STATEFUL APPLICATIONS IN KUBERNETES: READY FOR PRODUCTION!

kasten

Niraj Tolia, Co-Founder
@nirajtolia / ntolia@kasten.io

Julio Lopez, Member of Technical Staff
@julio5524 / julio@kasten.io

Kubernetes

Container Orchestration:
Automated Deployment, Scaling, & Management

Kubernetes, the greatest thing since sliced bread?

kubernetes philosophy

Developer and Application Focused

Puts the needs of the application and developer first and optimizes for agility

Enforces Good DevOps Hygiene

Immutability, config as code, automation makes it easy to repave all infrastructure

Declarative Approach

A robust systems approach where the state of the world is reconciled with the expectation

key kubernetes features

Self-Healing

Auto restart of unhealthy containers to match service levels

Resource Utilization

Better bin packing for higher resource utilization

Deployment Options

Variety of upgrade deployment strategies w/ rollback options

Auto Scaling

Scale applications up and down in response to load

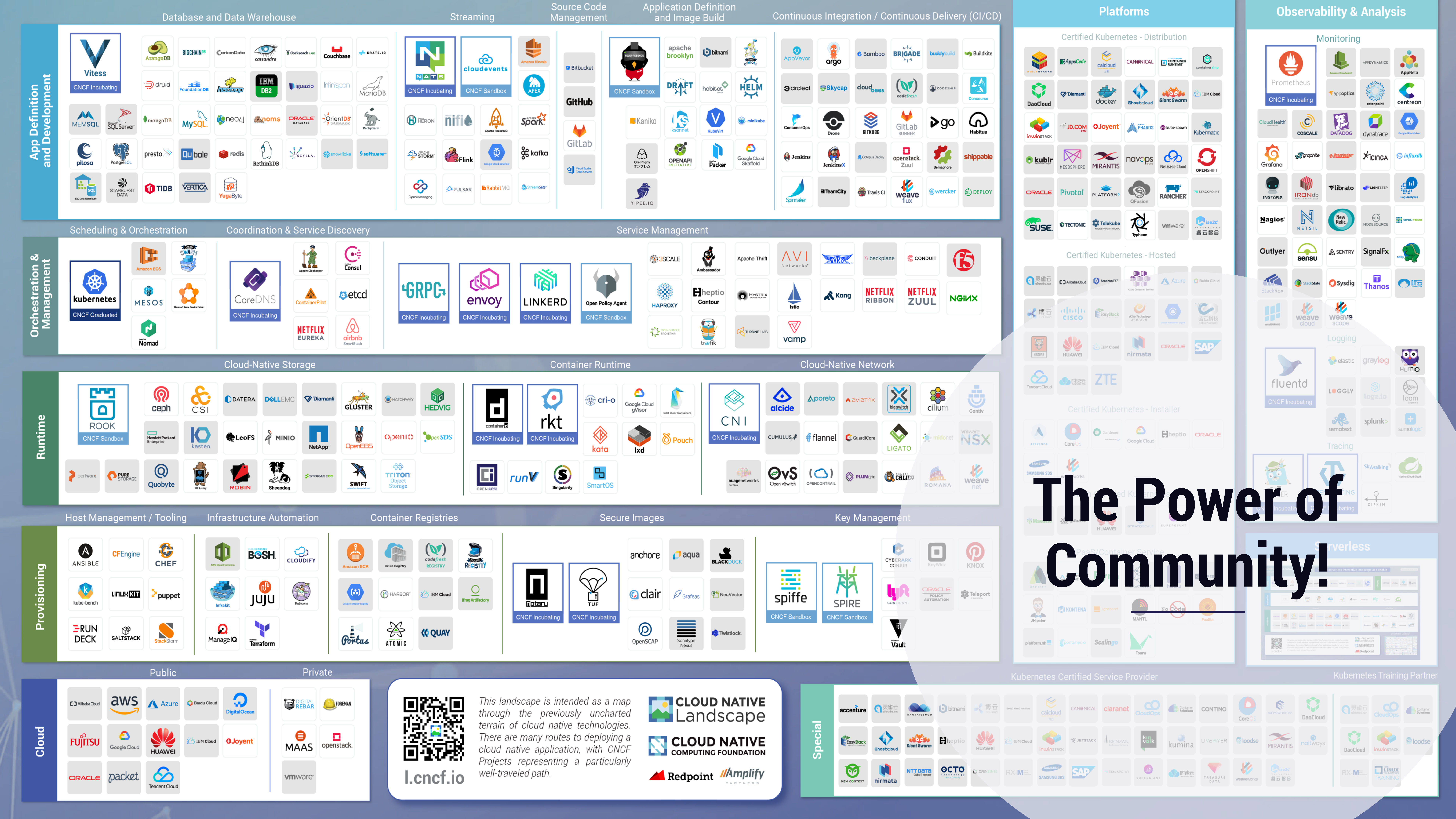
Portability

Isolates developers and applications from infrastructure

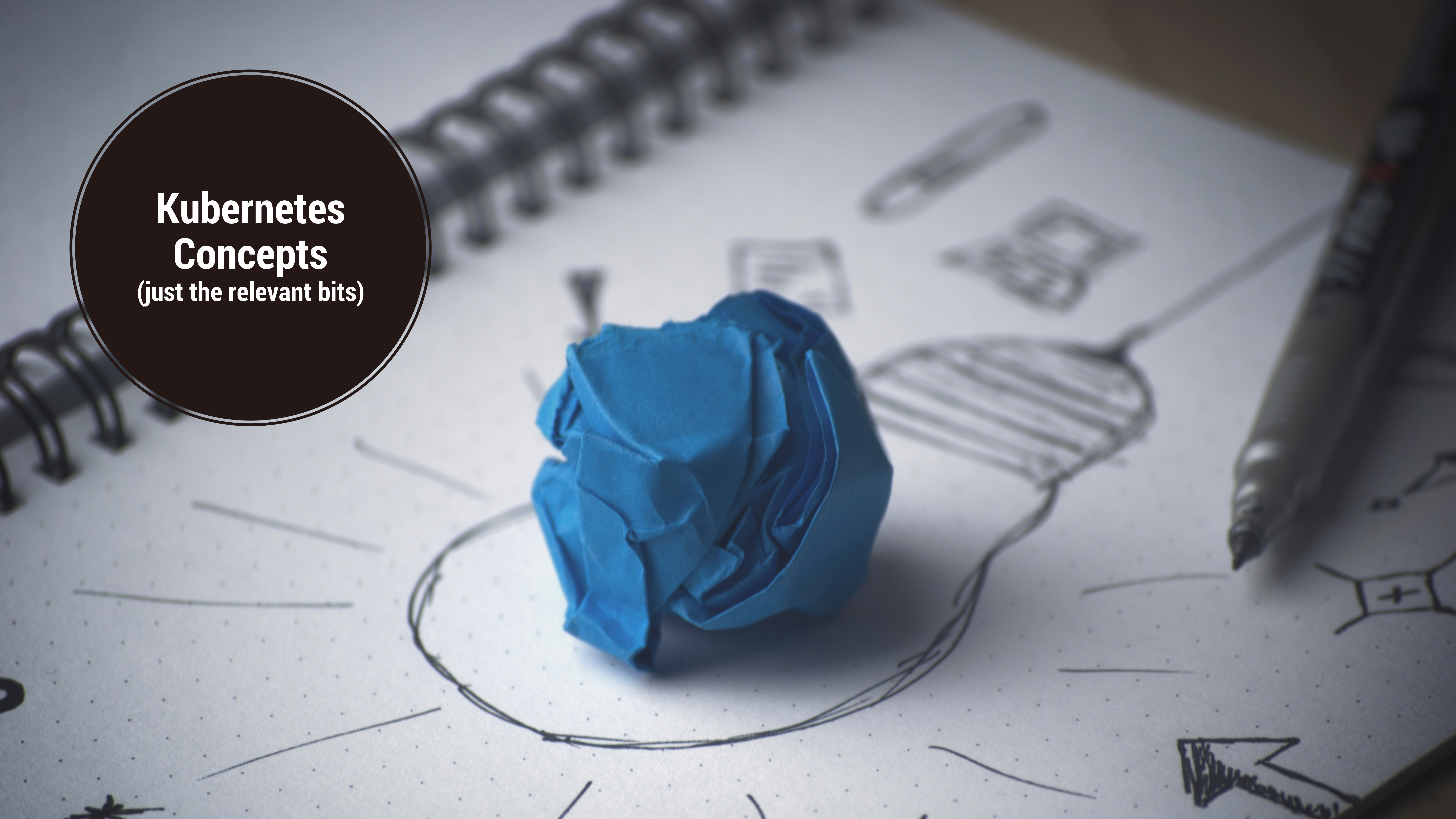
Service Discovery

Familiar IP and DNS-based service discovery and load balancing

The Power of Community!

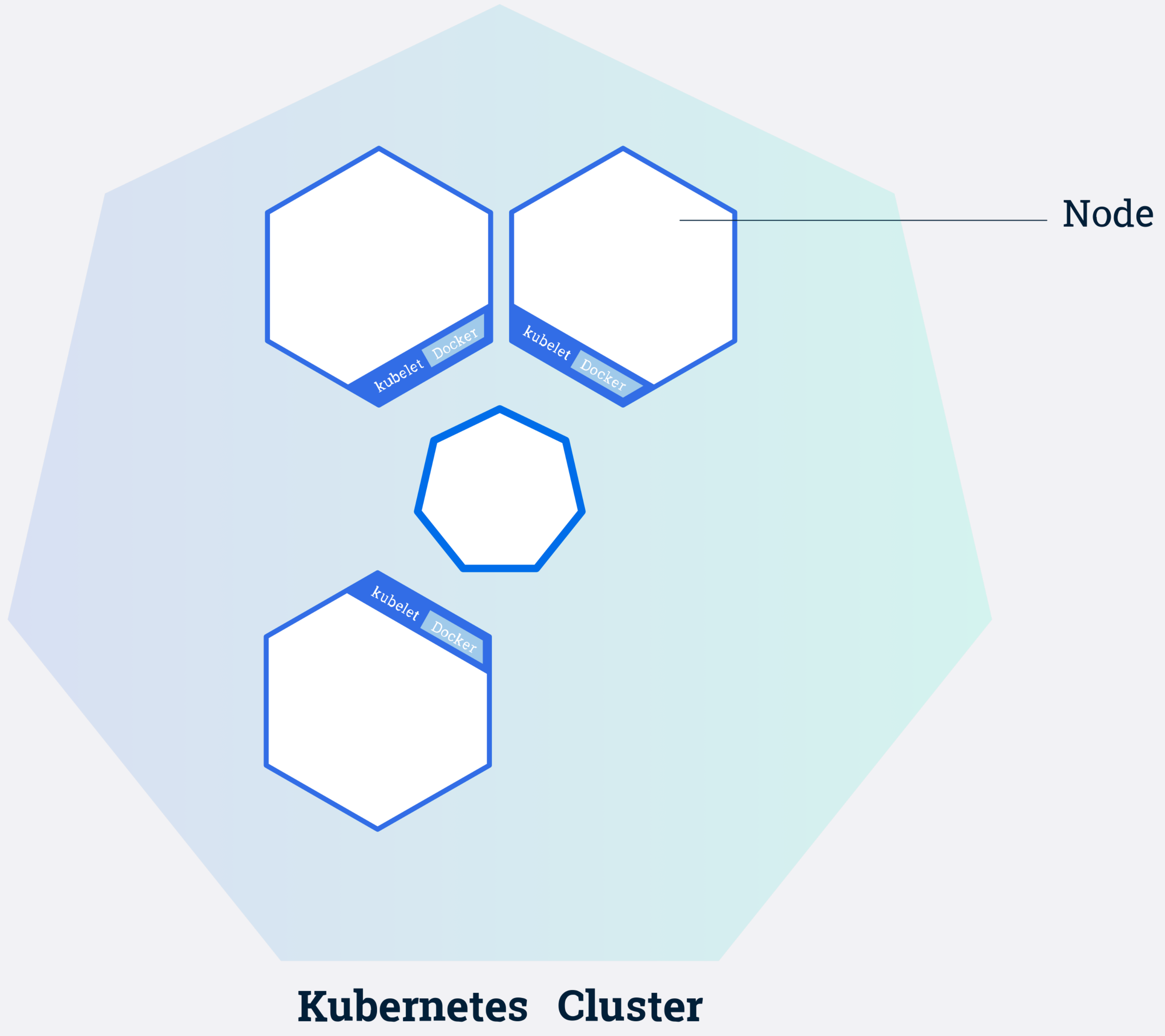

Database and Data Warehouse
Streaming
Source Code Management
Application Definition and Image Build
Continuous Integration / Continuous Delivery (CI/CD)
Platforms
Observability & Analysis
App Definition and Development
Certified Kubernetes - Distribution
Monitoring
Scheduling & Orchestration
Coordination & Service Discovery
Service Management
Orchestration & Management
Certified Kubernetes - Hosted
Logging
Cloud-Native Storage
Container Runtime
Cloud-Native Network
Runtime
Certified Kubernetes - Installer
Tracing
Host Management / Tooling
Infrastructure Automation
Container Registries
Secure Images
Key Management
Provisioning
PaaS/Container Service
Serverless
Public
Private
Cloud
Kubernetes Certified Service Provider
Kubernetes Training Partner
Special
CNCF Incubating
CNCF Sandbox
CNCF Graduated
l.cncf.io
CLOUD NATIVE Landscape
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

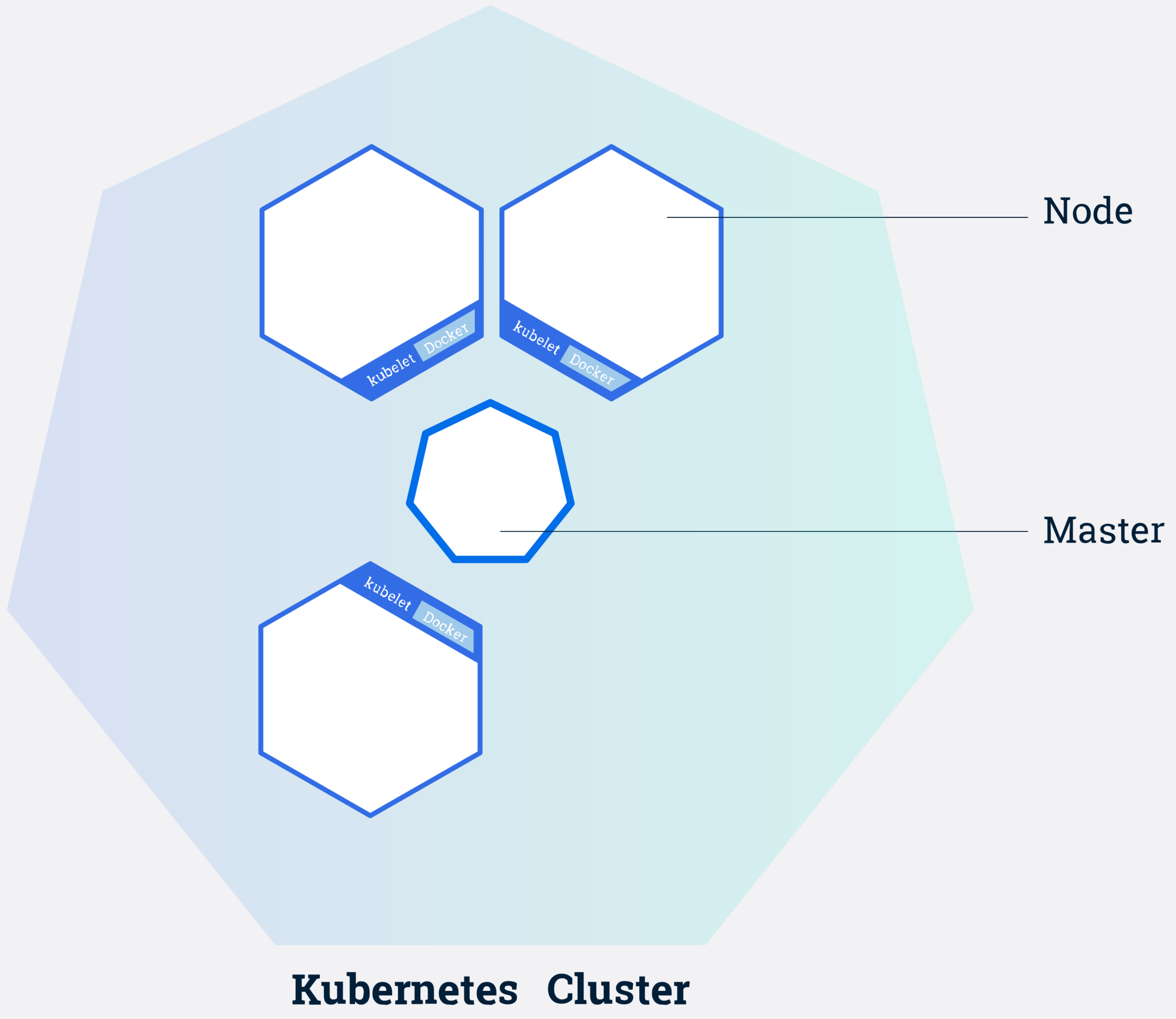


Kubernetes Concepts
(just the relevant bits)

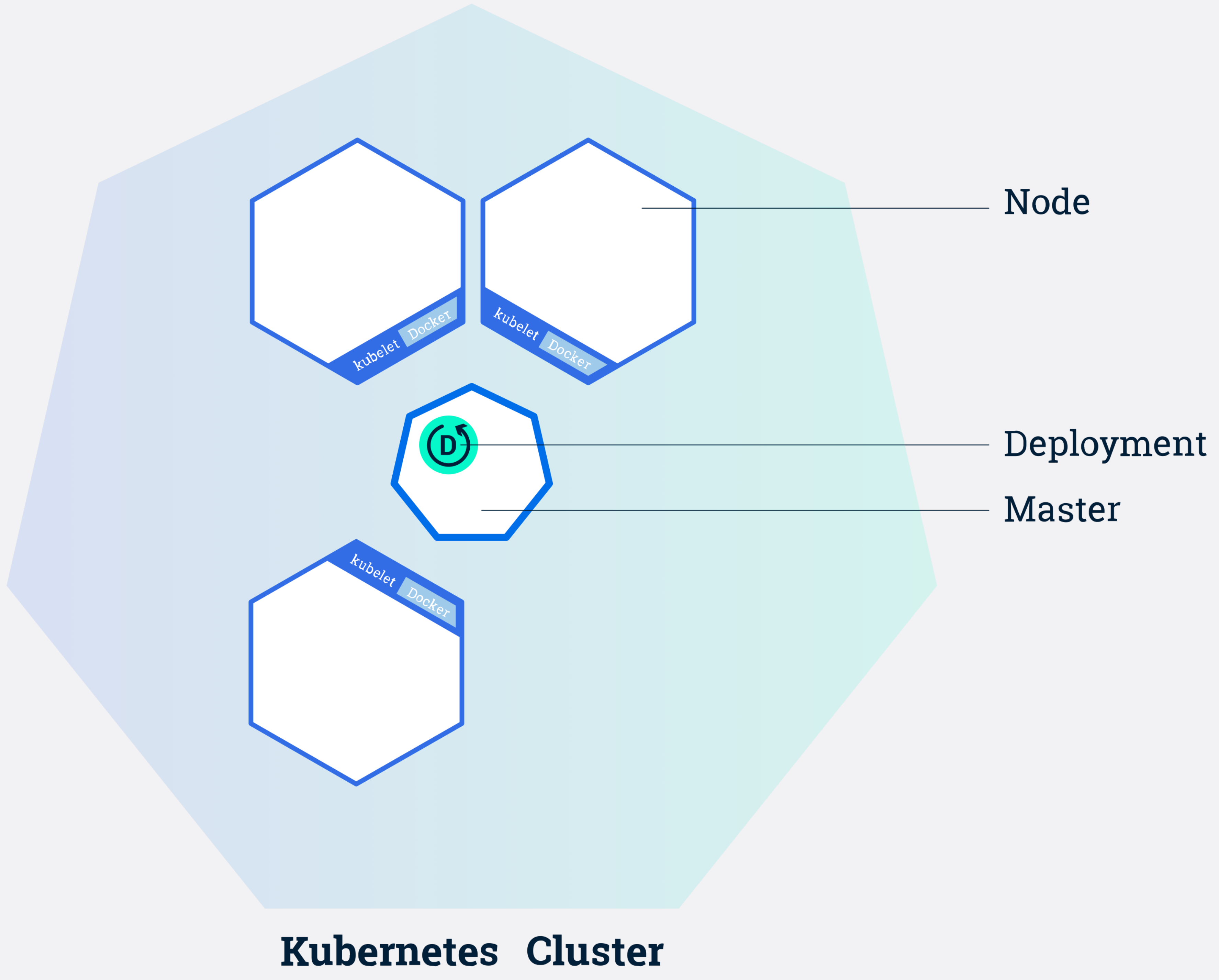
(selected) kubernetes
concepts – cluster + nodes



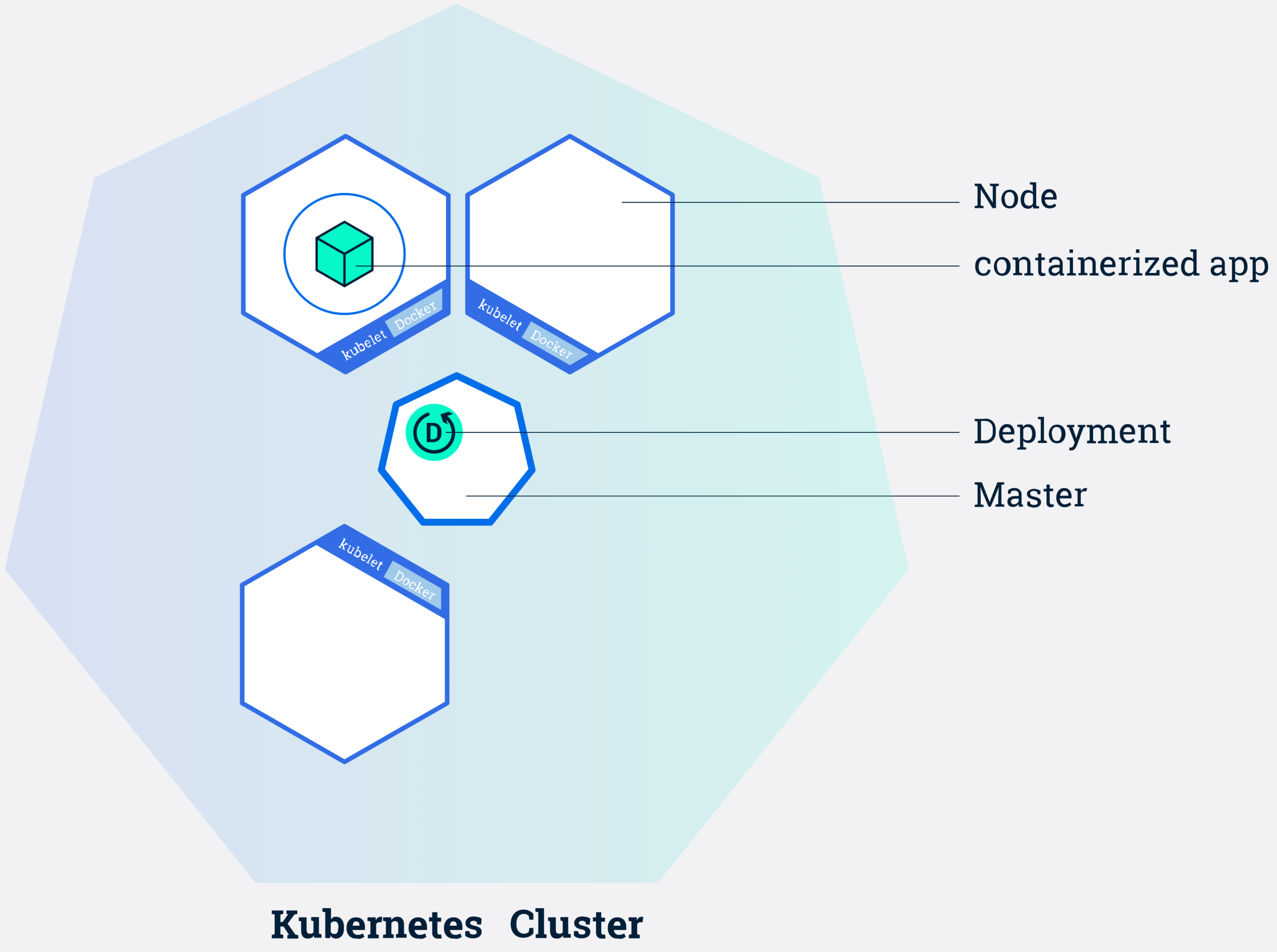
(selected) kubernetes concepts – master node



(selected) kubernetes concepts – deployments



(selected) kubernetes concepts – deployed app



Storage Options for Kubernetes

kubernetes portable storage abstractions
file and block focus

Azure aws Google Cloud

openstack CLOUD SOFTWARE ROOK NetApp GLUSTER Quobyte OpenEBS ceph DELL portworx

and more...

dynamic storage provisioning for persistent storage

01 Self Service

Allow high developer velocity, no admin in the loop

02 Portable

No references to underlying storage provider. Allows application portability

03 On-Demand

Provisioned at time of use. Lifecycle can be tied to the application.

dynamic storage provisioning
persistent volume (pv)

A Persistent Volume (PV) represents provisioned storage in the cluster (e.g., NFS, iSCSI, other block, etc.). A PV's lifecycle is independent of the container/pod that uses it.

dynamic storage provisioning
persistent volume claim (pvc)

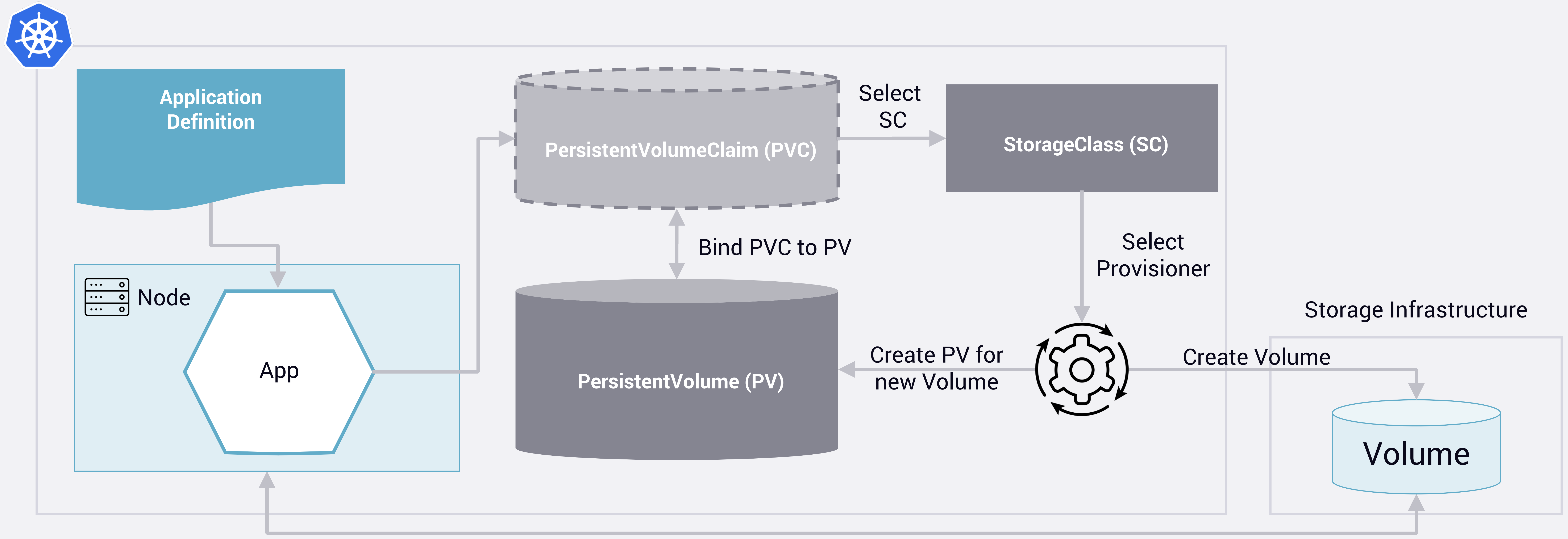
```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: ssd
```

dynamic storage provisioning
persistent volume claim (pvc)

```yaml
kind: Deployment
apiVersion: v1
metadata:
  name: my-app
spec:
  template:
    spec:
      containers:
      - name: app-container
        image: alpine:3.7
        command: ["my-app.sh"]
        args: ["--datadir", "/data/my-app"]
        volumeMounts:
        - name: data-volume
          mountPath: /data
      volumes:
      - name: data-volume
        persistentVolumeClaim:
          claimName: my-claim
```

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: ssd
```

dynamic storage provisioning
putting it all together



Volume mounted on node where Pod is scheduled
(based on Pod -> PVC -> PV mapping)

container storage interface
the path forward

CONTAINER STORAGE INTERFACE

See Managing Disk Volumes in Kubernetes
SDC 2018 talk by Saad and Nikhil for more info!

Out of Tree

Independent Development and Release Cycles, Easier to Maintain

Standard Deployment

Common deployment interface using native Kubernetes primitives

File & Block

Standardized implementation APIs for using file and block

Cross-Orchestrator

Vendor friendly. Kubernetes, Mesos, CloudFoundry,

other operational concerns
scheduling, backup, restore, migration

state is meaningful

- Instances are unique and are not interchangeable
- Access to persistent data is needed across restarts

resiliency is complex

- High-availability depends on instance coordination
- Frequent restarts/pre-empts destabilize service

data is important

- How does backup, recovery, and migrate work? See Kasten's K10 as an example!
- Resource contention concerns

Developer and Operator Support

StatefulSets
support for stateful applications

Stable Identifiers

Stable network identifiers for applications that depend on this

Stable Persistence

Includes persistent mapping across pod restarts and reschedules

Ordered Operations

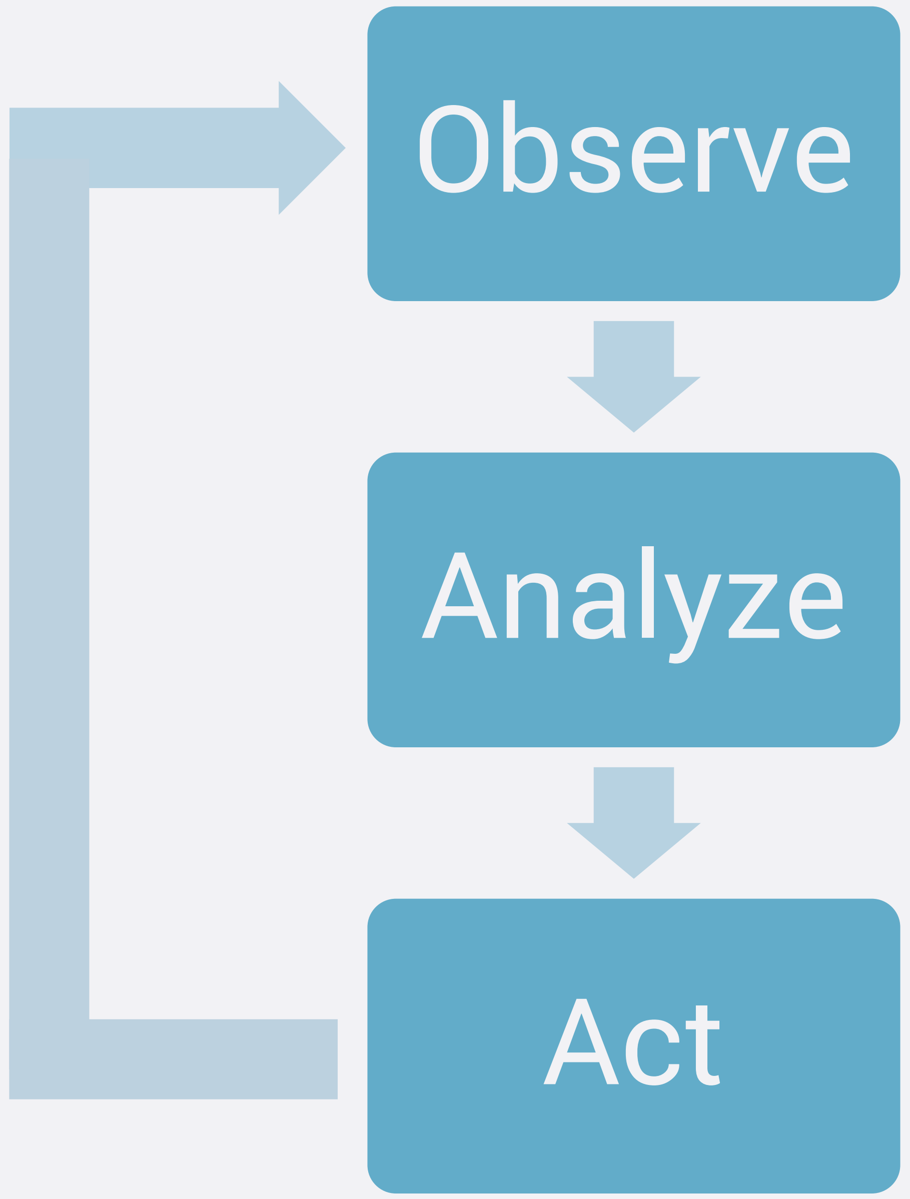
Ordered and graceful deployment, scaling, termination

Update Operations

Rolling updates with restrictions

the operator design pattern
to deploy and manage apps

human ops knowledge → software



- **Support Complex Ops**
  Backups, Recovery, Scaling, Upgrades
- **Active Reconciliation**
  Reconcile desired vs. actual state
- **SDK-based**
  Easy to get started with multiple SDKs. Still a few sharp edges though.
- **Extensible**
  Developer-extensible via CustomResourceDefinitions

kanister: A framework for application-level data management

- Supports complex distributed applications
- Separates mechanism from policy/orchestration
- Allows for unified schedulers and monitoring
- Clean API allows for developer extensions

https://github.com/kanisterio

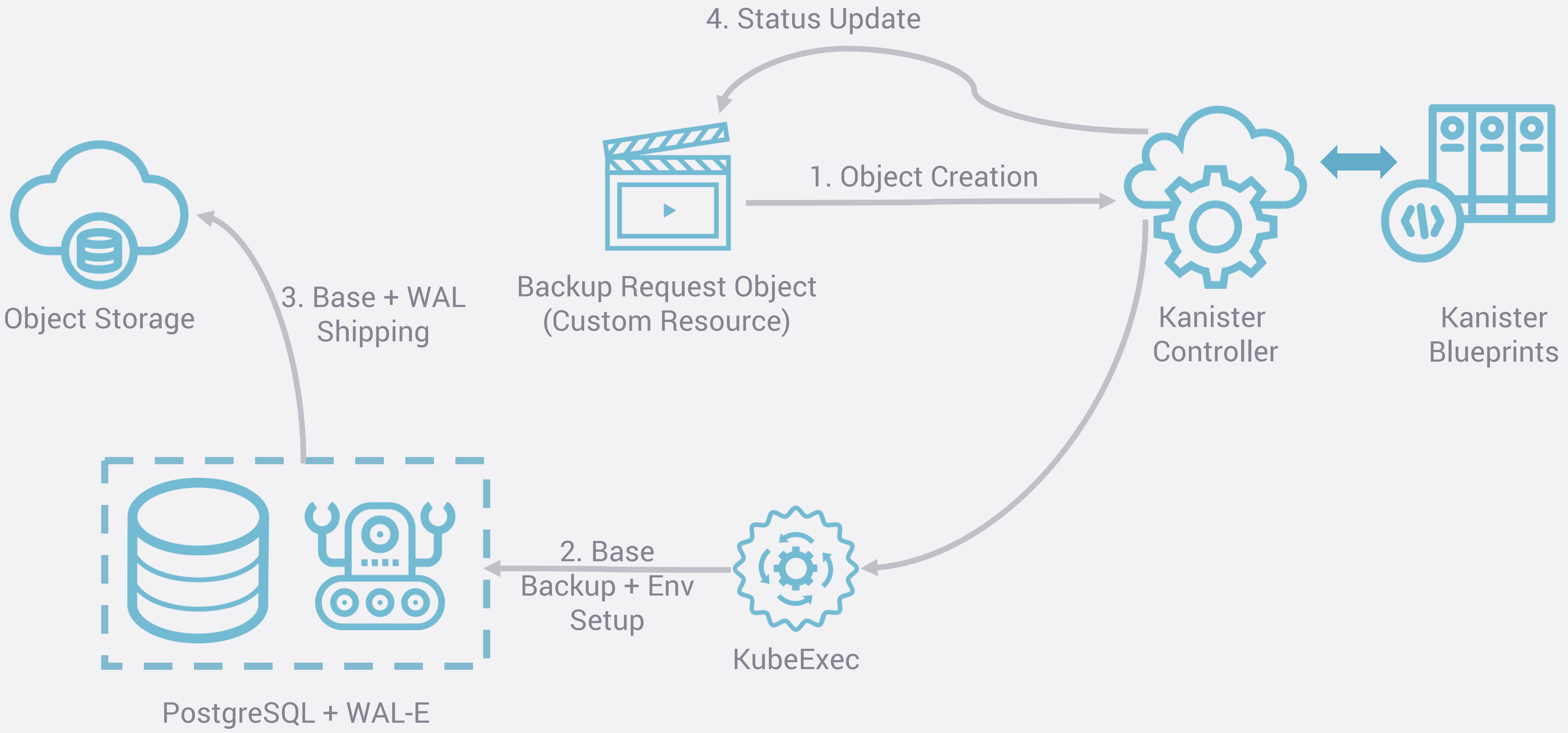
operator
high-level overview

Application

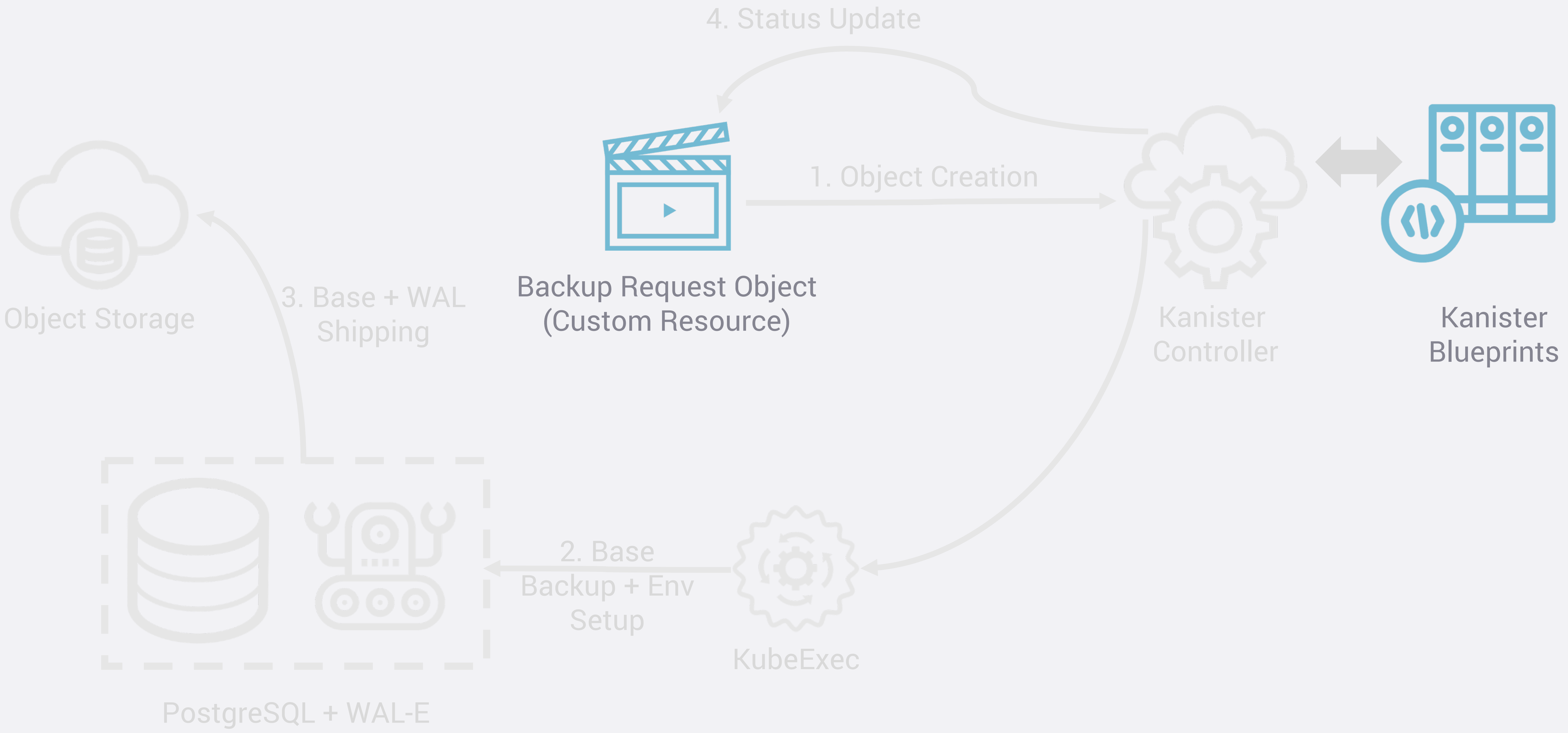
Action Request
(Custom Resource)

Controller

kanister operator example
postgresql backup



kanister operator example
postgresql backup



kanister actionset (abridged)

```yaml
apiVersion: cr.kanister.io/v1alpha1
kind: ActionSet
spec:
  actions:
    - name: backup
      blueprint: postgresql
      object:
        kind: StatefulSet
        name: postgresql-cluster
        namespace: default
      configMaps:
        ...
```

kanister
blueprint (abridged)

```
apiVersion: cr.kanister.io/v1alpha1
kind: Blueprint
actions:
  backup:
    type: StatefulSet
    phases:
    - func: KubeExec
      args:
      - '{{ .StatefulSet.Namespace }}'
      - '{{ index .StatefulSet.Pods 0 }}'
      - postgresql-tools-sidecar
      - bash
      - -c
      - wal-e ...
    - func: ...
  restore:
    ...
```

other awesome
stateful operators

Look at the extensive list at
https://github.com/operator-framework/awesome-operators

PostgreSQL mongoDB cassandra MySQL™

RethinkDB

and more...

packaging your applications
helm: the kubernetes package manager

off-the-shelf stateful “charts”

Multiple community charts available for databases, NoSQL systems, and more.

organize settings

Easy-to-use mechanisms and a single place to codify your application’s configuration options.

supports composability

Enhance or restrict based on your goals. Compose stateful services within your apps.

```
$ helm install stable/postgresql
  --set persistence.size=40Gi
--set persistence.storageClass=ssd
```

```
<your-app>/requirements.yaml
dependencies:
-name: postgresql
```

Upcoming Developments

cloud-native databases
cockroachdb, vitess, yugabyte, and more...

scalable

Auto-scaling built to respond to load and deliver predictable performance

resilient

Fault-tolerance built in to support transparent self-healing infra

self-managing

Reduces ops overhead by automatically handling system management tasks

local persistent volumes (beta)
local disks “done right”

Leverage Local Disks

For systems (Ceph, Cassandra, etc.) that work best on local storage

Common Primitives

Uses well-know PersistentVolume, PersistentVolumeClaim, StorageClass

Smarter Scheduling

Smarter pod scheduling and volume binding compared to hostPath

Expose as Block

Not just file system access anymore

kubernetes and state
wrapping up

Stateful is Ready for Production!

01 Platform Support
Equivalent features and concepts that made stateless successful

02 Storage Vendor Choices
Large number of storage provider choices, CSI, Portability Abstractions

03 Relational / NoSQL Systems
Support from traditional relational and NoSQL systems. First-class operators. Cloud-Native DBs.

04 Increased Production Usage
50%+ users using stateful applications - SIG-APPs Survey, Apr'18

thank you