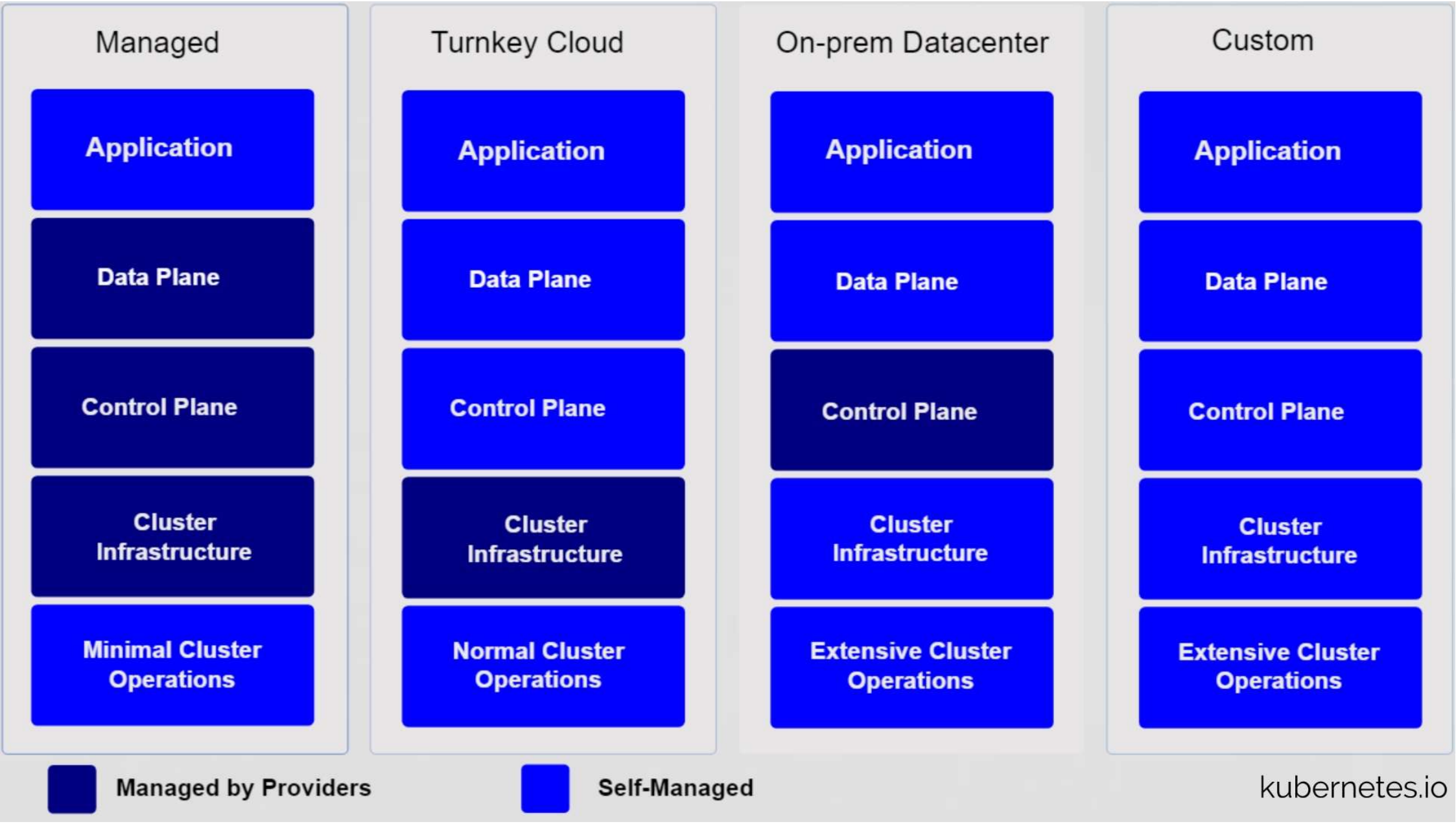
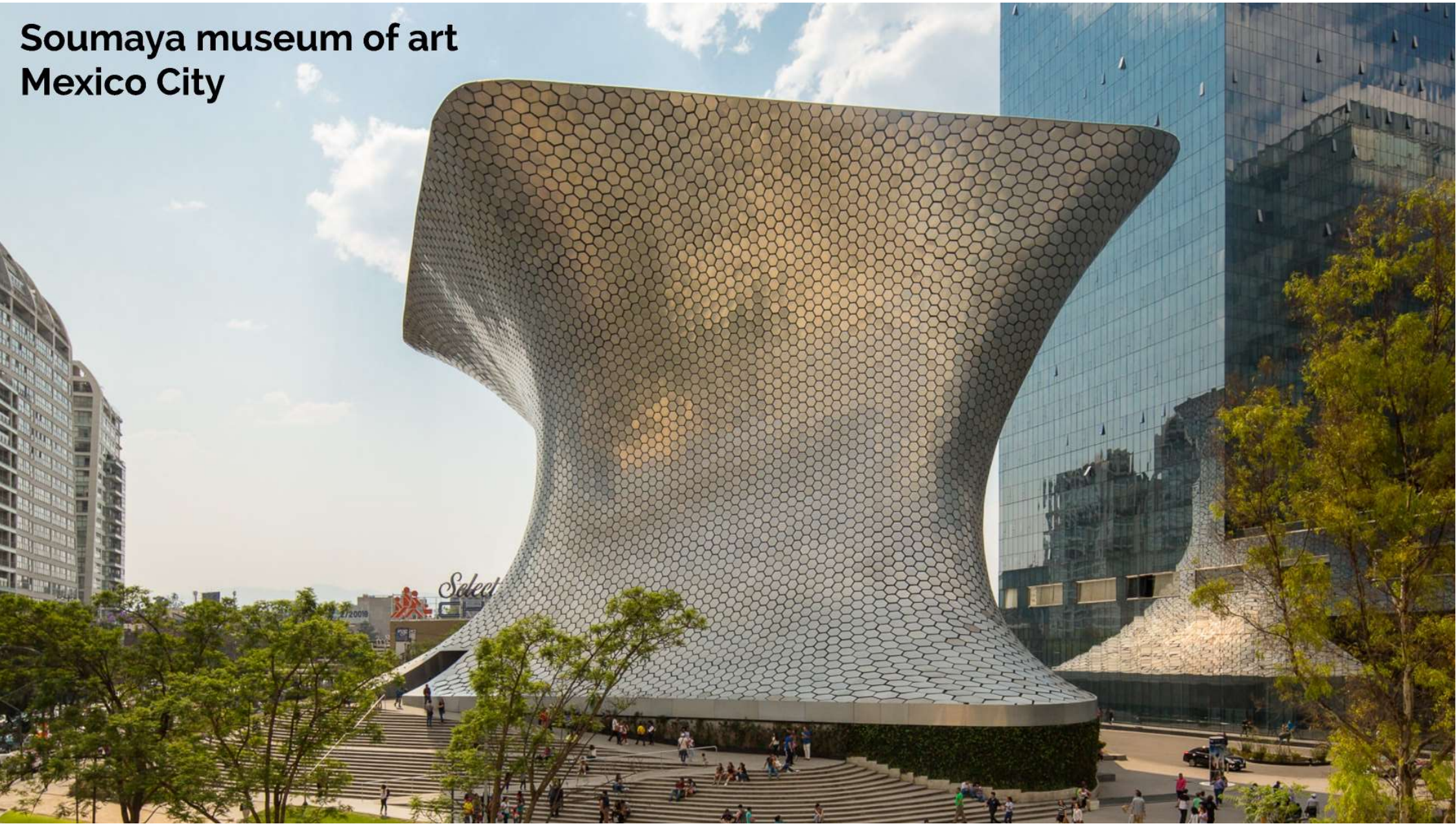
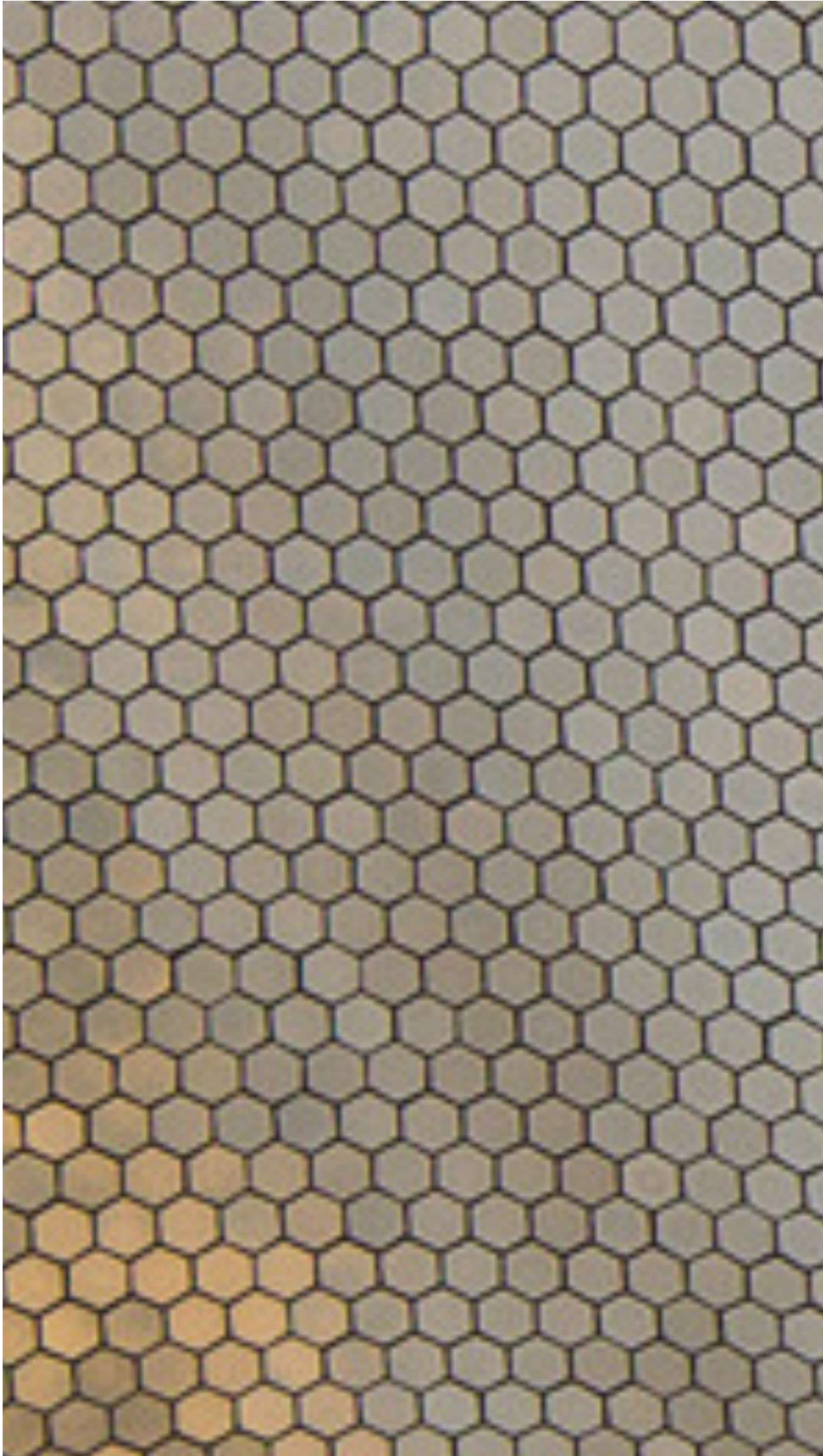

Managed
Application
Data Plane
Control Plane
Cluster Infrastructure
Minimal Cluster Operations
Turnkey Cloud
Application
Data Plane
Control Plane
Cluster Infrastructure
Normal Cluster Operations
On-prem Datacenter
Application
Data Plane
Control Plane
Cluster Infrastructure
Extensive Cluster Operations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

Soumaya museum of art
Mexico City



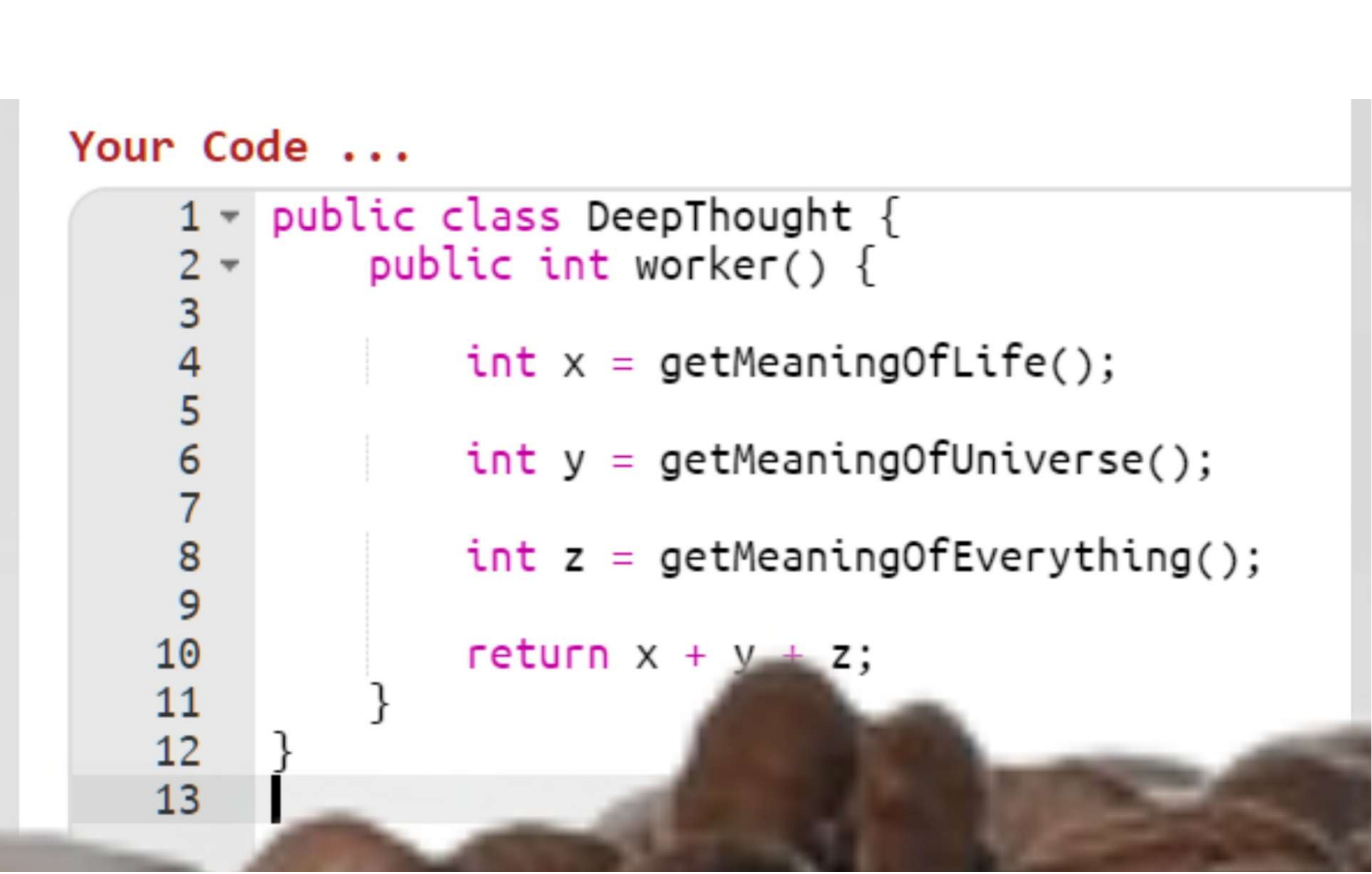


Your Code ...

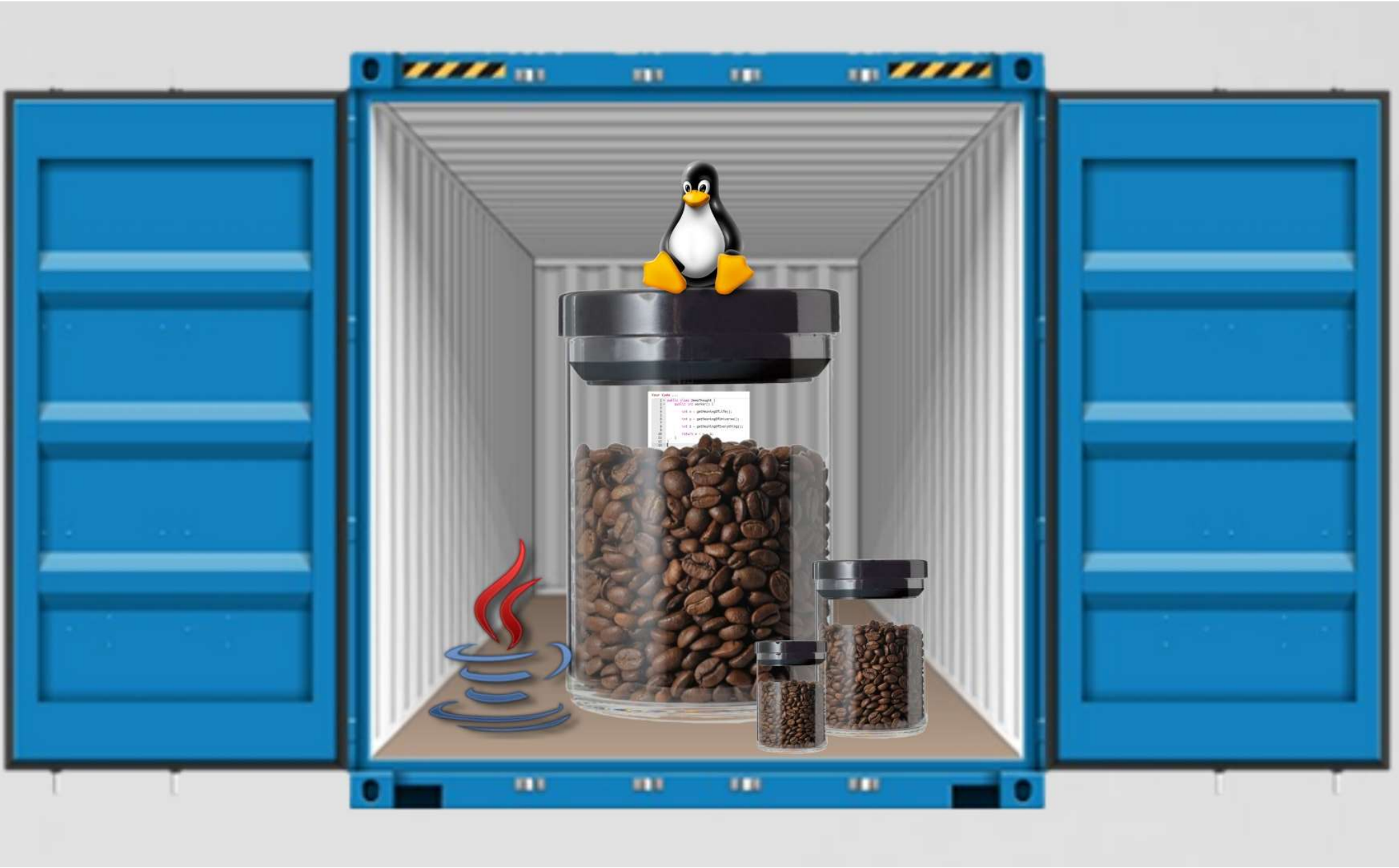
```
1  public class DeepThought {
2      public int worker() {
3
4          int x = getMeaningOfLife();
5
6          int y = getMeaningOfUniverse();
7
8          int z = getMeaningOfEverything();
9
10         return x + y + z;
11     }
12 }
13
```

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 +
}
}




SHIPPING
CONTAINER

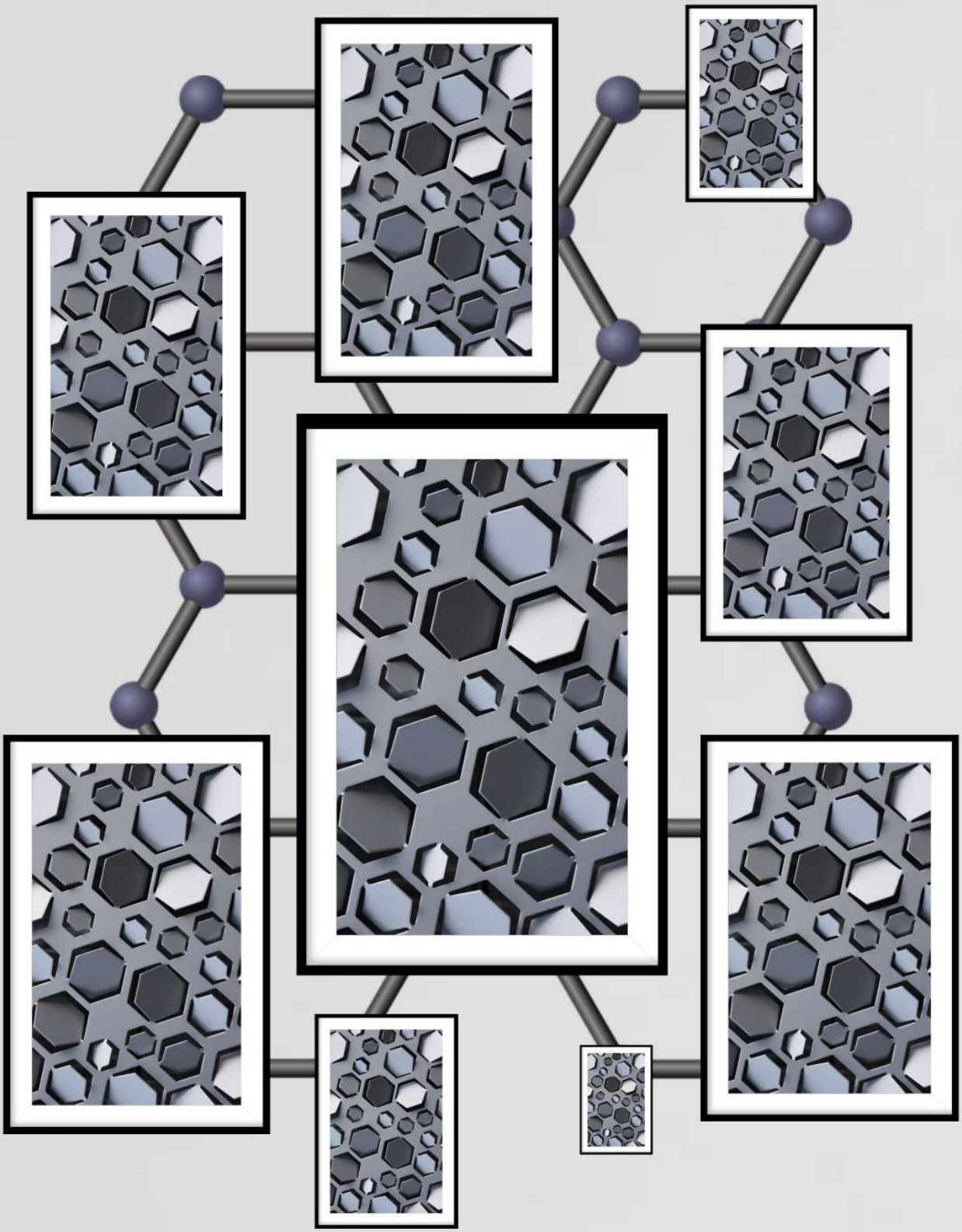

SHIPPING
CONTAINER





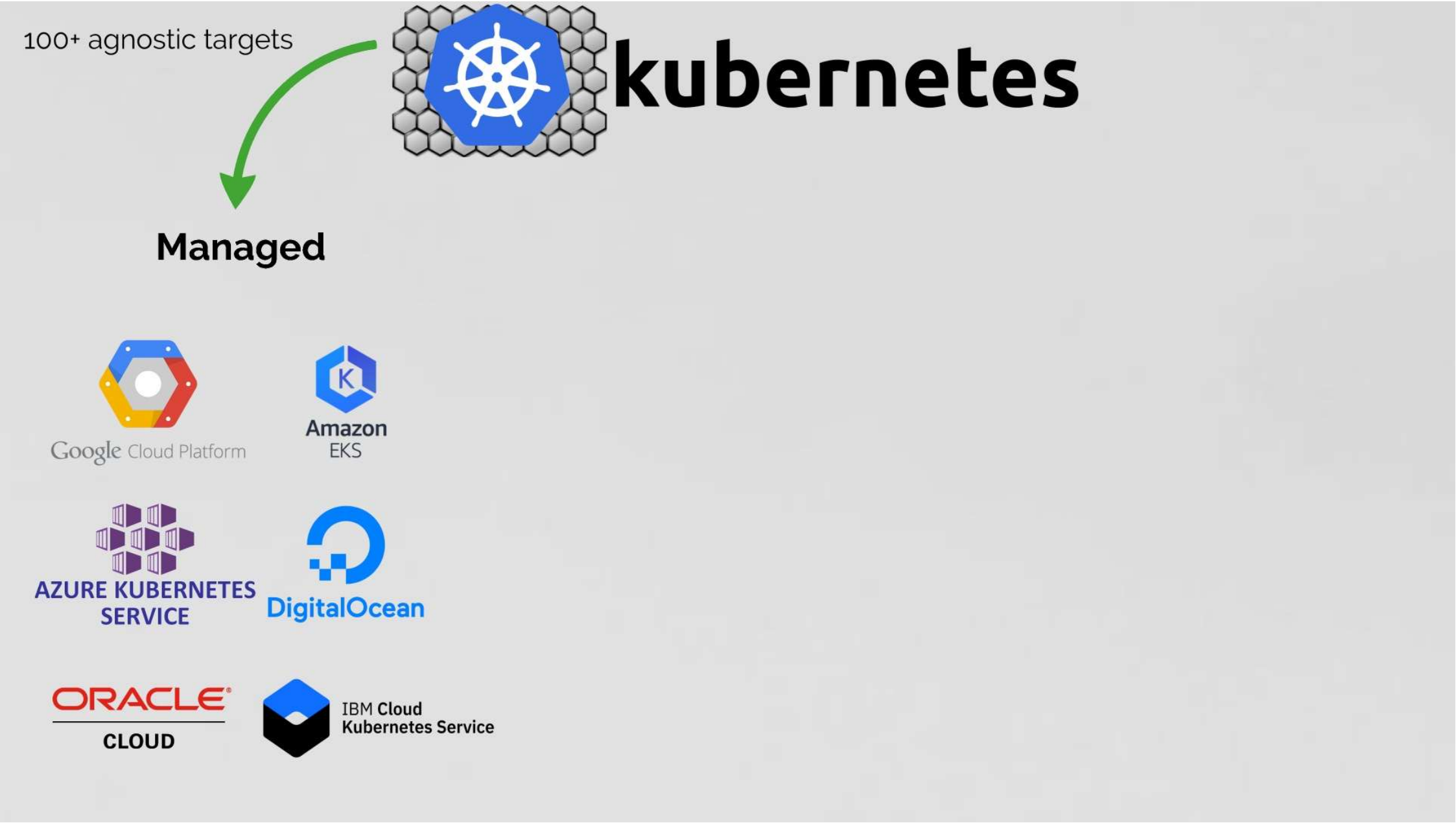






100+ agnostic targets

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

Anthos

Red Hat OpenShift Container Platform

MIRANTIS

RANCHER

100+ agnostic targets

kubernetes

Managed

Google Cloud Platform

Amazon EKS

AZURE KUBERNETES SERVICE

DigitalOcean

ORACLE CLOUD

IBM Cloud Kubernetes Service

Hybrid, on-prem

Anthos

Red Hat OpenShift Container Platform

MIRANTIS

RANCHER

Custom, Edge

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

- Namespaces
- 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
- Namespaces
- 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

In a **Cluster** each machine is a **Node**. (physical, bare metal, virtual)

Cluster Operating System

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

In a **Cluster** each machine is a **Node**. (physical, bare metal, virtual)

Control-plane nodes run processes that manage cluster

Cluster Operating System

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

In a **Cluster** each machine is a **Node**. (physical, bare metal, virtual)

Control-plane nodes run processes that manage cluster

Worker nodes run your applications

Cluster Operating System

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

In a **Cluster** each machine is a **Node**. (physical, bare metal, virtual)

Control-plane nodes run processes that manage cluster

Worker nodes run your applications

Pods are groupings of one or more containers

Cluster Operating System

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

In a **Cluster** each machine is a **Node**. (physical, bare metal, virtual)

Control-plane nodes run processes that manage cluster

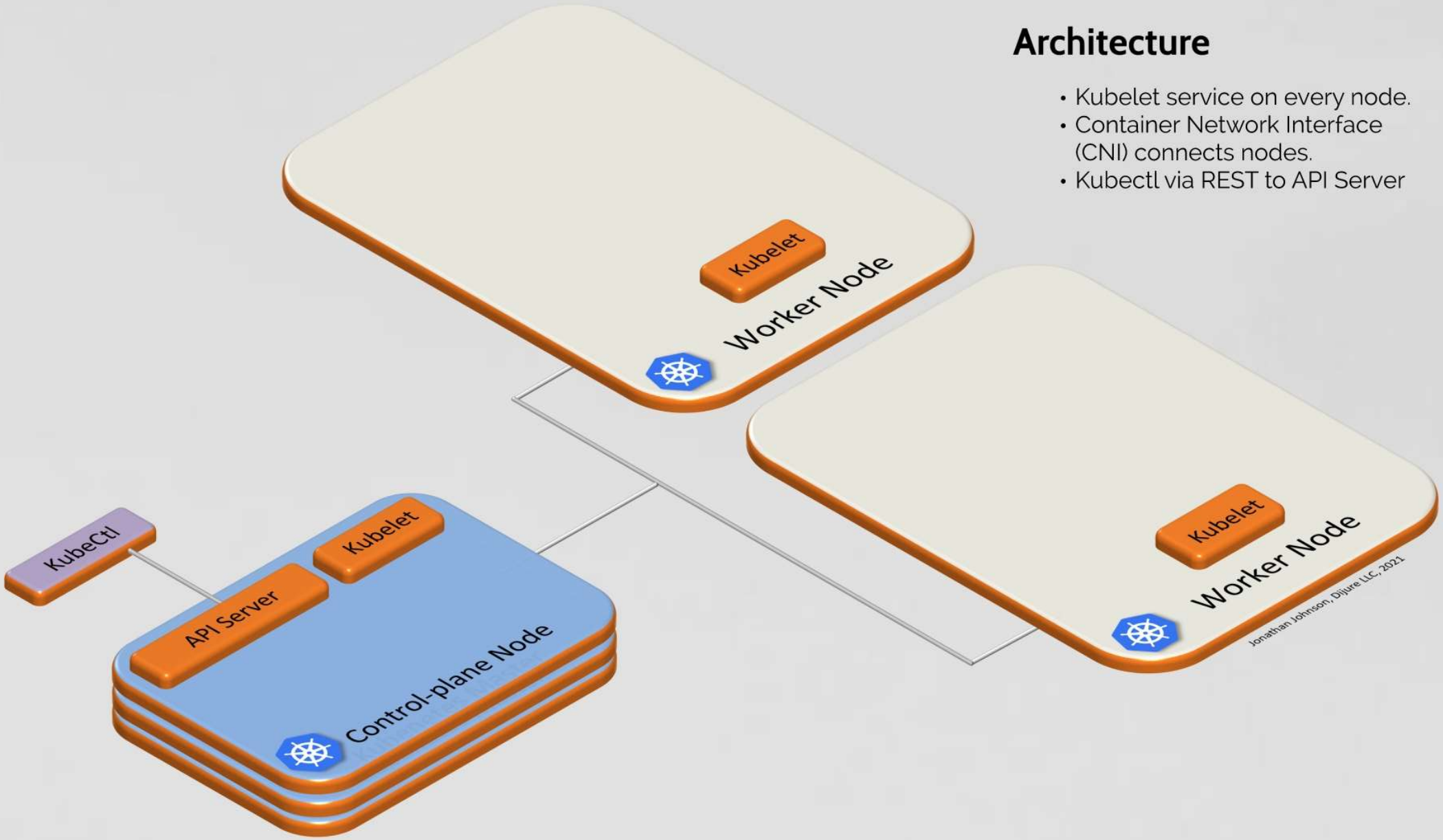
Worker nodes run your applications

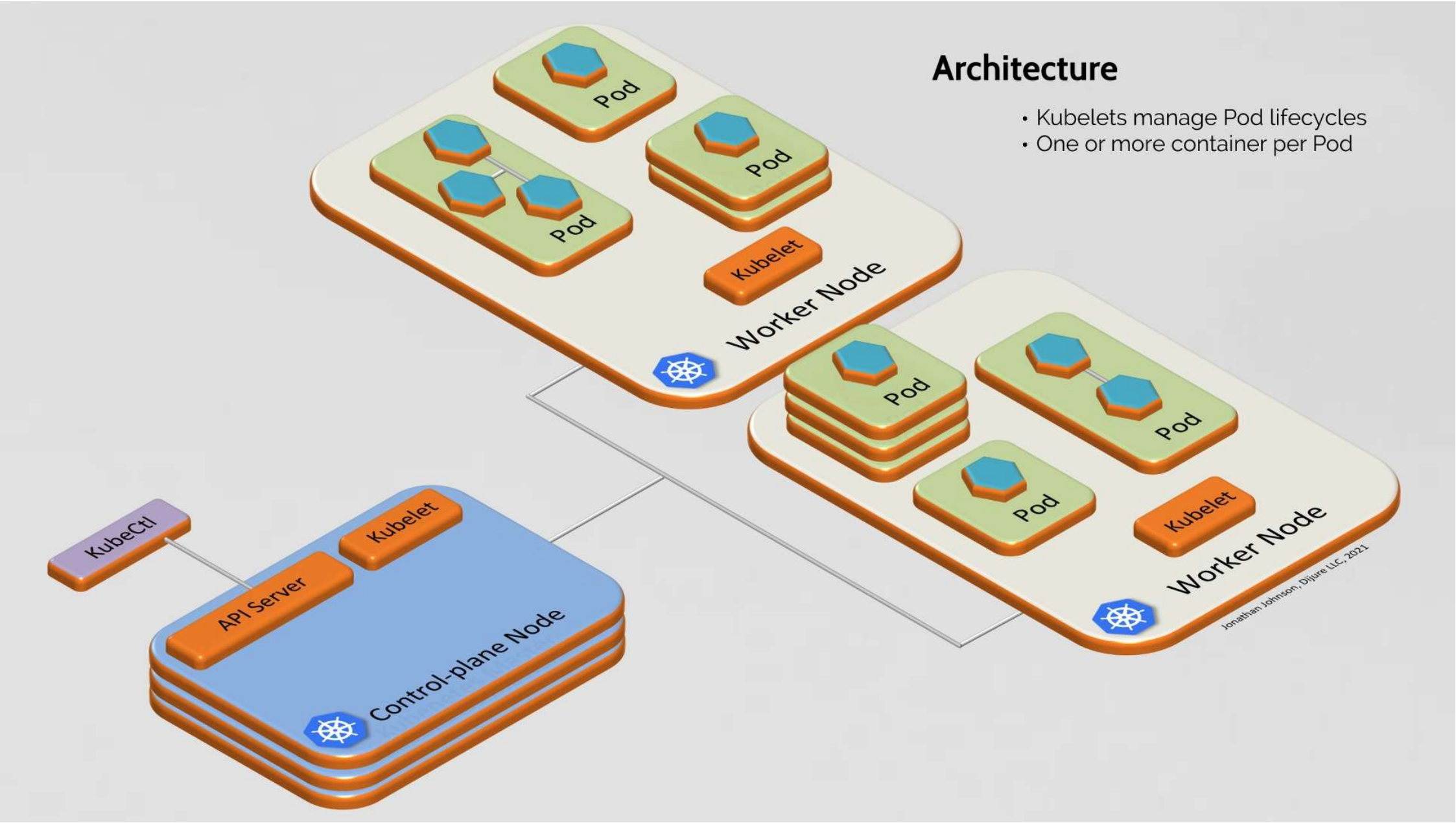
Pods are groupings of one or more containers

Services load balance between replicated pods

Architecture

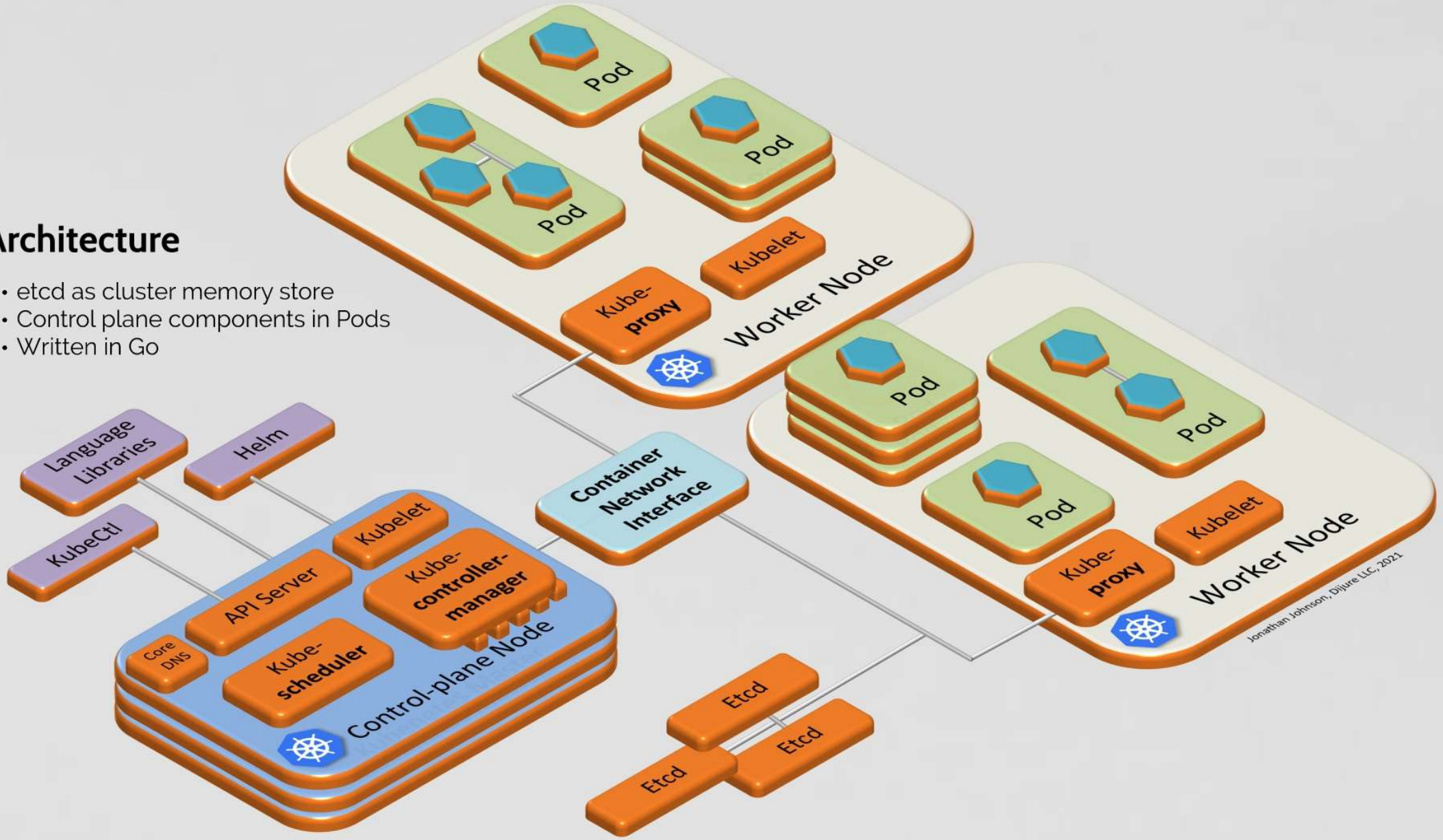
- Kubelet service on every node.
- Container Network Interface (CNI) connects nodes.
- Kubectl via REST to API Server




Architecture
• Kubelets manage Pod lifecycles
• One or more container per Pod
Pod
Pod
Pod
Kubelet
Worker Node
Pod
Pod
Pod
Kubelet
Worker Node
KubeCtl
API Server
Kubelet
Control-plane Node
Jonathan Johnson, Dijure LLC, 2021

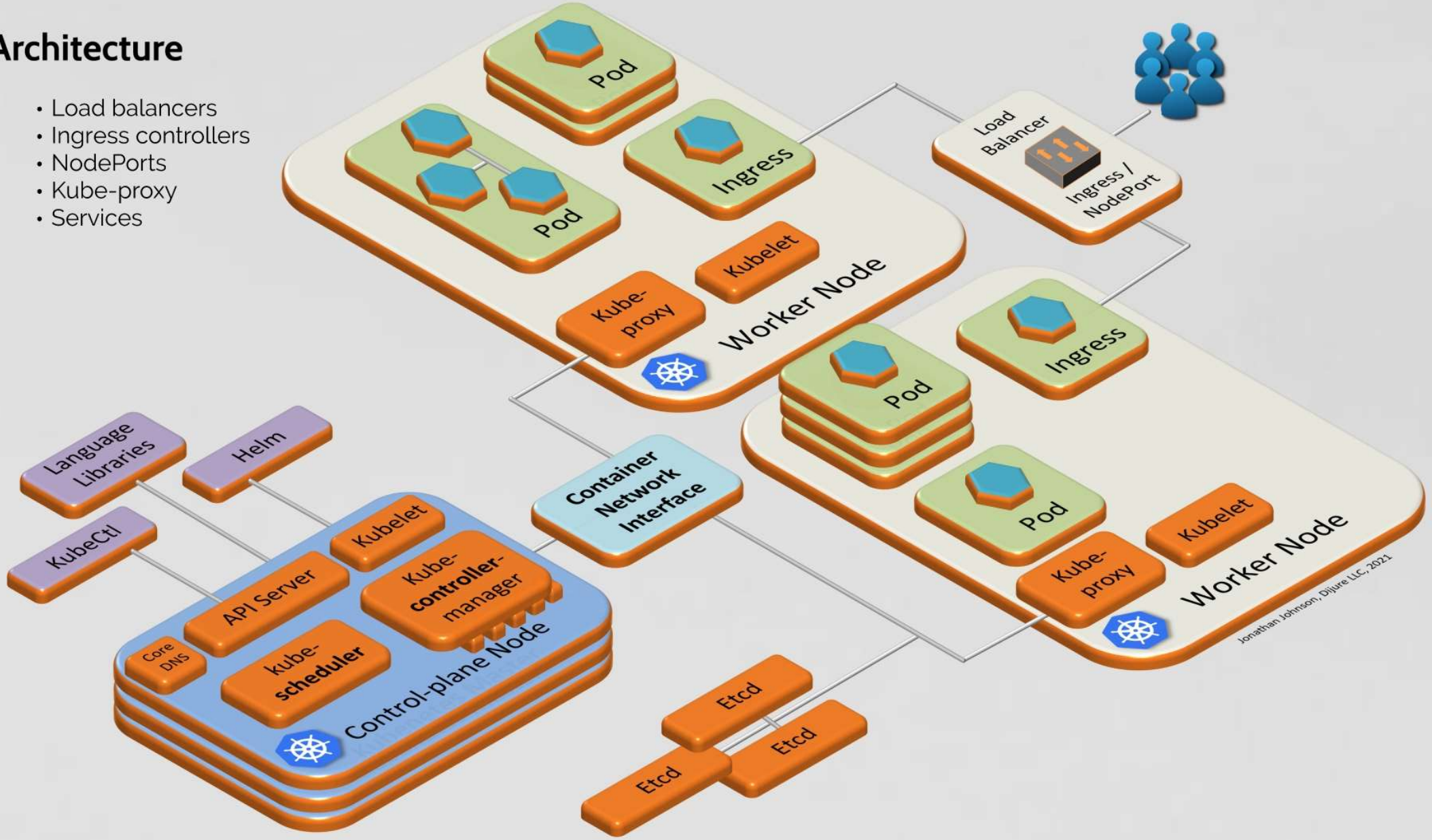
Architecture

- etcd as cluster memory store
- Control plane components in Pods
- Written in Go



Architecture

- Load balancers
- Ingress controllers
- NodePorts
- Kube-proxy
- Services



Kubernetes truths stored here

Distributed key-value store

Readable once consensus reached

Fast reads, slower writes

High available solution at minimum 3 nodes

etcd

Raft Consensus Algorithm

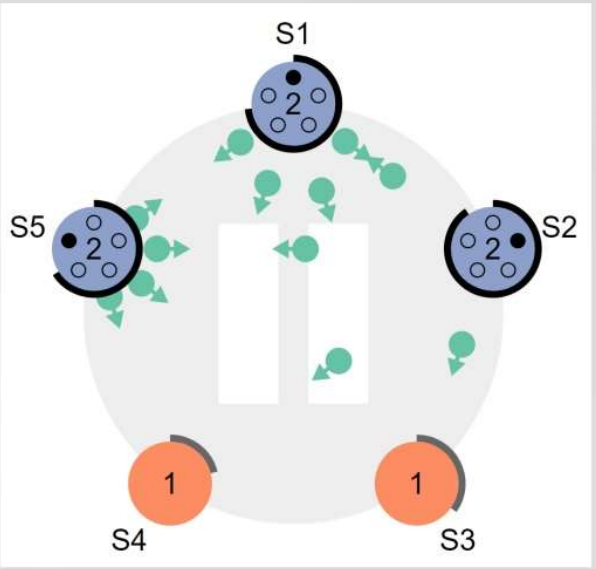
Interactive tutorials:
raft.github.io
thesecretlivesofdata.com/raft

Kubernetes truths stored here

Distributed key-value store

Readable once consensus reached

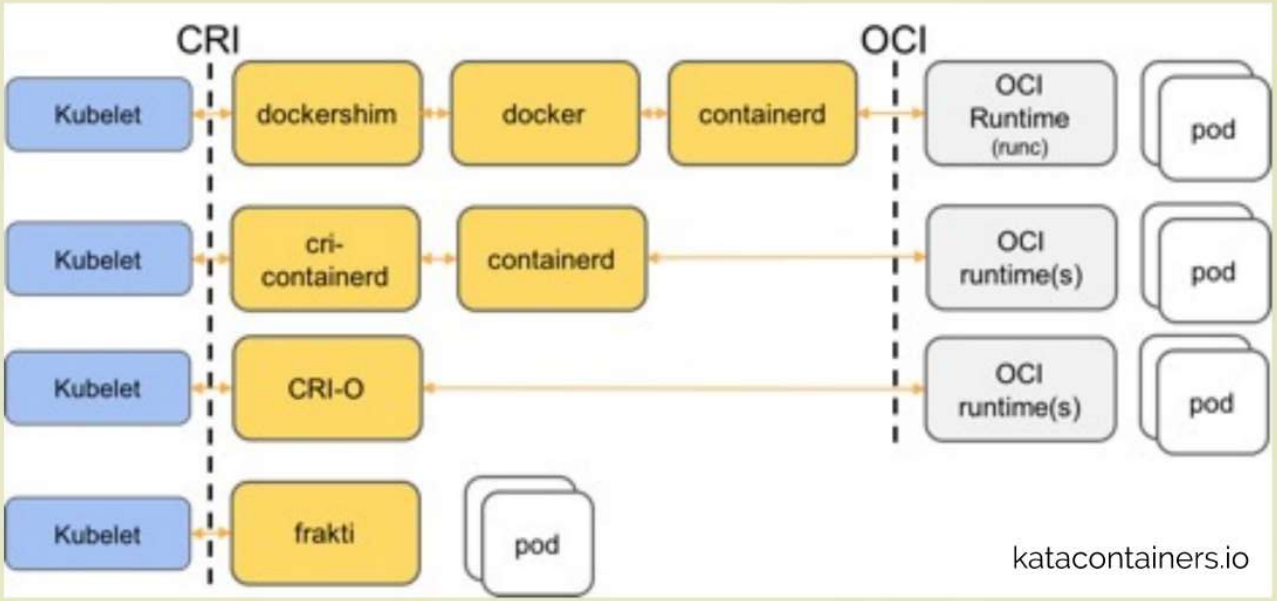
Fast reads, slower writes



High available solution at minimum 3 nodes

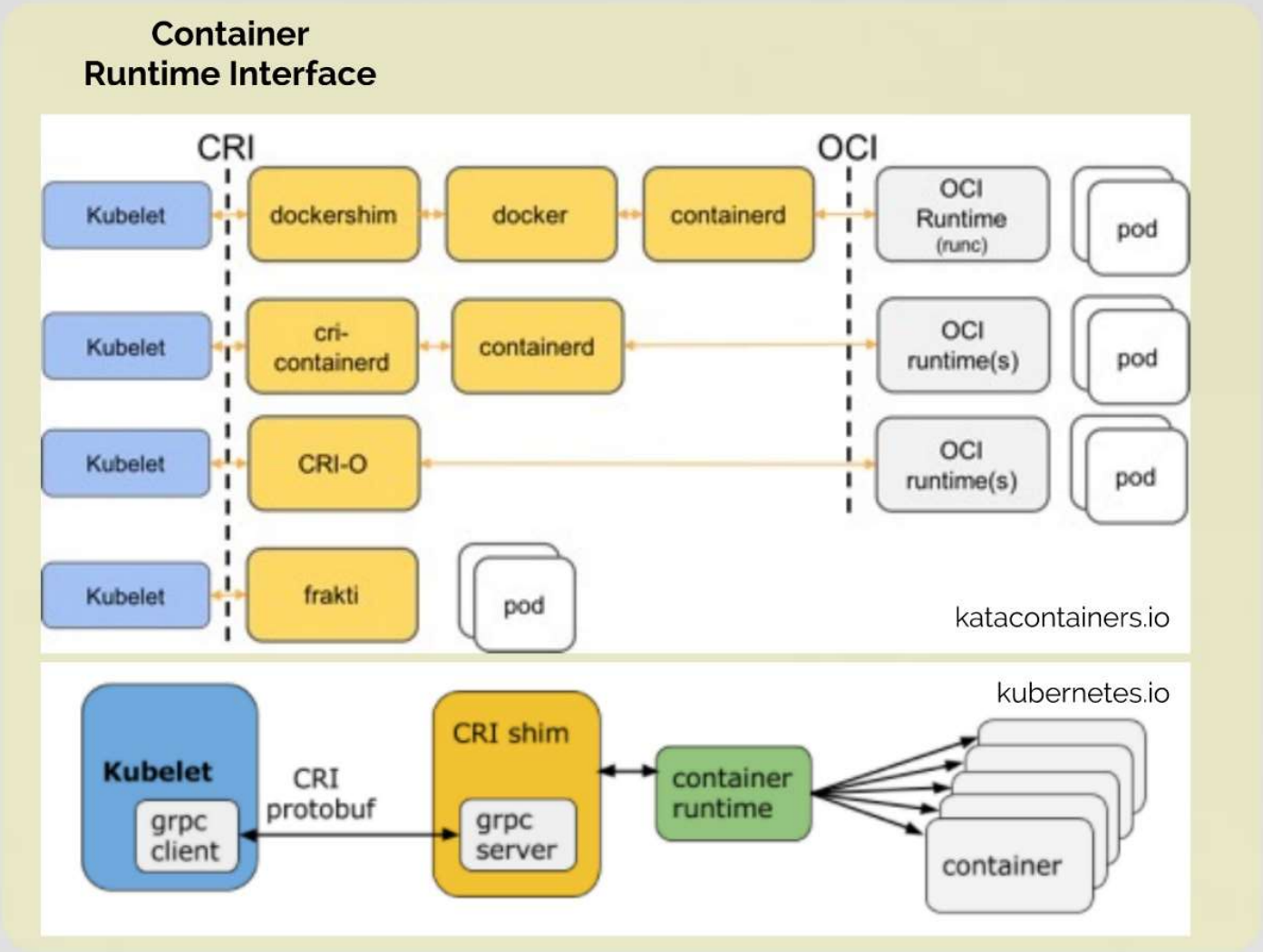
CRI and OCI

OPEN CONTAINER INITIATIVE

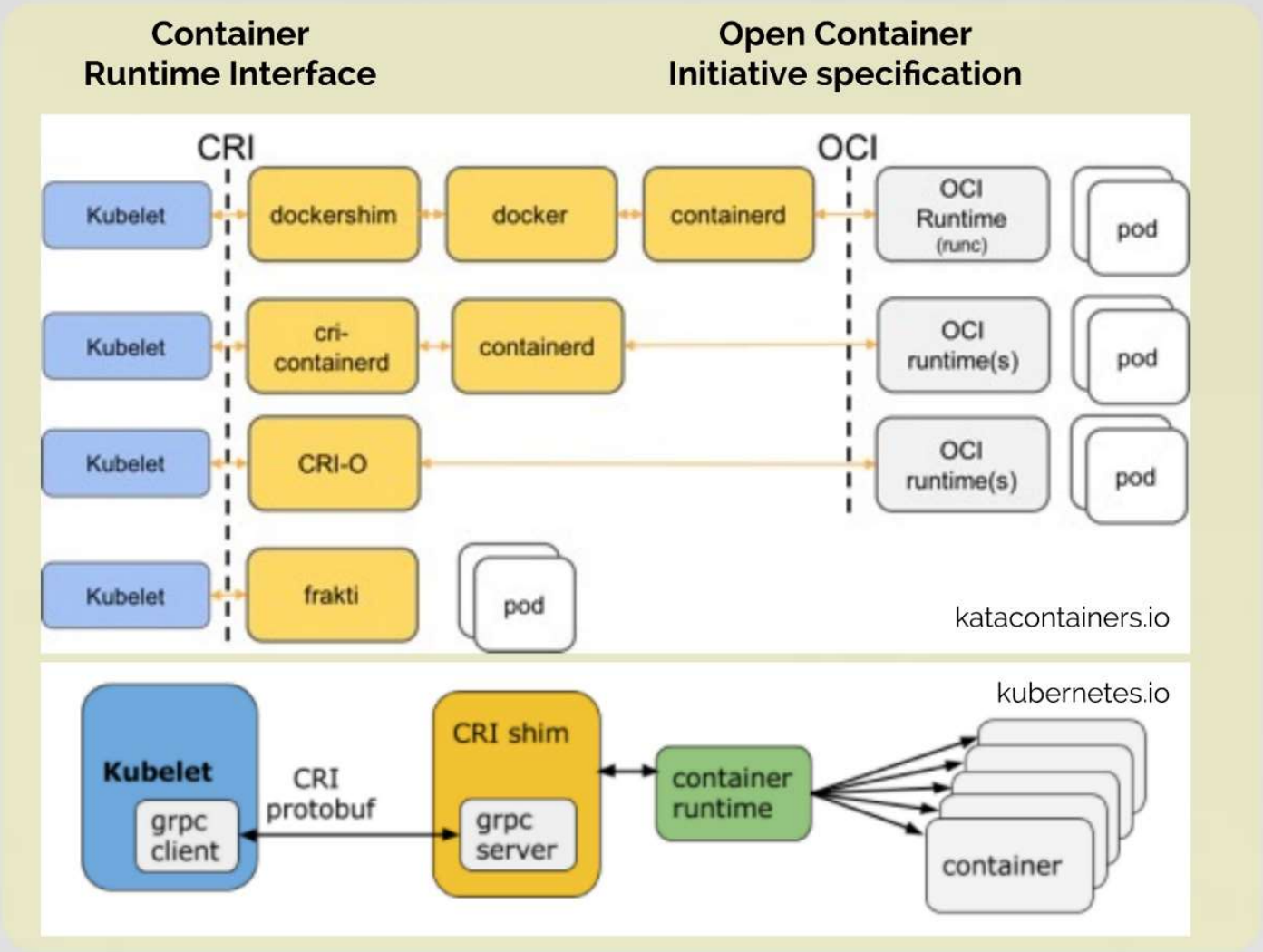


CRI and OCI

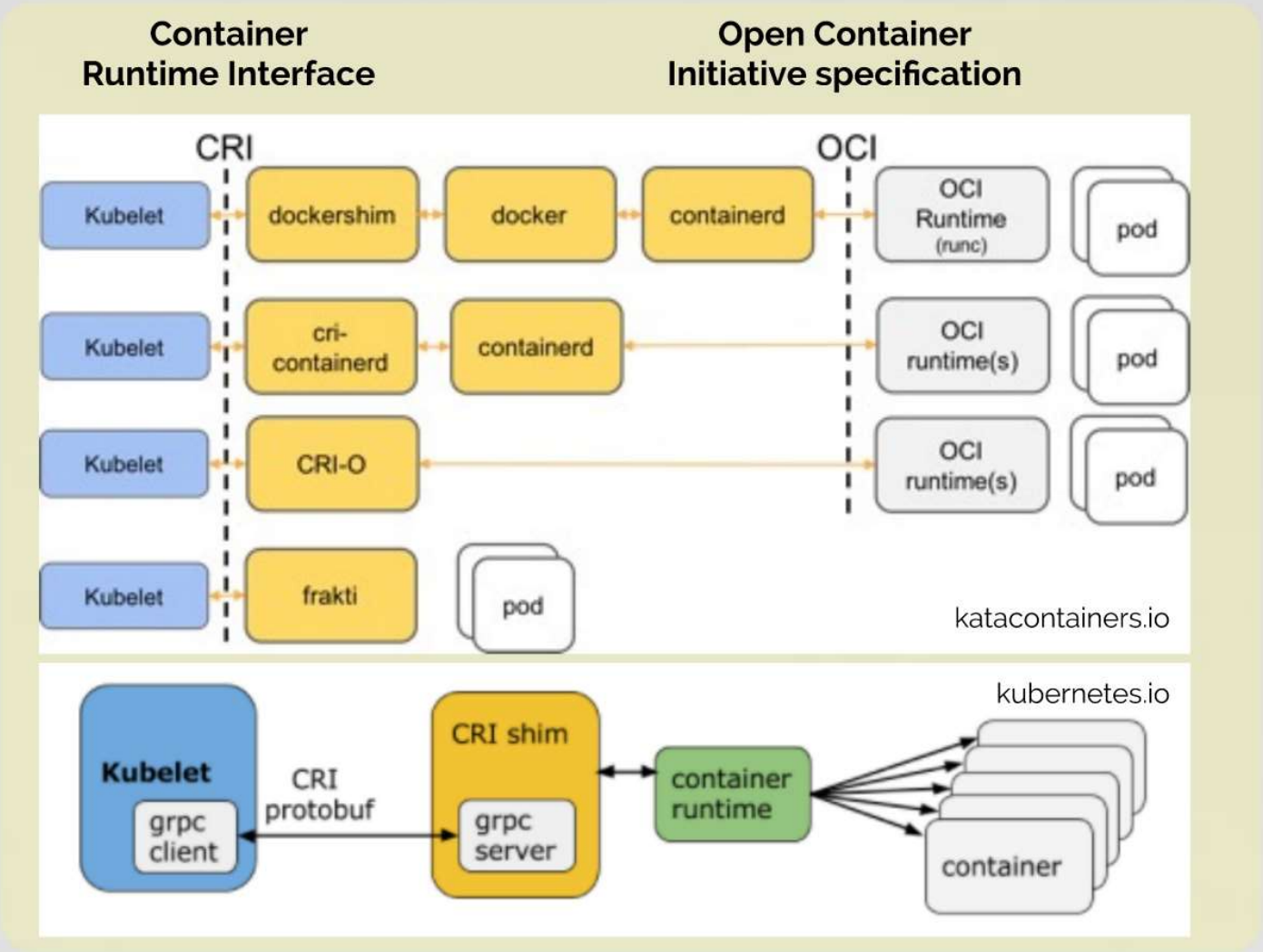
OPEN CONTAINER INITIATIVE



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 (nix Q4 2020)
- cri-o (OpenShift)
- **cri-containerd** (default)
  - runc
  - gVisor
  - Kata containers
  - Firecracker
  - Nabla containers
- crun (c based)
- Rktlet (CRI for Rkt)
- rktnetes
- Frakti
- Singularity
- * Virtual Kubelet

CNCF Container Runtime

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 ticketing -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 ticketing -o YAML
```

Declarative

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ticketing
  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 ticketing -o YAML
```

Declarative

```
apiVersion: v1
kind: Namespace
metadata:
  name: ticketing
  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 ticketing -o YAML
```

Declarative

```
apiVersion: v1
kind: Namespace
metadata:
  name: ticketing
  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 ticketing -o YAML
```

Declarative

```
apiVersion: v1
kind: Namespace
metadata:
  name: ticketing
  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 ticketing -o YAML
```

Declarative

```
apiVersion: v1
kind: Namespace
metadata:
  name: ticketing
  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

```
$ kubectl create namespace ticketing

$ kubectl label namespace ticketing venue=opera watch=cpu

$ kubectl get namespaces

$ kubectl get namespace ticketing -o YAML
```

Imperative

Declarative

YAML — Human friendly data serialization standard

HELM — Package manager for Kubernetes

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

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

- `apiVersion` — Object controller version
- `kind` — Object classification
- `metadata` — Associated data
- `spec` — 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
```

Service Associated to Pods

Labels for search filtering and associations

- Deployments manage ReplicaSets
- ReplicaSets manage Pods
- Services manage Endpoints

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Service Associated to Pods

Labels for search filtering and associations

Deployment

(StatefulSet, Jobs, DaemonSet)

Unique name of object instance

optional object labels, for searching

pod template

optional object labels, for searching

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Service Associated to Pods

Labels for search filtering and associations

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Unique name of object instance

optional object labels, for searching

label for pod grouping

pod template

must match

match for grouping

optional object labels, for searching

Service Associated to Pods

Labels for search filtering and associations

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app.kubernetes.io/name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: nginx-pod-port
  selector:
    run: my-nginx
```

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Unique name of object instance

optional object labels, for searching

label for pod grouping

must match

pod template

match for grouping

optional object labels, for searching

Service Associated to Pods

Labels for search filtering and associations

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app.kubernetes.io/name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: nginx-pod-port
  selector:
    run: my-nginx
```

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Unique name of object instance

optional object labels, for searching

label for pod grouping

pod template

must match

match for grouping

optional object labels, for searching

Service Associated to Pods

Labels for search filtering and associations

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app.kubernetes.io/name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: nginx-pod-port
  selector:
    run: my-nginx
```

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Unique name of object instance

optional object labels, for searching

label for pod grouping

pod template

must match

match for grouping

optional object labels, for searching

Service Associated to Pods

Labels for search filtering and associations

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app.kubernetes.io/name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: nginx-pod-port
  selector:
    run: my-nginx
```

Deployment

(StatefulSet, Jobs, DaemonSet)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx-group
  labels:
    app.kubernetes.io/name: nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
        app.kubernetes.io/name: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:1.22.0-alpine
        ports:
        - name: nginx-pod-port
          containerPort: 80
```

Unique name of object instance

optional object labels, for searching

label for pod grouping

pod template

must match

match for grouping

optional object labels, for searching

connect ports

Service Associated to Pods

Labels for search filtering and associations

Declaring ConfigMaps and Secrets

Data can be linked to:

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

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

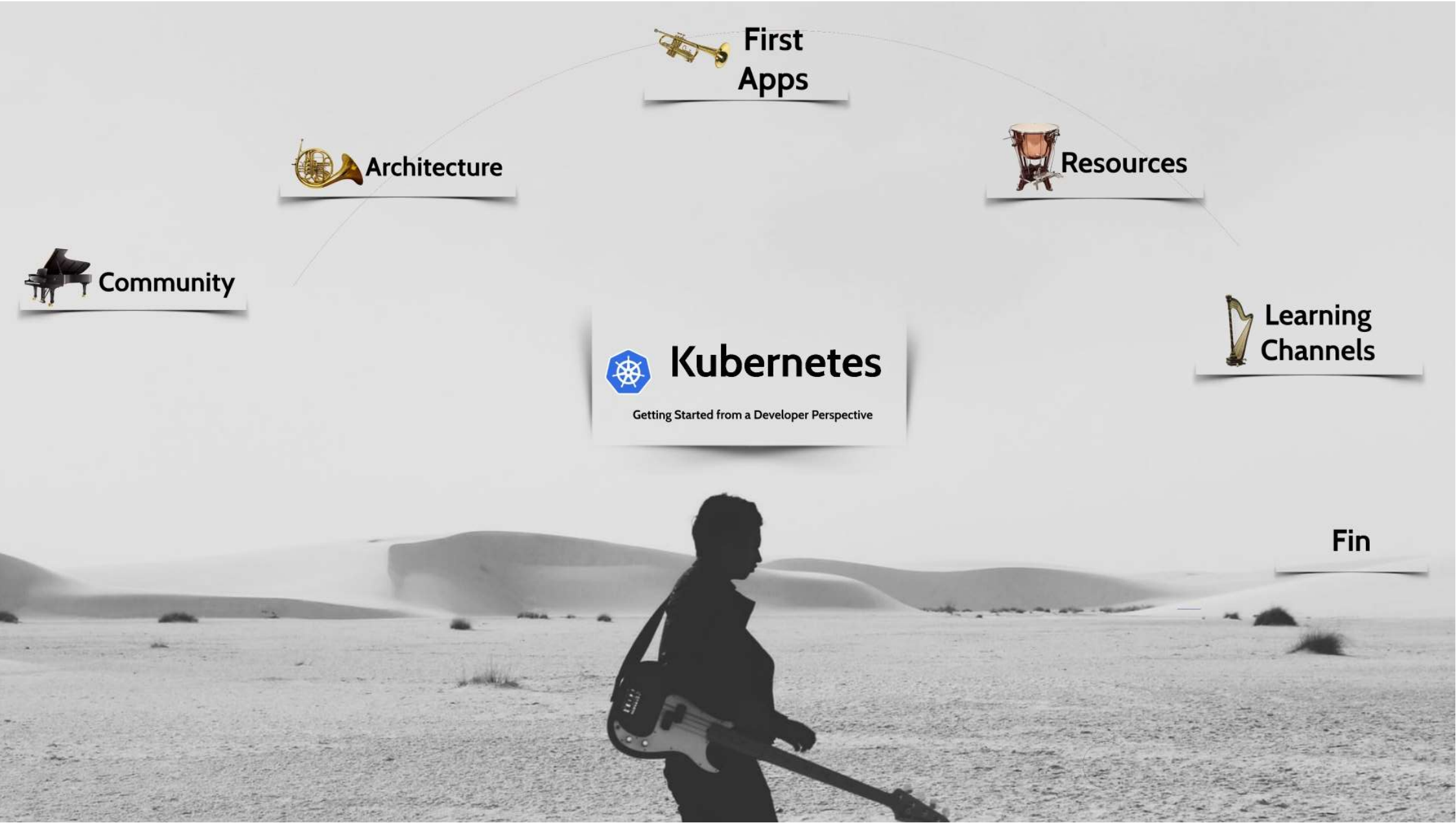
Probes

Checking Container and Pod Health

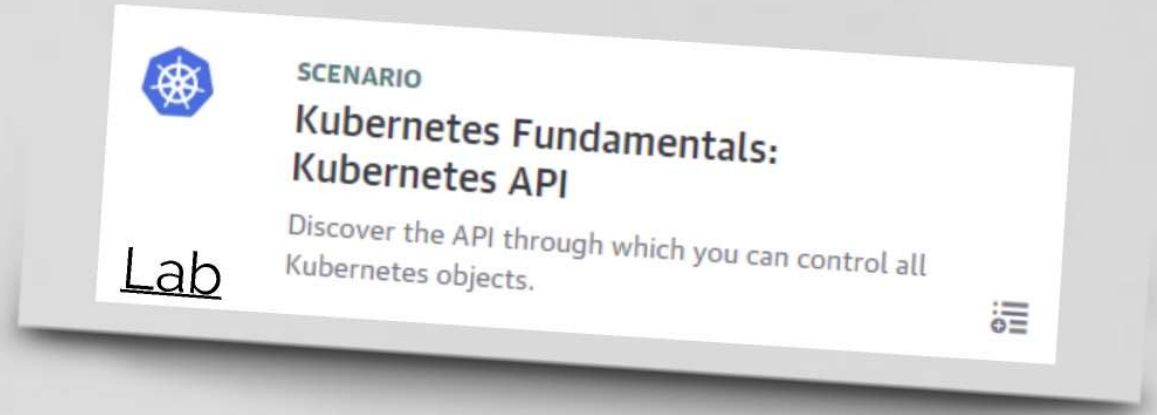
Startup
Liveness
Readiness

- HTTP Status codes
- TCP Connection
- gRPC, HTTP
- Exec, zero=pass, non zero=fail

```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
      failureThreshold: 3
      initialDelaySeconds: 45
      periodSeconds: 30
```


First Apps
Architecture
Resources
Community
Learning Channels
Kubernetes
Getting Started from a Developer Perspective
Fin

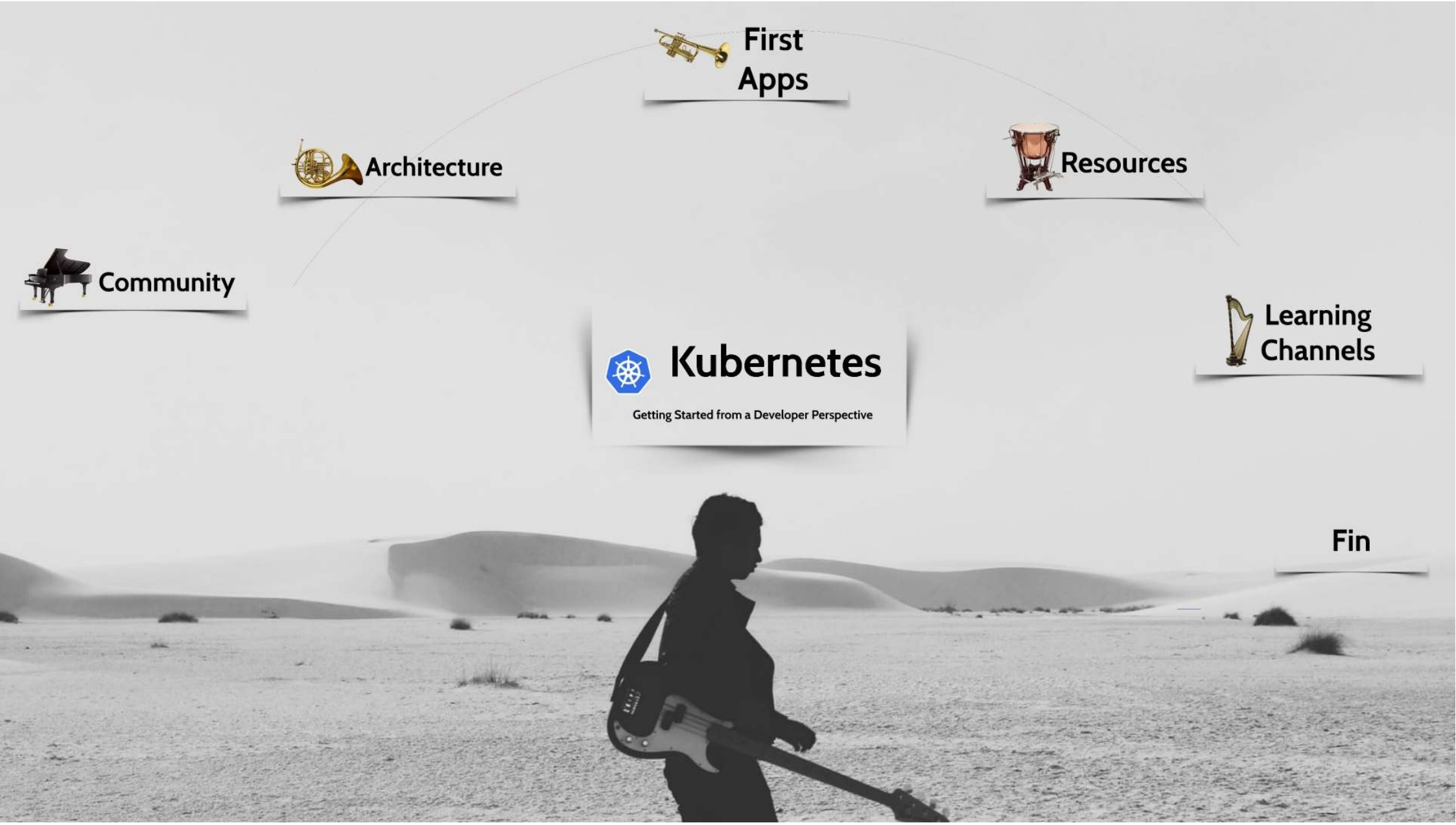
API and Objects



Lab 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

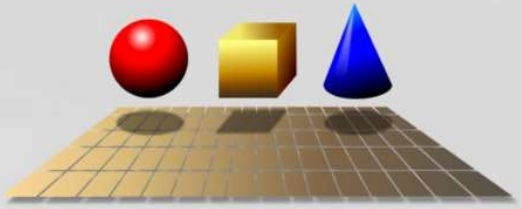
```
$ kubectl get --raw
```


First Apps
Architecture
Resources
Community
Learning Channels
Kubernetes
Getting Started from a Developer Perspective
Fin

Kubernetes Resources

Declaring State

Resources represent
state of cluster

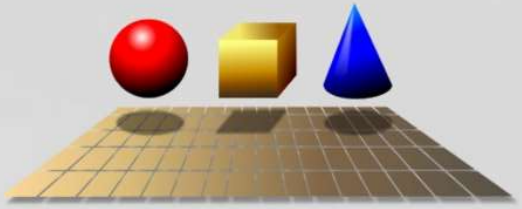


Kubernetes Resources

Declaring State

Resources represent state of cluster

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

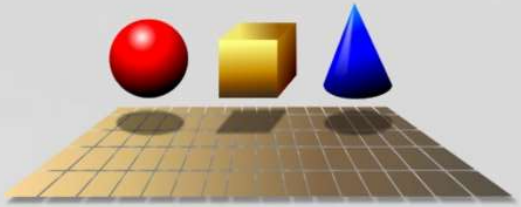


Kubernetes Resources

Declaring State

Resources represent state of cluster

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



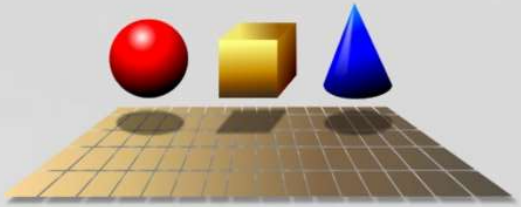
Instantiated resources are persisted objects across cluster

Kubernetes Resources

Declaring State

Resources represent state of cluster

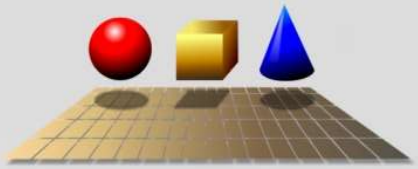
Resources declared in Kubernetes manifests (YAMLs) as "Kinds"



Instantiated resources are persisted objects across cluster

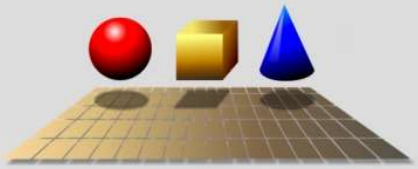
Objects managed via API and often through kubectl

Kubernetes Resources



They will be
Kind to you

Kubernetes Resources



They will be **Kind** to you

Pod Contructs

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

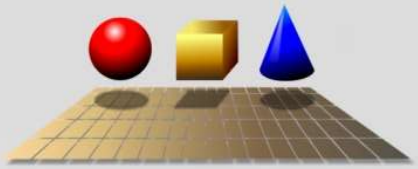
Context / Data

- ConfigMap
- Secrets

Networking

- Ingress
- Service
- NetworkPolicy
- Endpoints

Kubernetes Resources



They will be **Kind** to you

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

Context / Data
- ConfigMap
- Secrets

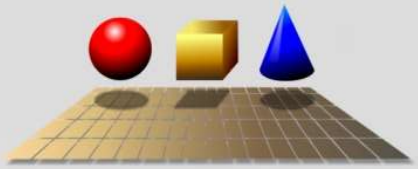
Networking
- Ingress
- Service
- NetworkPolicy
- Endpoints

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

RBAC
- ServiceAccount
- Role
- RoleBinding

Cluster Scope
- Node
- Namespace
- CustomResourceDefinition

Kubernetes Resources



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

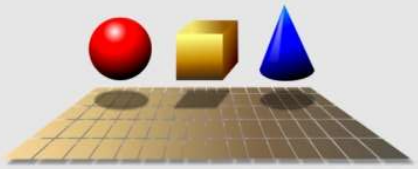
Persistence
- Volumes
- Persistent Volumes
- StorageClass
- CSIDrivers
- CSINodes

RBAC
- ServiceAccount
- Role
- RoleBinding

Cluster Scope
- Node
- Namespace
- CustomResourceDefinition

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

~55 Standard Kubernetes Resources



They will be **Kind** to you

```
APIService
ClusterRole
ClusterRoleBindiing
ConfigMap
CronJob
CSIDriver
CSINode
DaemonSet
Deployment
EphemeralContainers
HorizontalPodAutoscalar
Ingress
IngressClass
Job
Namespace
Node
PersistentVolume
Pod
PodDisruptionBudget
PodTemplate
ReplicaSet
ResourceQuota
Role
RoleBinding
Secret
Service
ServiceAccount
StatefulSet
StorageClass
VolumeAttachment
```

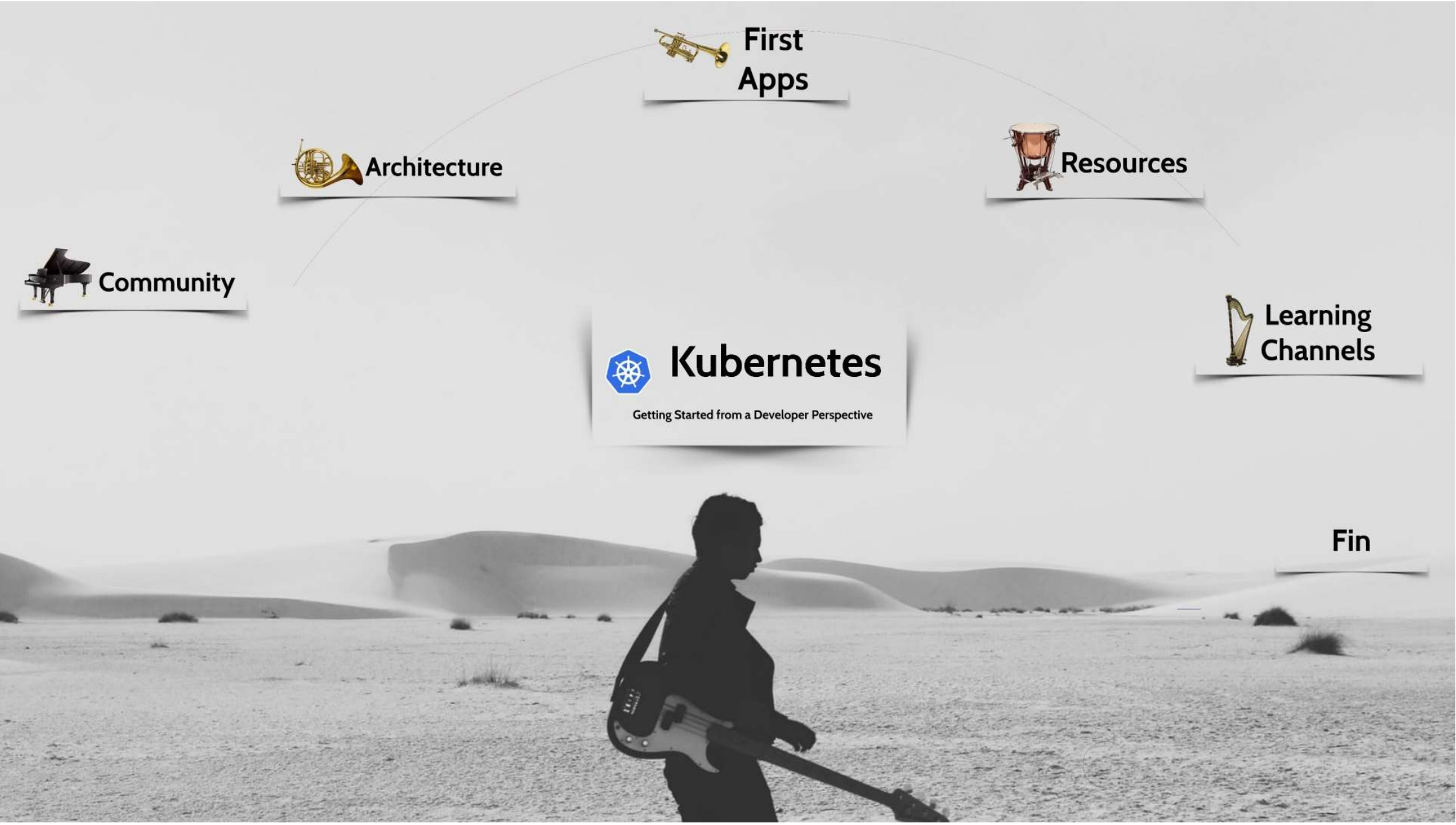
```
$ kubectl api-resources
```

We touched on many

There are more

There are custom resources too !!

```
Binding
CertificateSigningRequest
ComponentStatus
ControllerRevisiion
CustomResourceDef
Endpoints
EndpointSlice
LeaseReplicationController
LimitRange
LocalSubjectAccess
MutatingWebhookConfiguration
NetworkPolicy
PodSecurityPolicy
PriorityClass
RuntimeClass
SelfSubjectAccess
SelfSubjectRules
SubjectAccessReview
TokenReview
ValidatingWebhook
```


First Apps
Architecture
Resources
Community
Learning Channels
Kubernetes
Getting Started from a Developer Perspective
Fin

Original Google Team 1998

Sergey Brin
Larry Page
Craig Silverstein
Heather Cairns
Ray Sidney
Harry Cheung
Amit Patel
Urs Hölzle
Georges Harik
Salar Kamangar
Omid Kordestani
Rachael Chambers
Chris Skarakis
Joan Braddi
Susan Wojcicki
Gerald Aigner
Jim Reese
Larry Schwimmer
Kendra DiGirolamo
Marissa Mayer