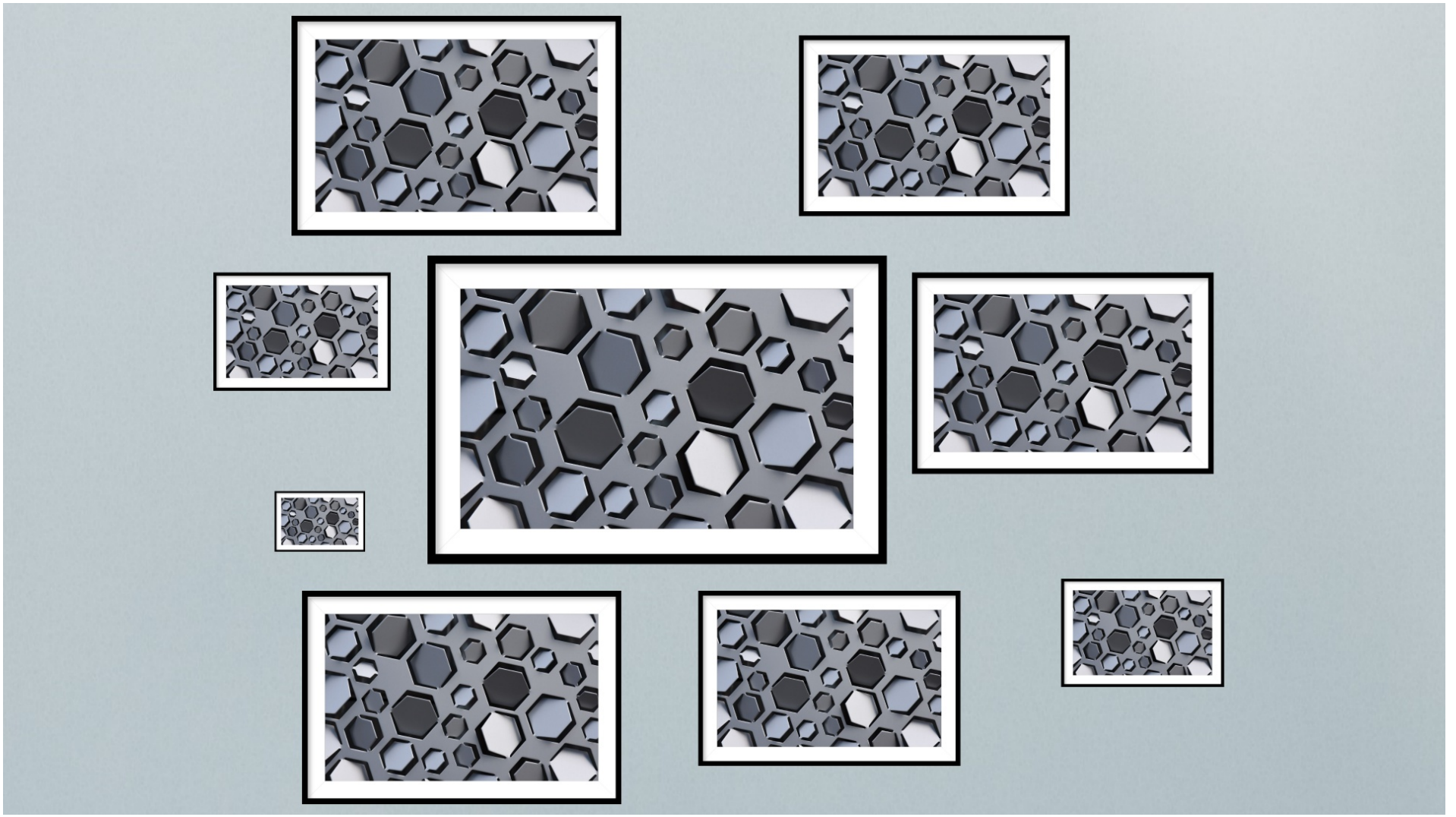


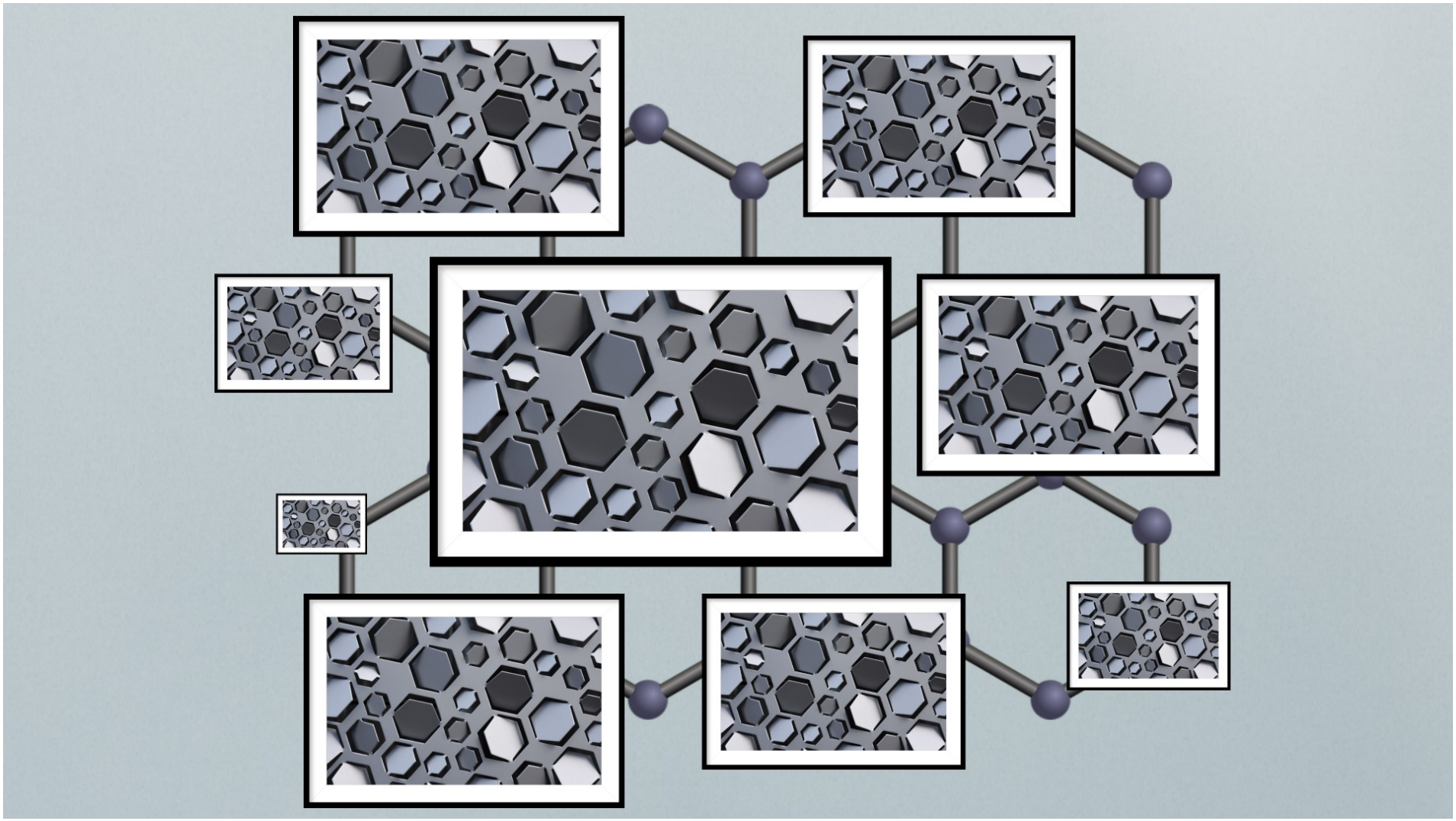

Your Code ...
public class DeepThought {
public int worker() {
int x = getMeaningOfLife();
int y = getMeaningOfUniverse();
int z = getMeaningOfEverything();
return x + y + z;
}
}





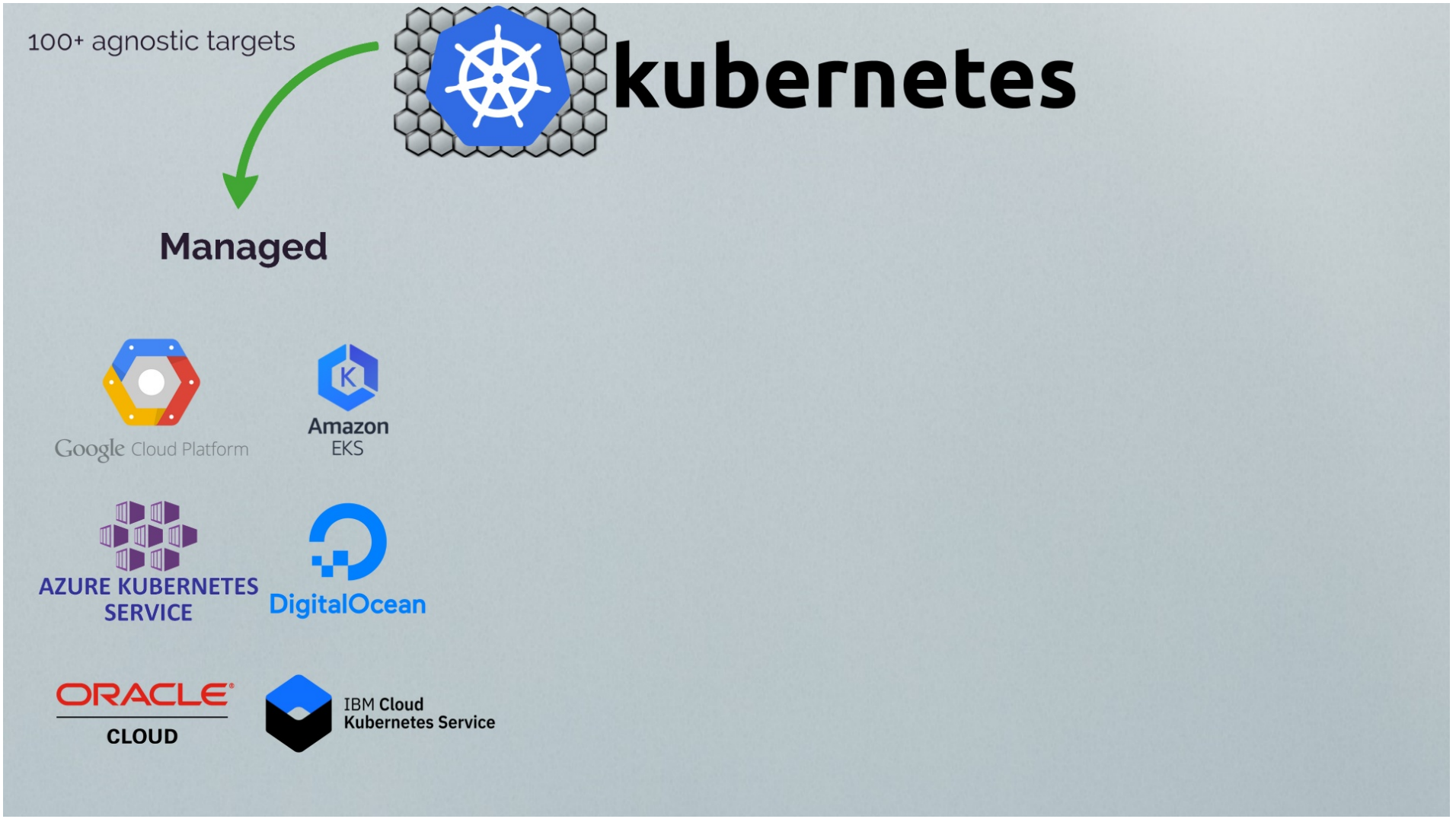






100+ agnostic targets

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

Hybrid, on-prem

Google Cloud Platform

Amazon EKS

AZURE KUBERNETES SERVICE

DigitalOcean

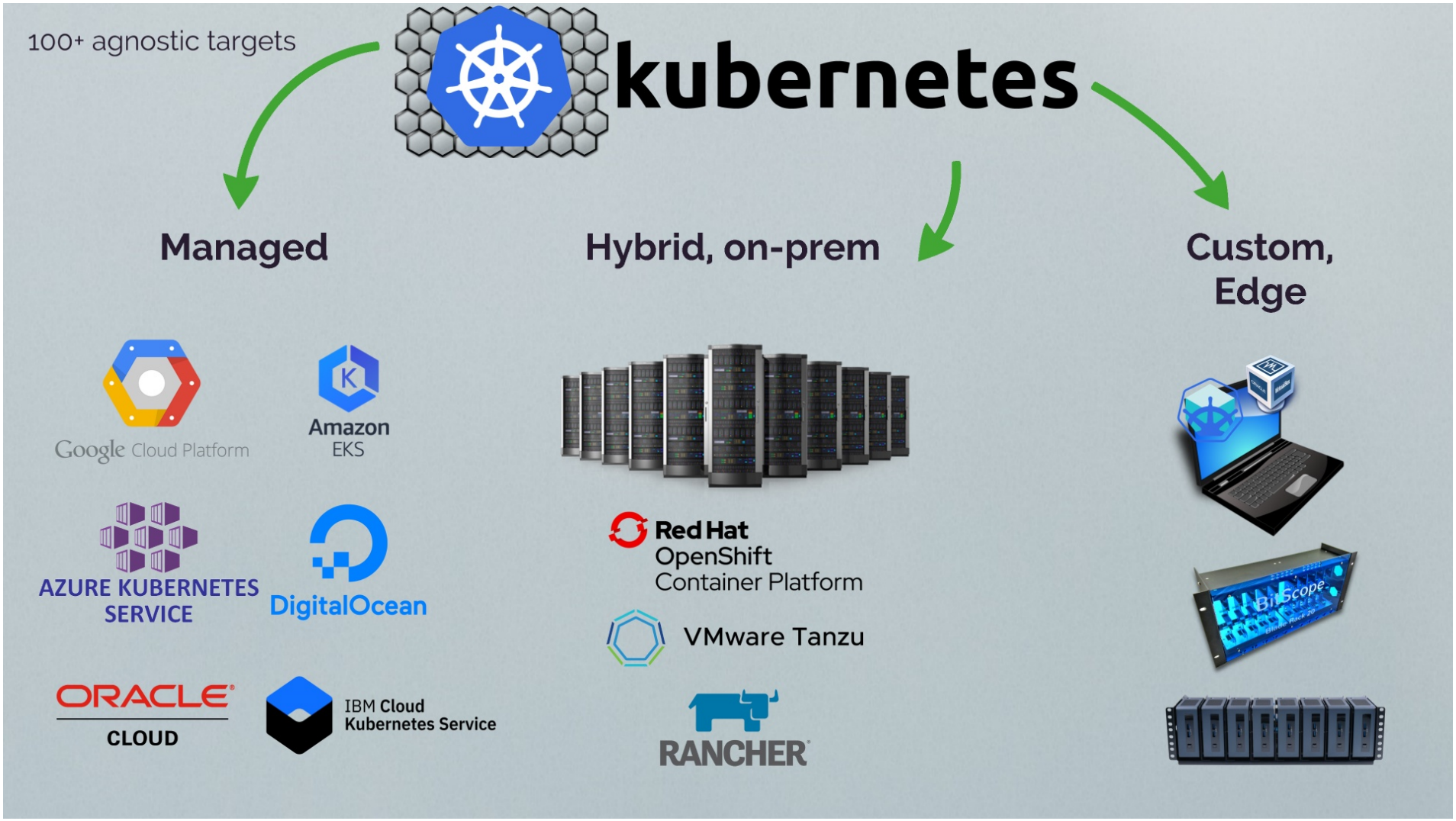
ORACLE CLOUD

IBM Cloud Kubernetes Service

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

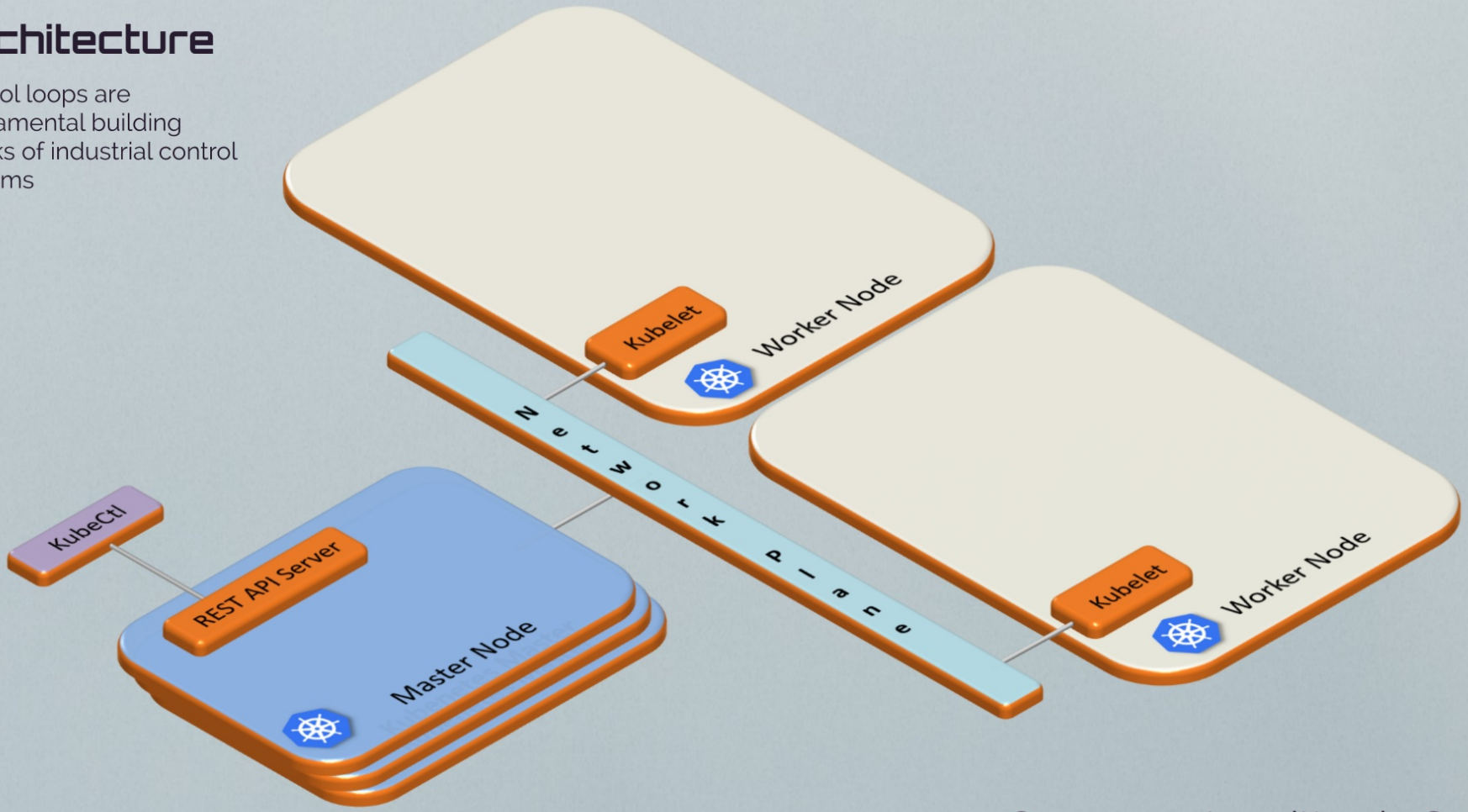
Worker nodes run your processes

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



Components written in Go

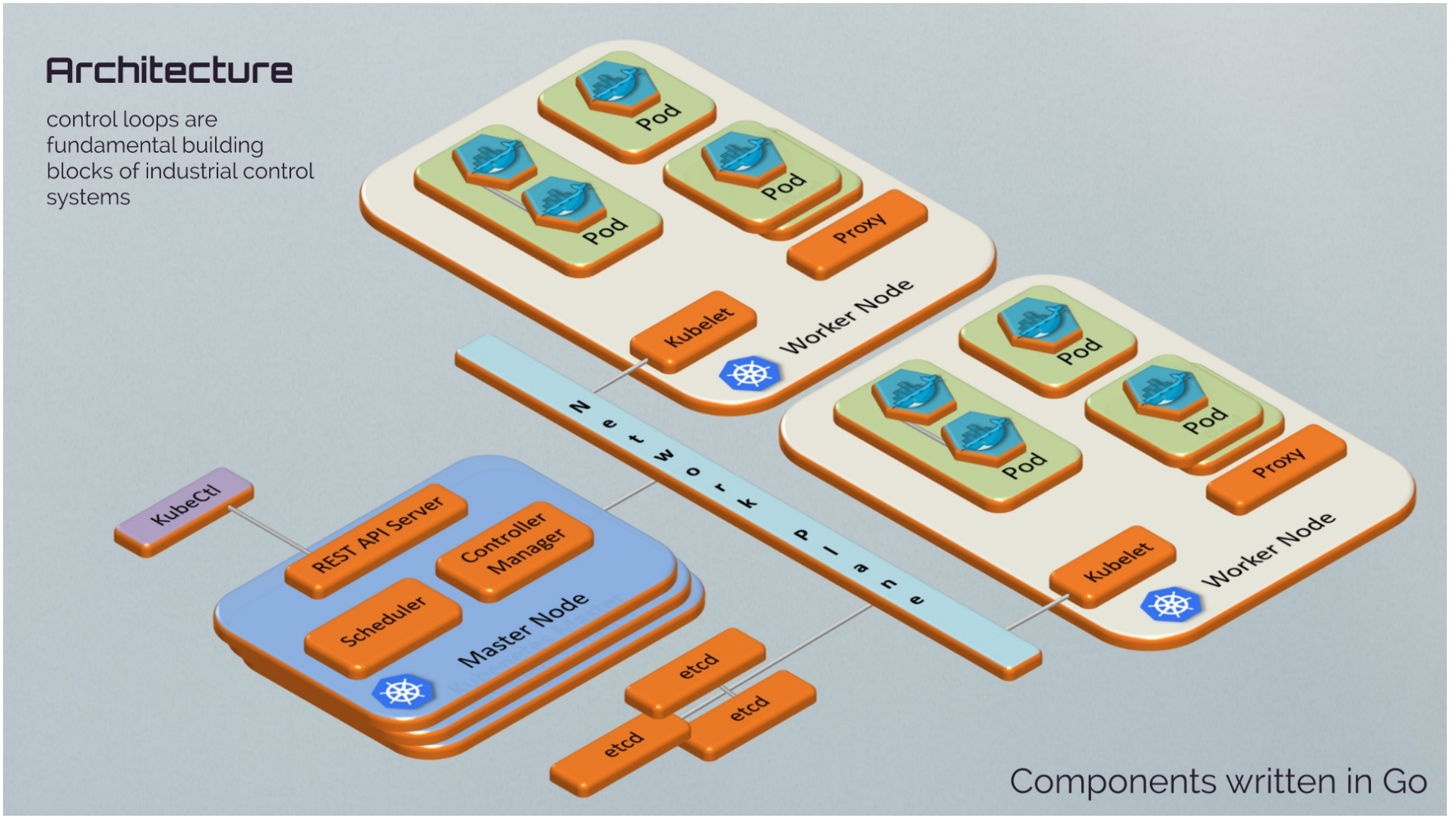
Architecture

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



Components written in Go

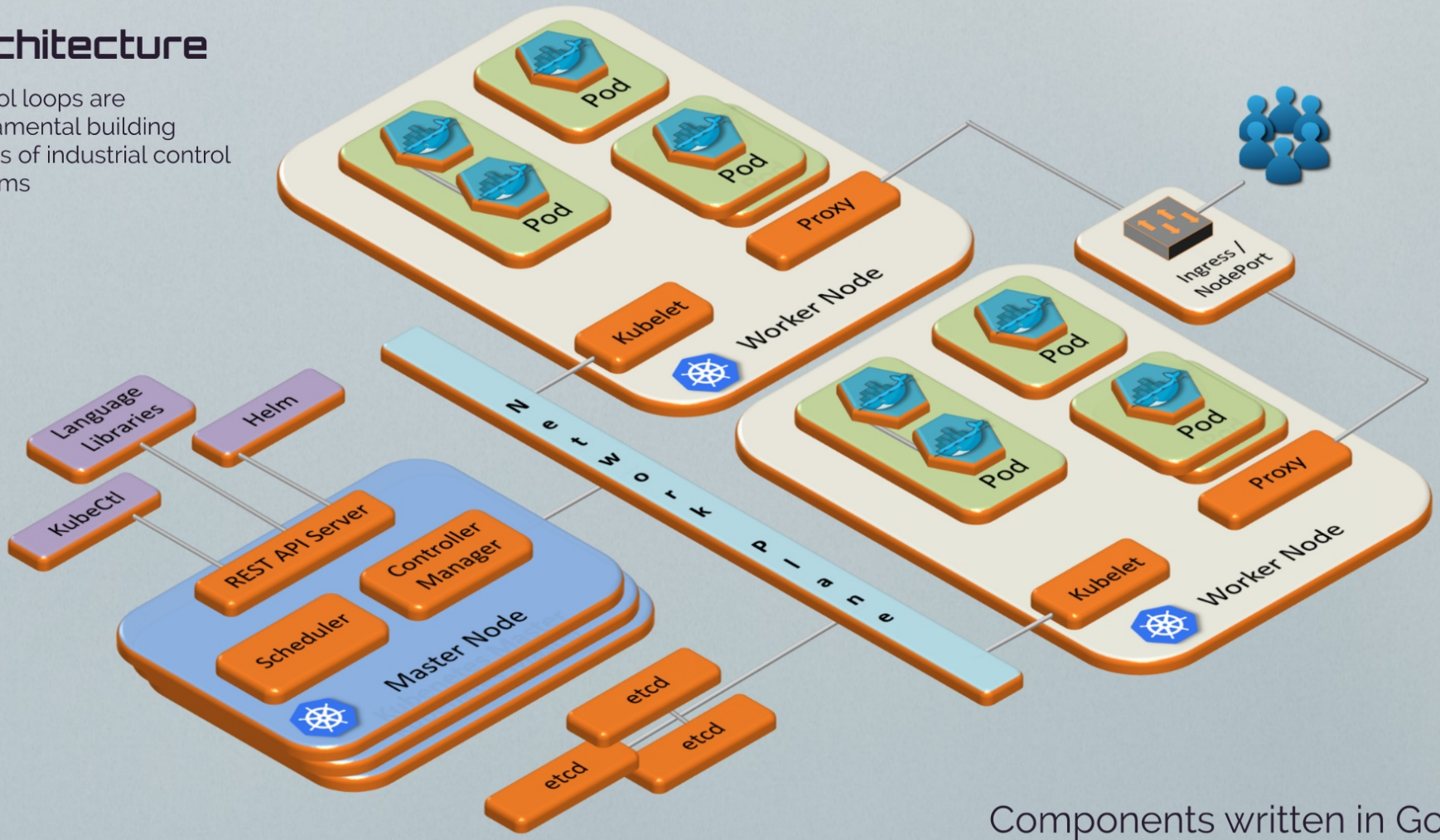
Architecture

control loops are fundamental building blocks of industrial control systems

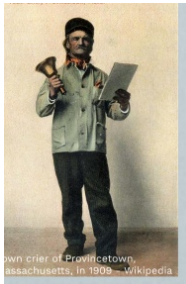
Components written in Go

Architecture

control loops are fundamental building blocks of industrial control systems



Components written in Go

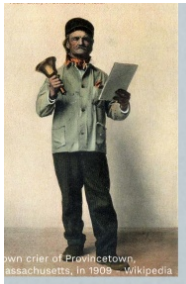


State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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



State machine instructions

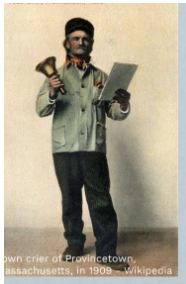
- You declare, not script
- Model resources with YAMLs and charts

Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

Declarative

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



State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

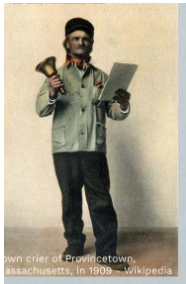
Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

Declarative

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

Object controller version



State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

Imperative

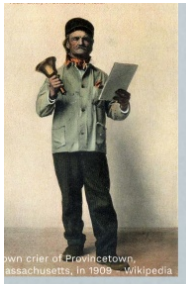
```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

Declarative

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

Object controller version

Object classification



State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

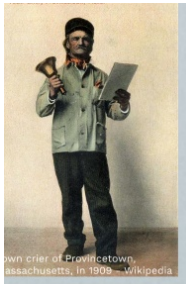
Declarative

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

Object controller version

Object classification

Associated data



State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

Declarative

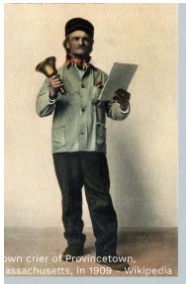
```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    venue: opera
    watch: cpu
spec:
```

Object controller version

Object classification

Associated data

Specific object details



State machine instructions

- You declare, not script
- Model resources with YAMLs and charts

Imperative

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

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

Declarative

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

Object controller version

Object classification

Associated data

Specific object details

YAML — Human friendly data serialization standard

HELM — Package manager for Kubernetes

💡 **VSCode** and **Intellij** extensions for writing YAMLs

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
```


```
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)

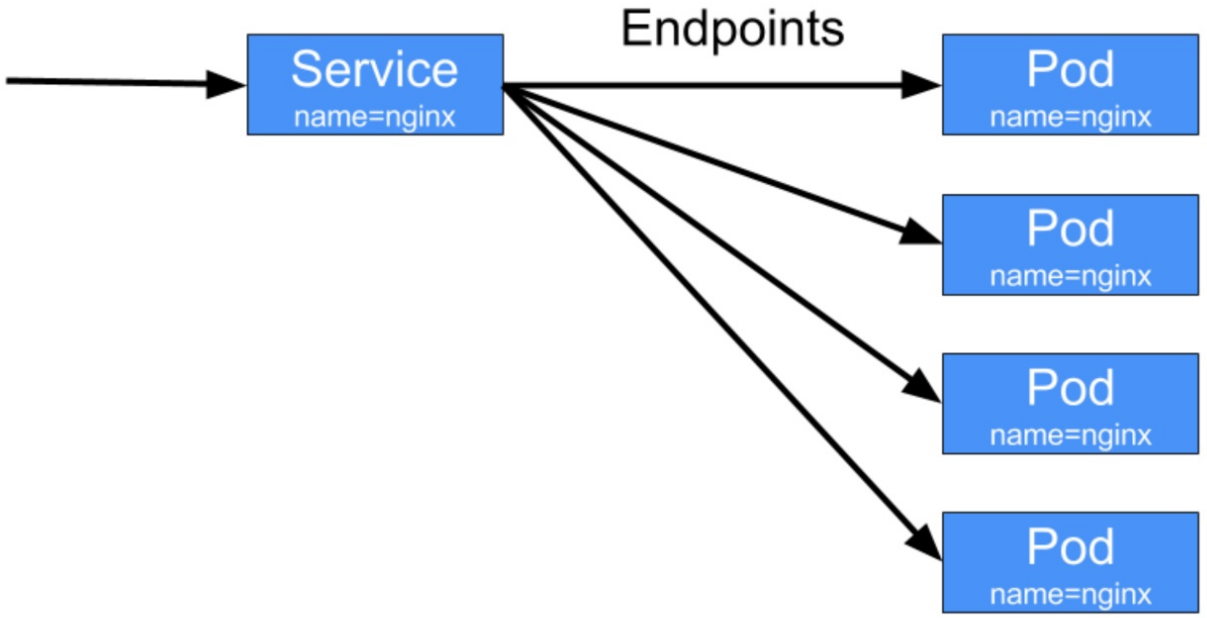
```
$ docker run --name my-nginx -p 80 nginx:1.19.2
```

```
$ 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
```

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
```

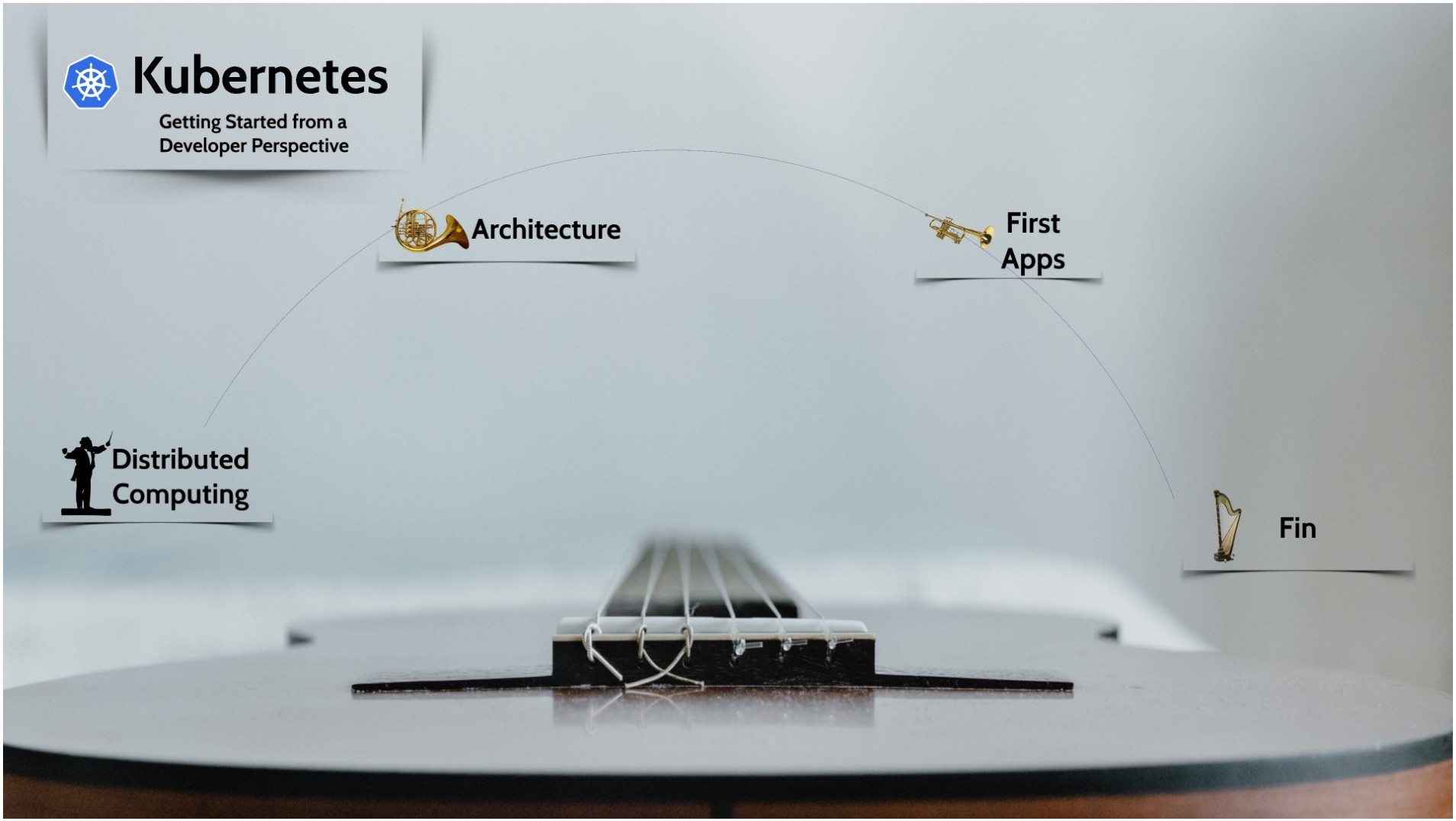
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.1
        ports:
    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

Declaring Service Associated to Pods

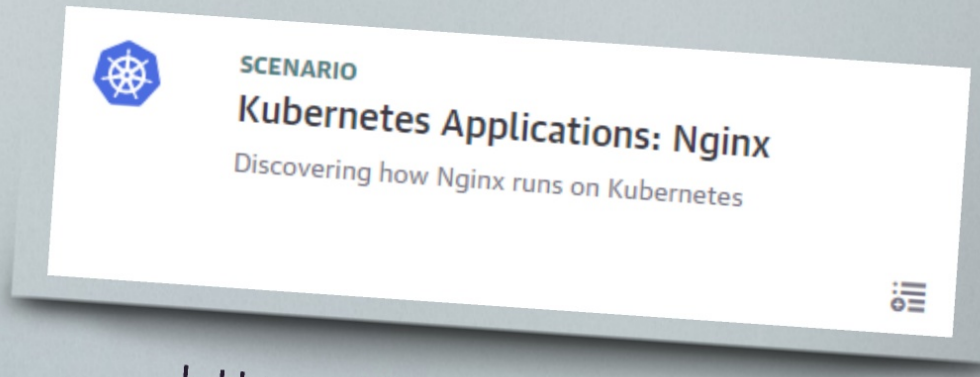
Labels used to declare associations

```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
```


Kubernetes
Getting Started from a
Developer Perspective
Architecture
First
Apps
Distributed
Computing
Fin

A common Hello World
for 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

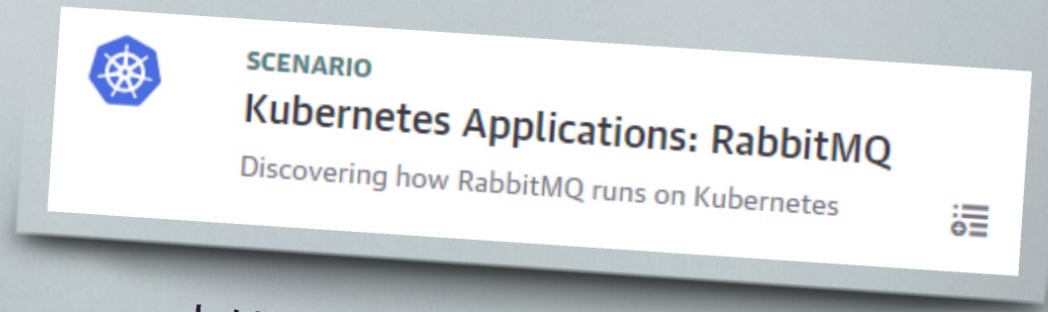
First App

https://bit.ly/33Cd4Uw

Goals

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

AMQP messaging benefiting
from Kubernetes

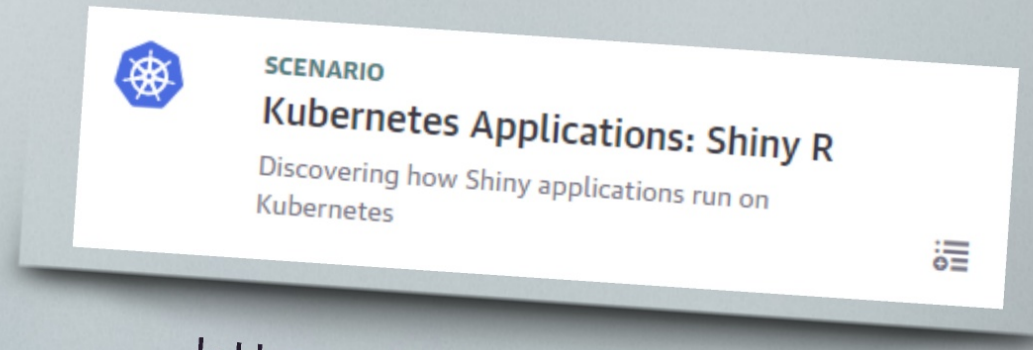


https://bit.ly/**3dcRg4Y**

Goals

- More Kubectl
- Helm charts
- Install RabbitMQ on Kubernetes
- StatefulSet application
- Play with a little chaos

R Language



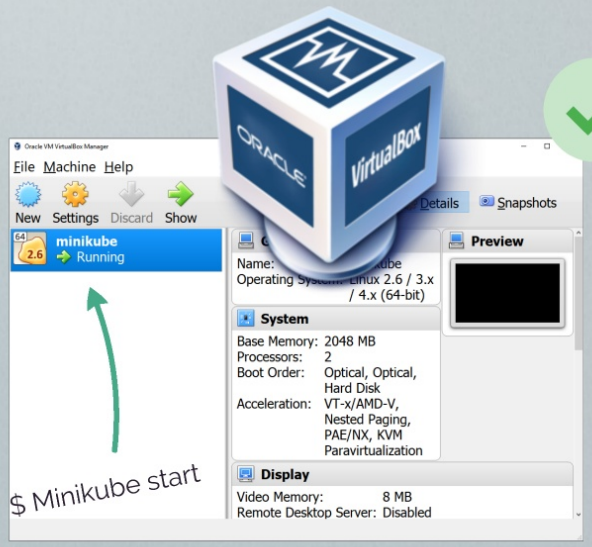
https://bit.ly/3jEK5EP

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

Explore Kubernetes

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



Install Minikube

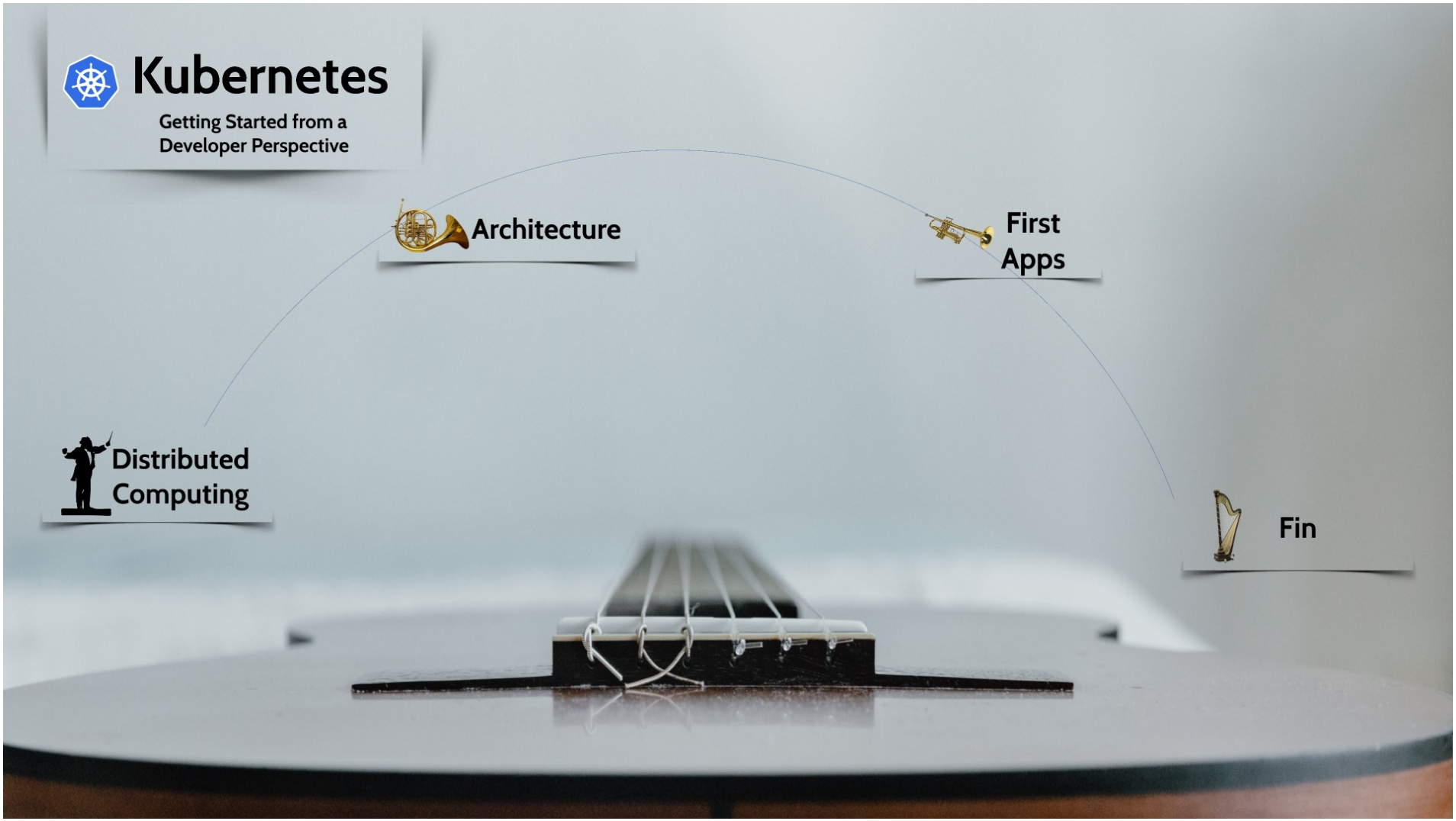
https://kubernetes.io/docs/tasks/tools/install-minikube

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


Kubernetes
Getting Started from a
Developer Perspective
Architecture
First
Apps
Distributed
Computing
Fin

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

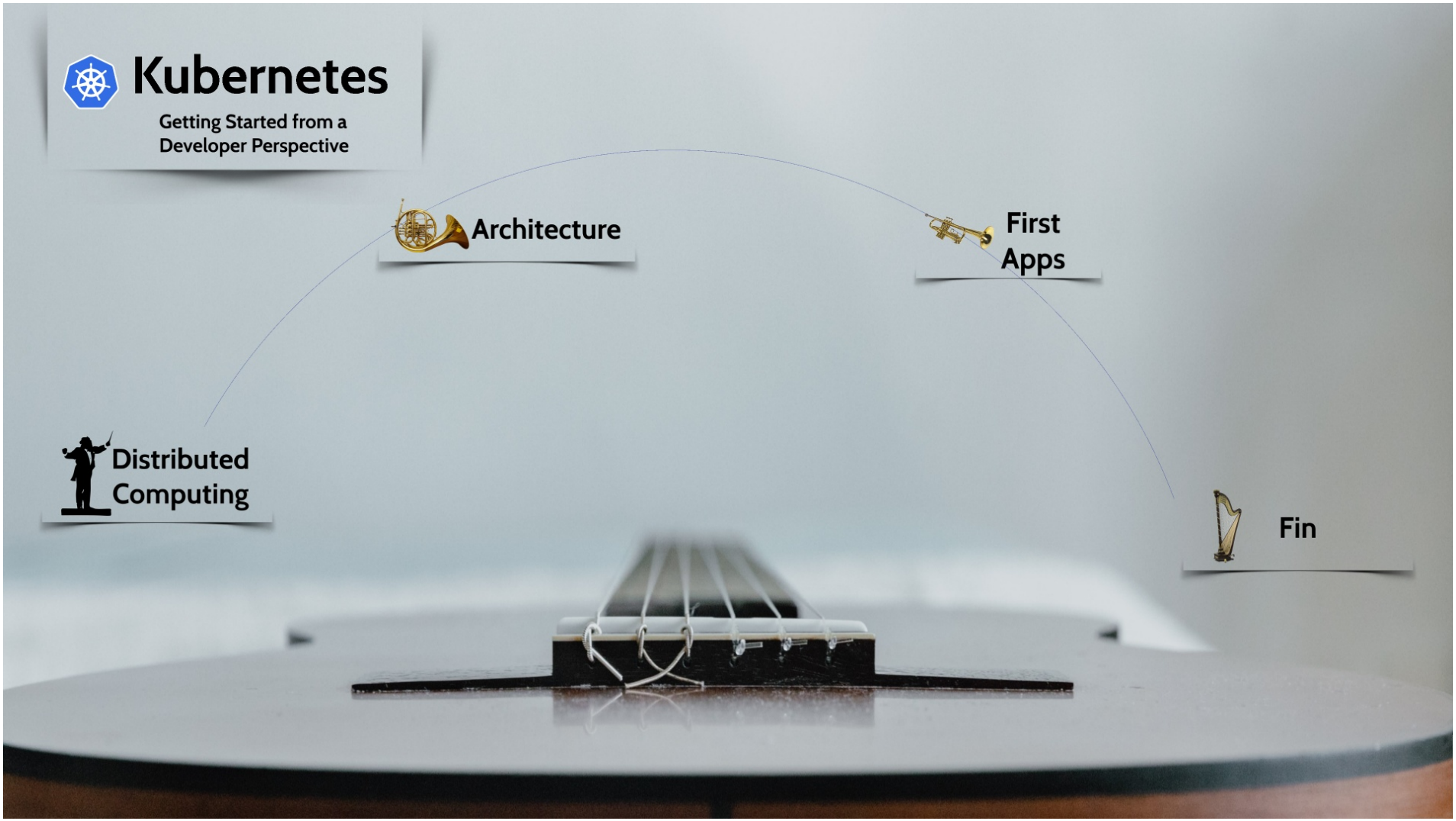
dijure.com

jonathan.johnson
@dijure.com

twitter.com/
javajonjohn

github.com/
javajon

linkedin.com/in/
javajon


Kubernetes
Getting Started from a
Developer Perspective
Architecture
First
Apps
Distributed
Computing
Fin