
Kubernetes I Deep Dive

Agenda

Red Hat Developer | DevNation

Part I

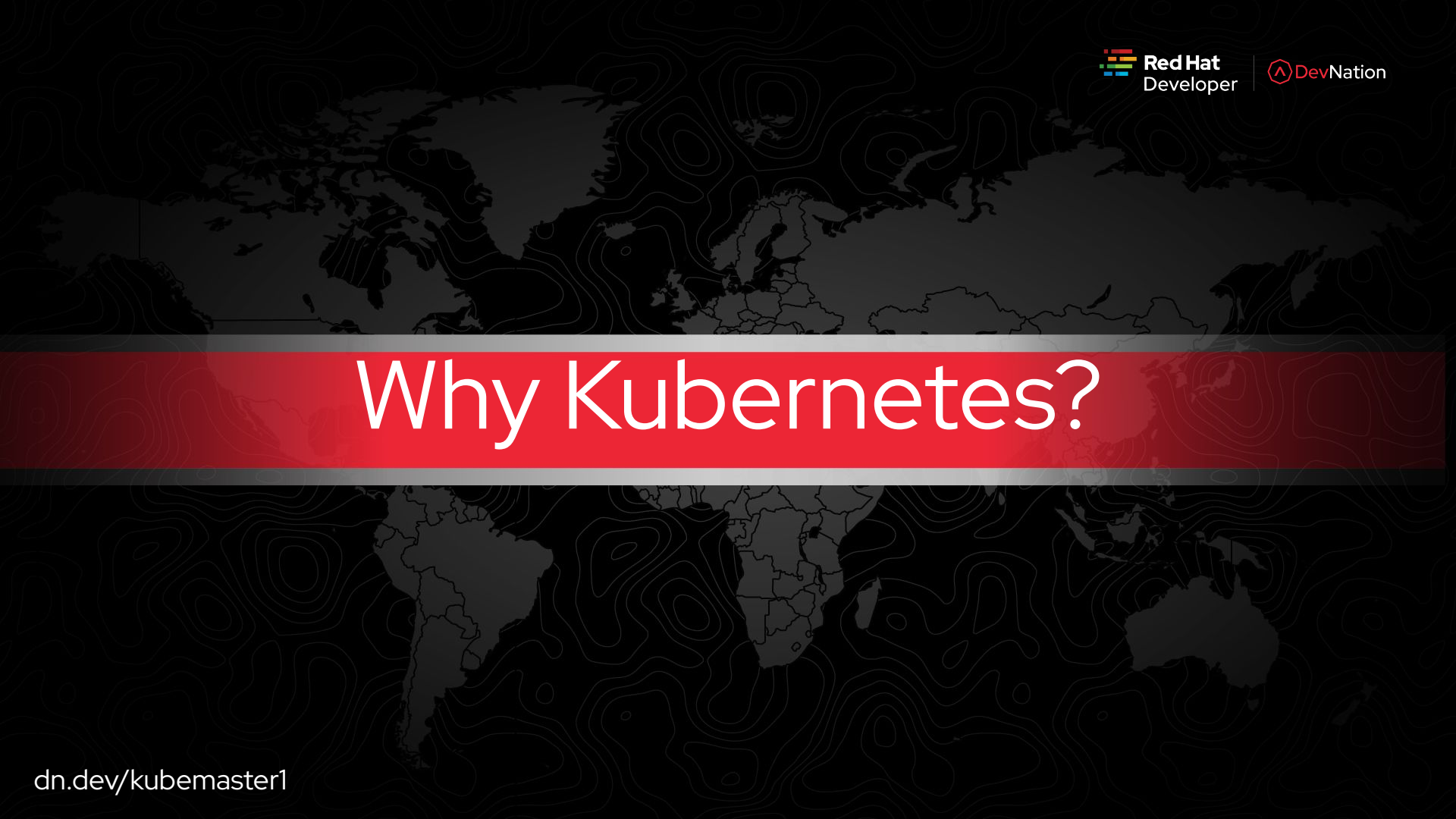
- Why Kubernetes
- What is Kubernetes
- Installation
- kubectl
- Pod
- ReplicaSet
- Deployment

Part II

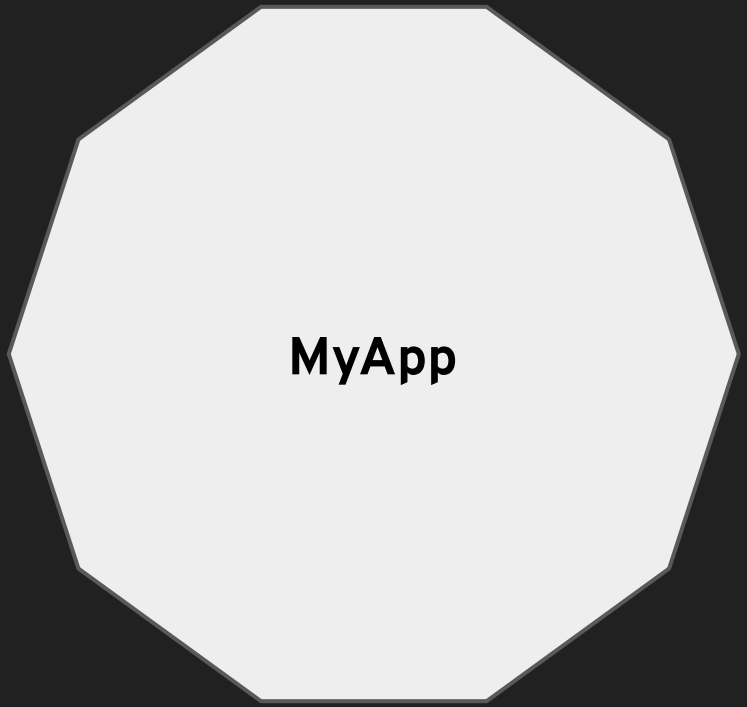
- Building Images
- Resource Limits
- Rolling Updates
- Live & Ready
- Env & ConfigMap

Part III

- Volumes
- Secrets
- Operators
- Taints & Affinity
- Jobs & CronJobs
- DaemonSet
- StatefulSets

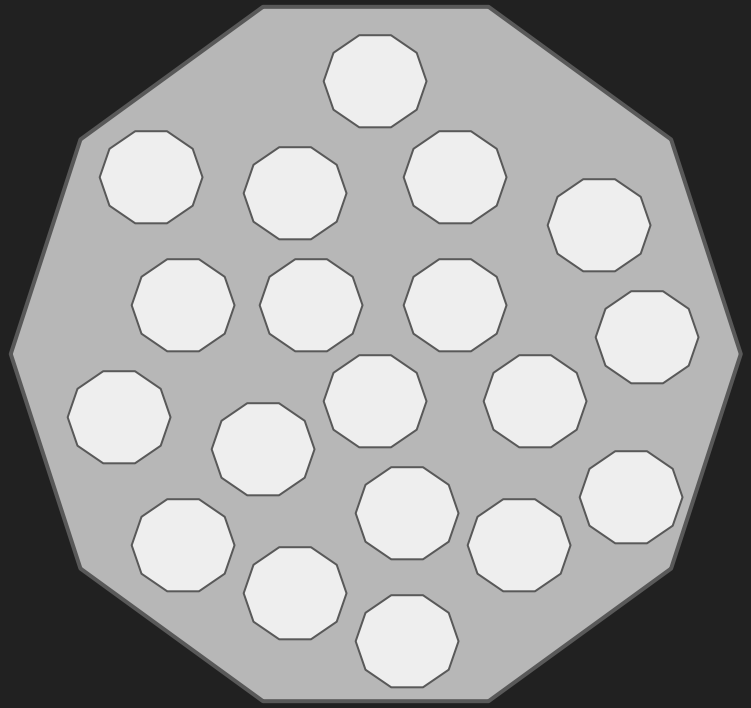

Red Hat
Developer
DevNation
Why Kubernetes?
dn.dev/kubemaster1

Monolith



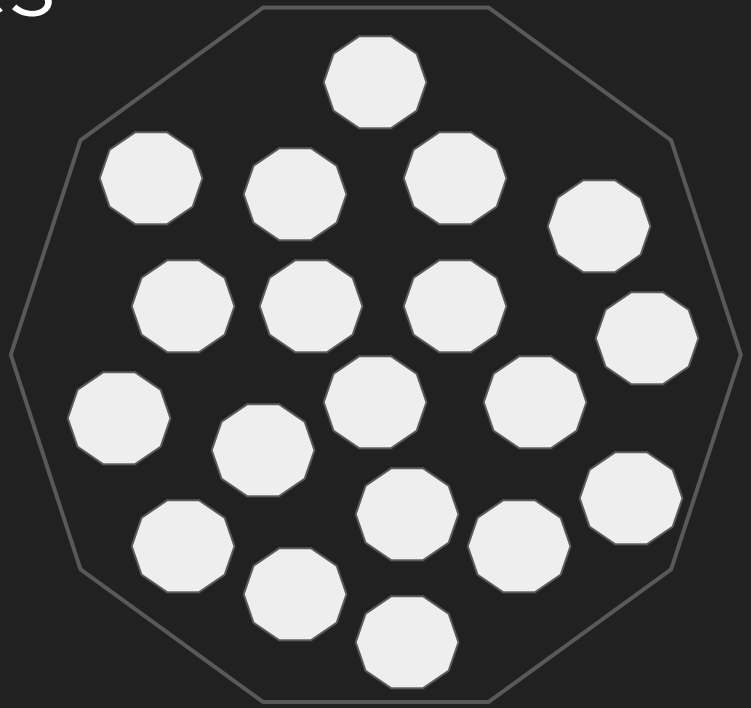
Red Hat Developer | DevNation

Application Modules



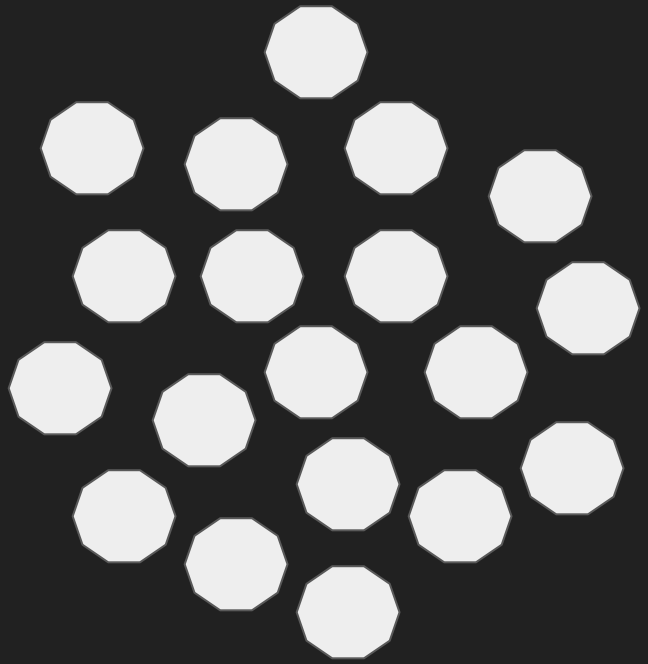
Red Hat Developer | DevNation

Microservices

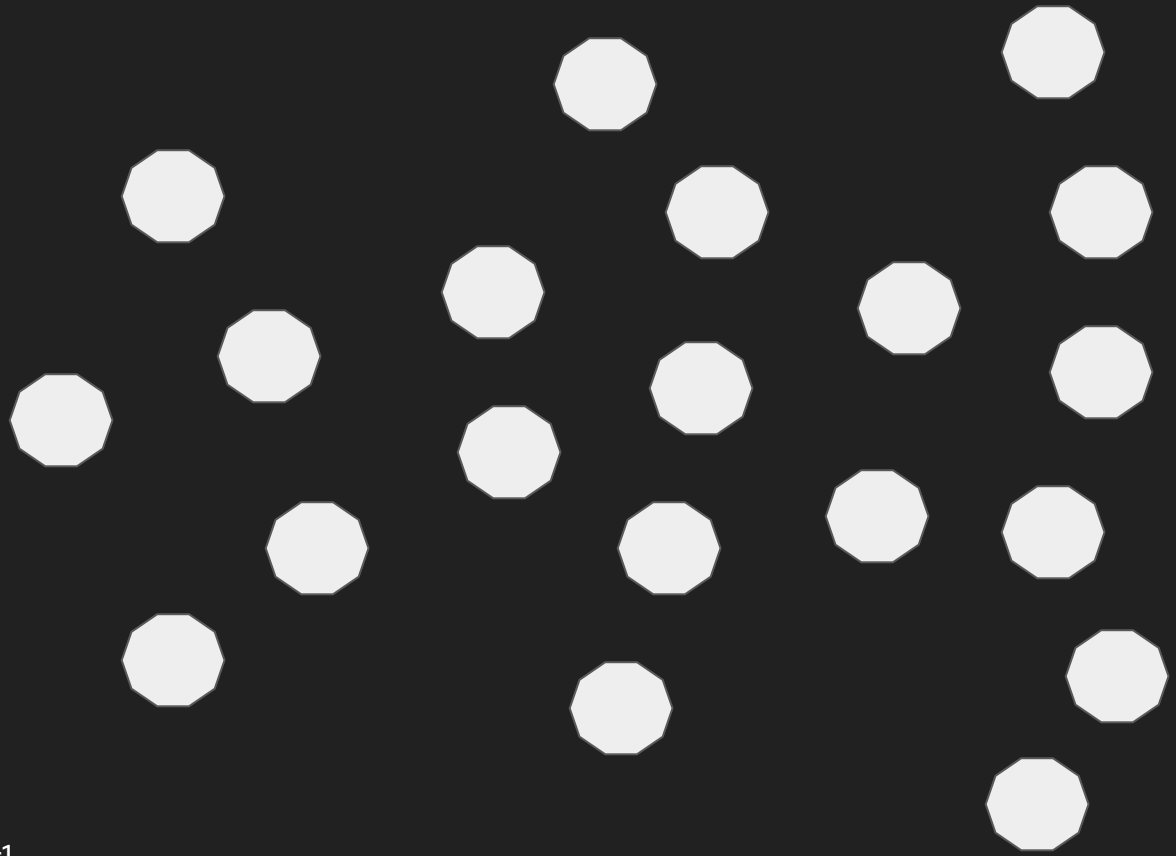


dn.dev/kubemaster1

Containers



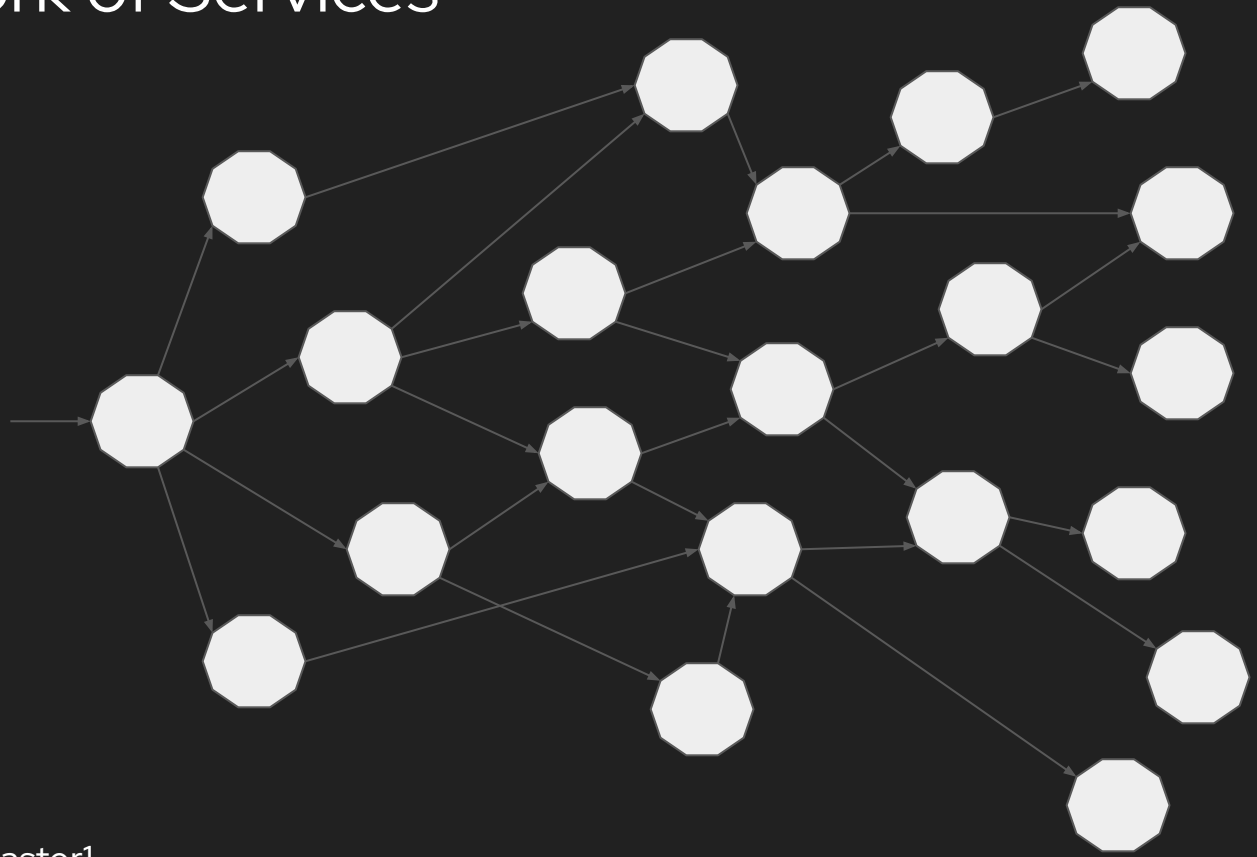
Red Hat Developer | DevNation



dn.dev/kubemaster1

Red Hat Developer | DevNation

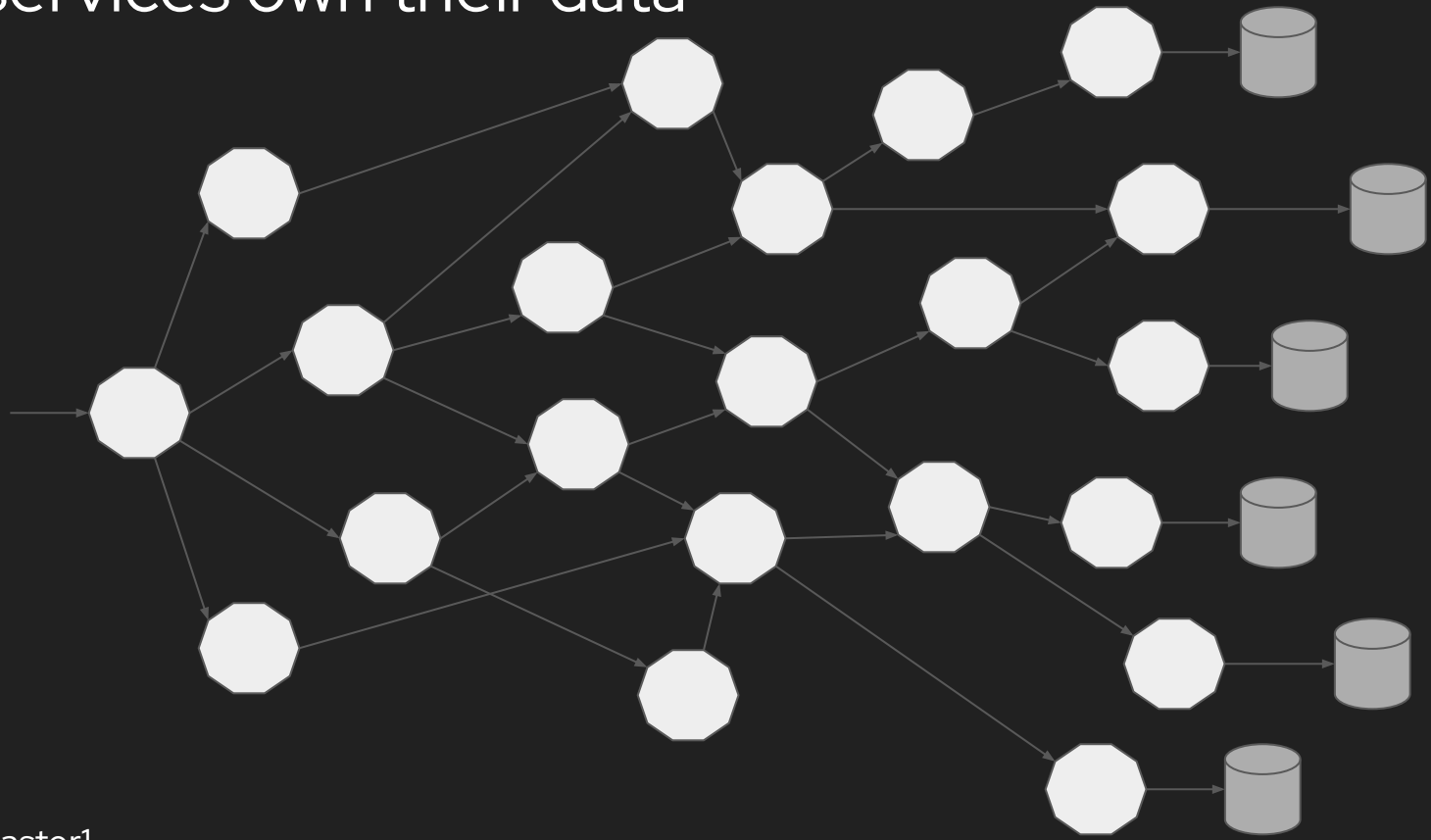
Network of Services



dn.dev/kubemaster1

Red Hat Developer | DevNation

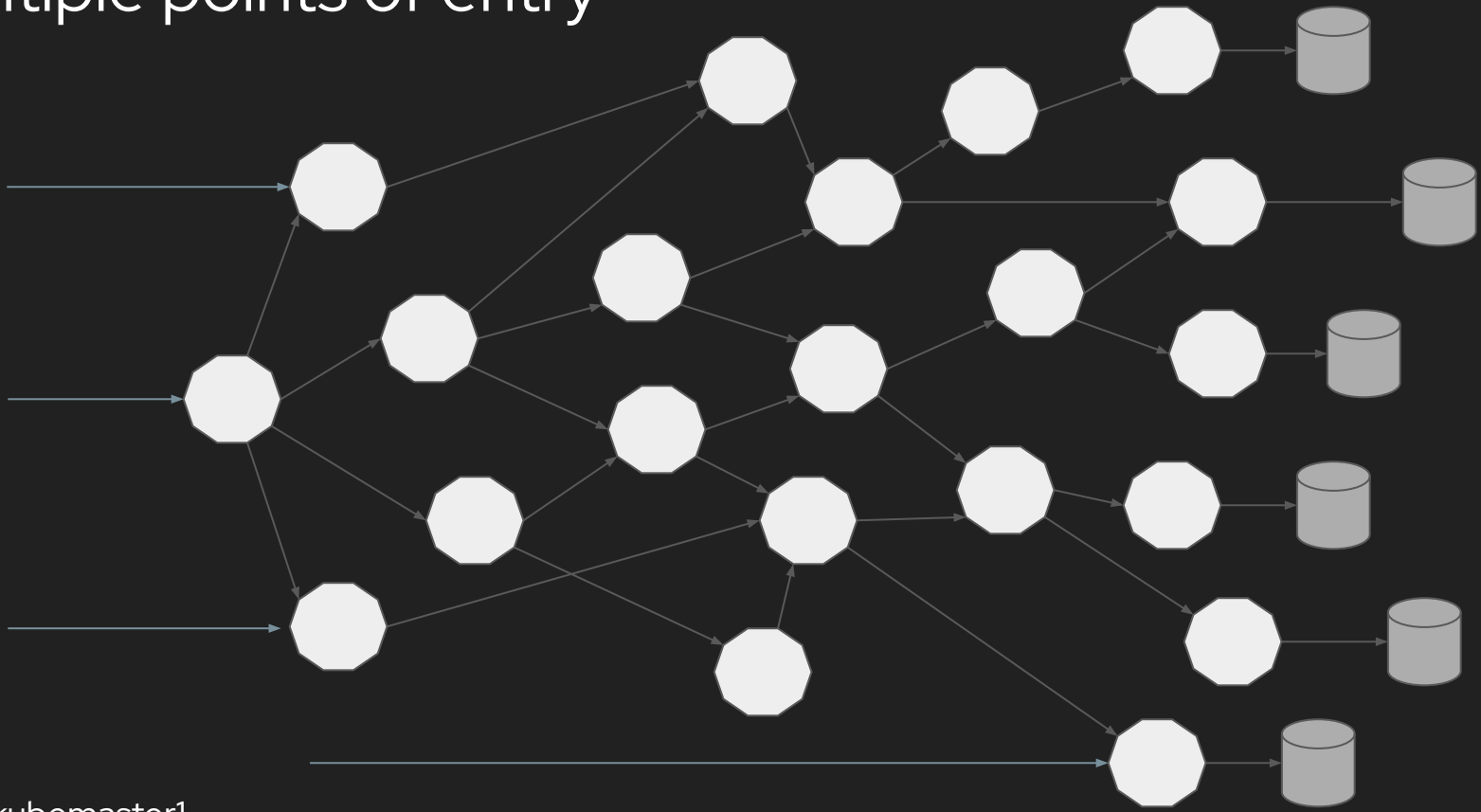
Microservices own their data



dn.dev/kubemaster1

Red Hat Developer | DevNation

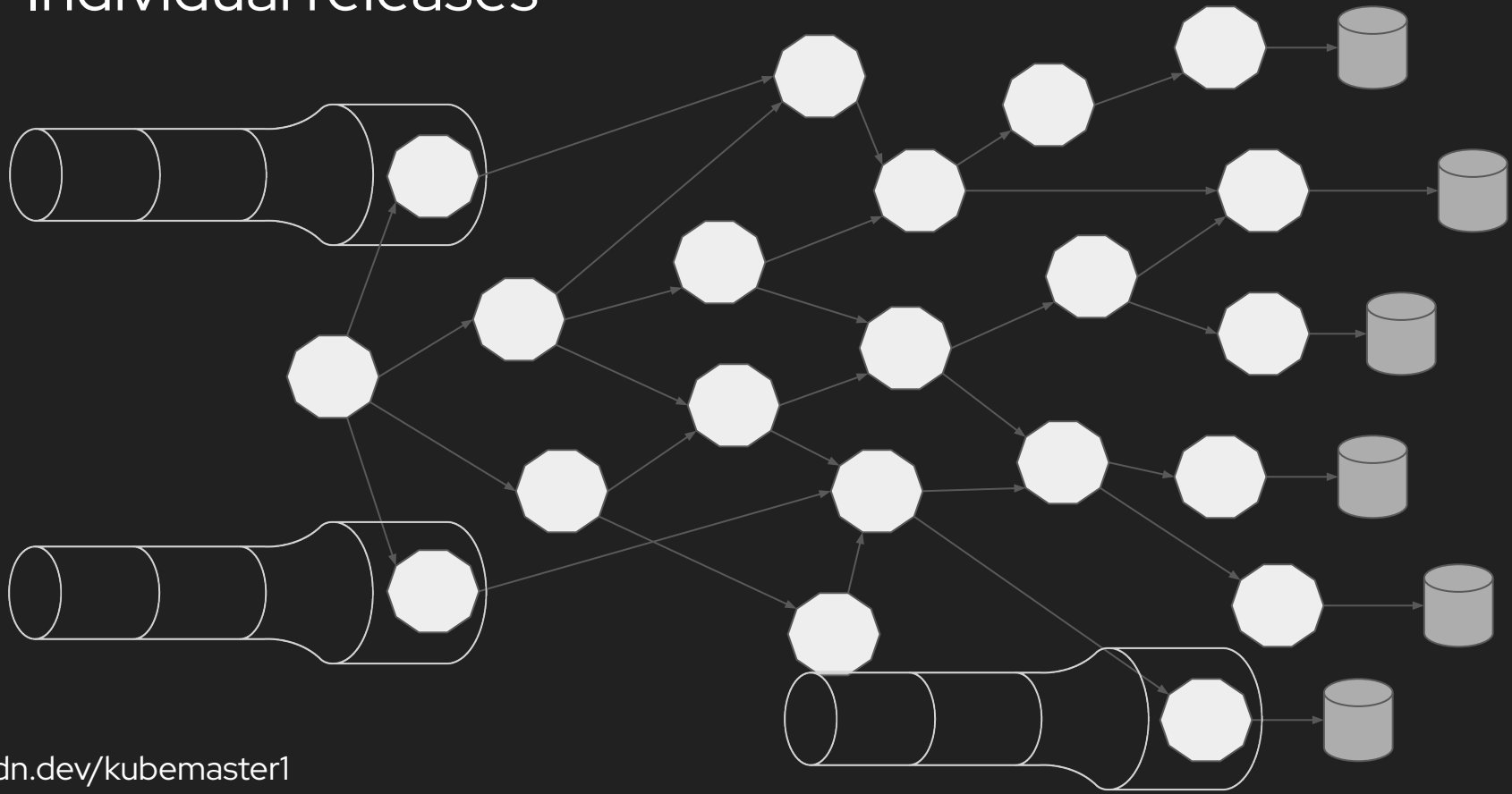
Multiple points of entry



dn.dev/kubemaster1

Red Hat Developer | DevNation

Individual releases

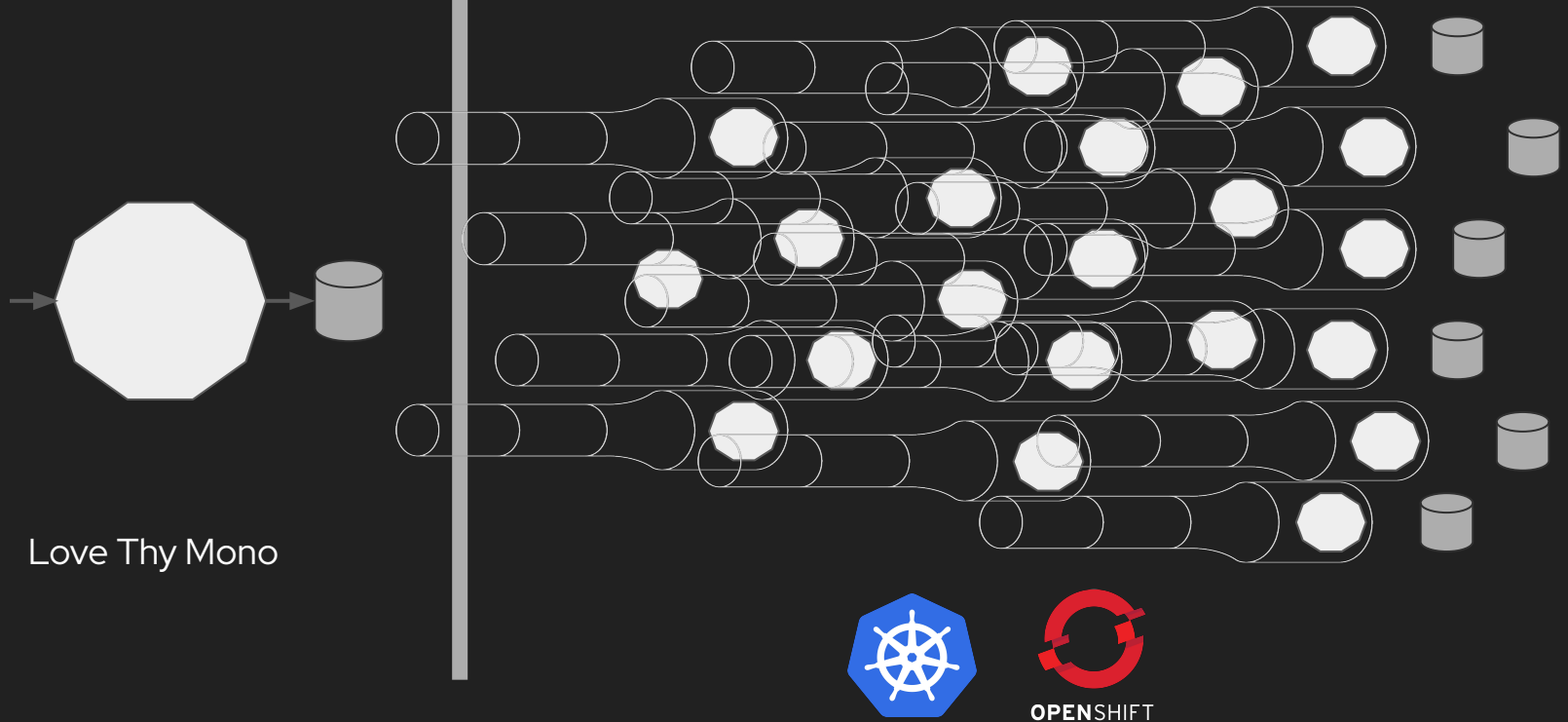


dn.dev/kubemaster1

Red Hat Developer | DevNation

Every 3 months

Every 3 weeks/days/hours



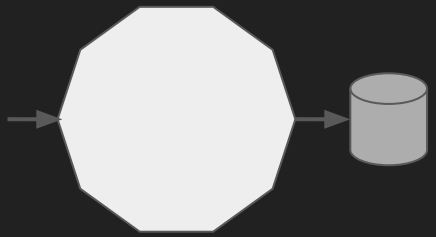
Love Thy Mono

OPENSHIFT

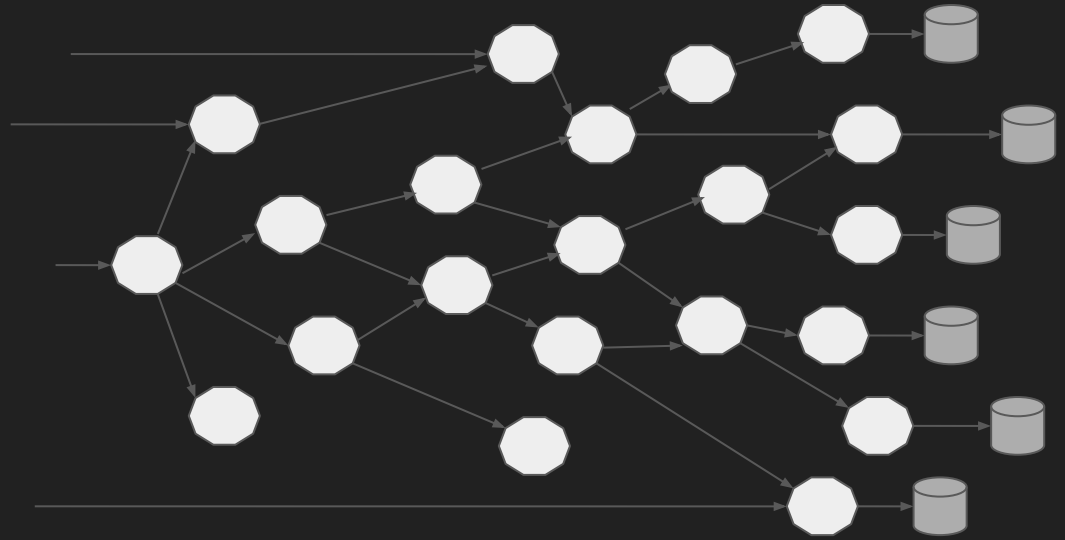
dn.dev/kubemaster1

Red Hat Developer | DevNation

Old School



New School

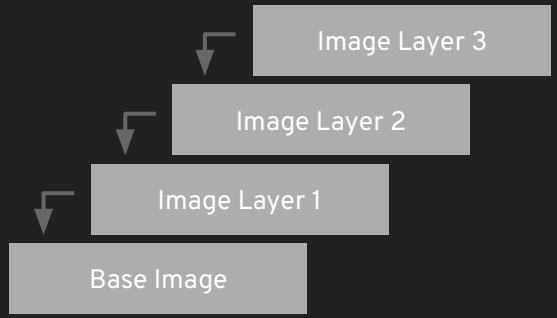


OPENSHIFT

dn.dev/kubemaster1


Red Hat
Developer
DevNation
Containers
dn.dev/kubemaster1

Container Image Layers



Container Image Layers

Example Container Image

Dockerfile: create your Container

Red Hat Developer | DevNation

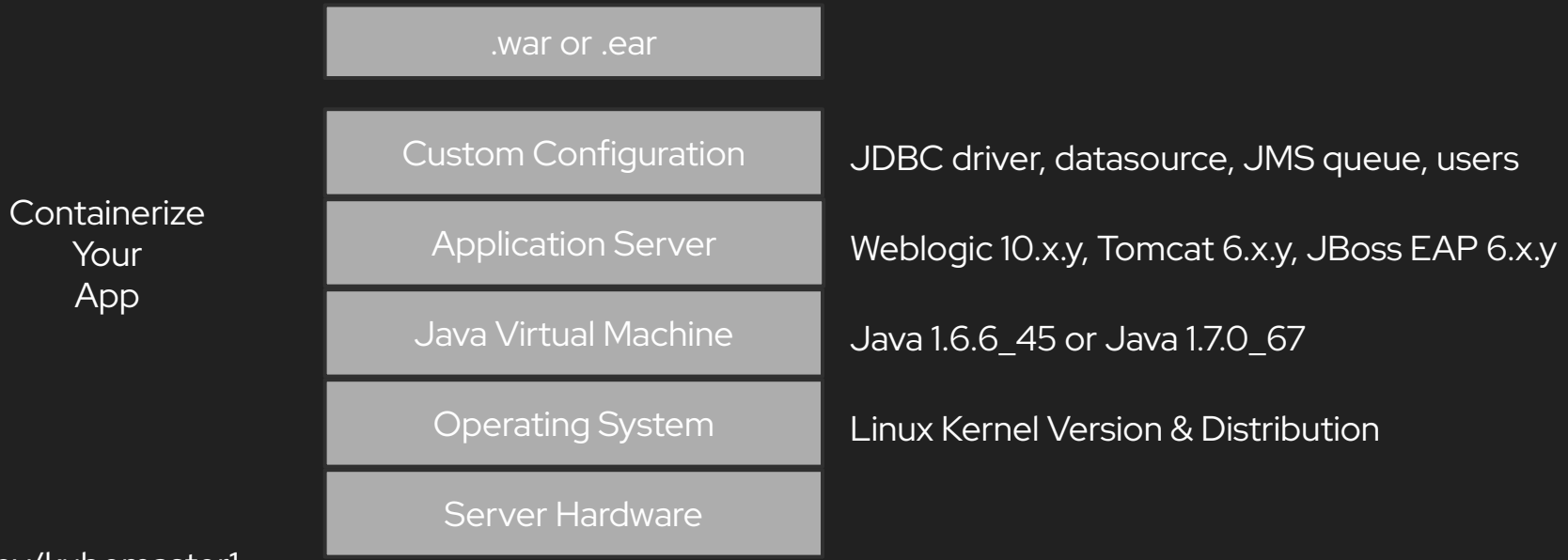
1. Inherit from a base image
2. Parameters as environment variables
3. Install dependencies (tooling from base image)
4. Add your app as a new Layer
5. Expose the port your app will use
6. Run the app

```
FROM registry.access.redhat.com/ubi8/ubi
ENV foo=text
RUN dnf install -y java-11-openjdk
ADD my-app.jar /home/my-app.jar
EXPOSE 8080
CMD java -jar /home/my-app.jar
```

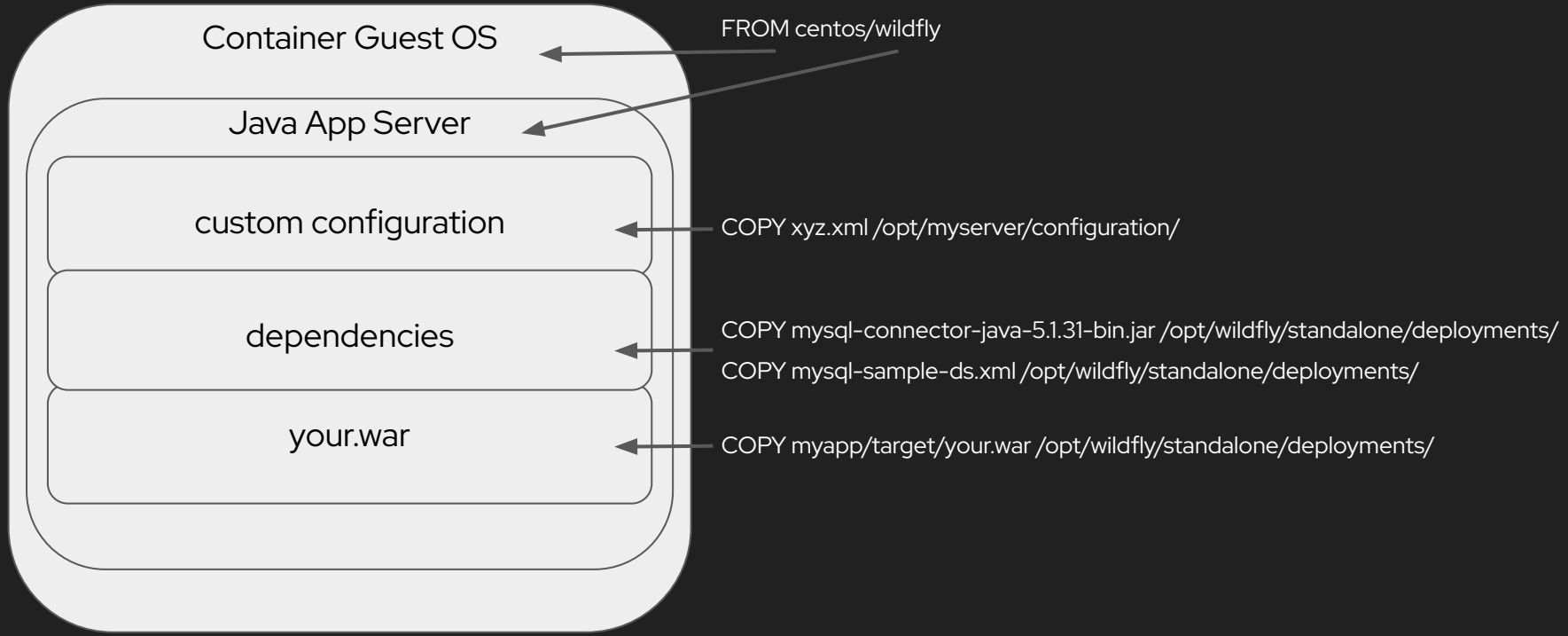
A Challenge

Red Hat Developer | DevNation

Have you ever had "/" vs "\" break your app? Or perhaps needed a unique version of a JDBC driver? Or had a datasource with a slightly misspelled JNDI name? Or received a patch for the JVM or app server that broke your code?



Anatomy of a Container Image



dn.dev/kubemaster1

DevOps Challenges for Multiple Containers

Red Hat Developer | DevNation

- How to scale?
- How to avoid port conflicts?
- How to manage them on multiple hosts?
- What happens if a host has trouble?
- How to keep them running?
- How to update them?
- Rebuild Container Images?


Red Hat
Developer
DevNation
Kubernetes
dn.dev/kubemaster1

Meet Kubernetes

Red Hat Developer | DevNation

- Greek for “Helmsman,” also the root of the word “Governor” (from latin: gubernator)
- Container orchestrator
- Supports multiple cloud and bare-metal environments
- Open source, written in Go
- Manage applications, not machines

OPENSHIFT

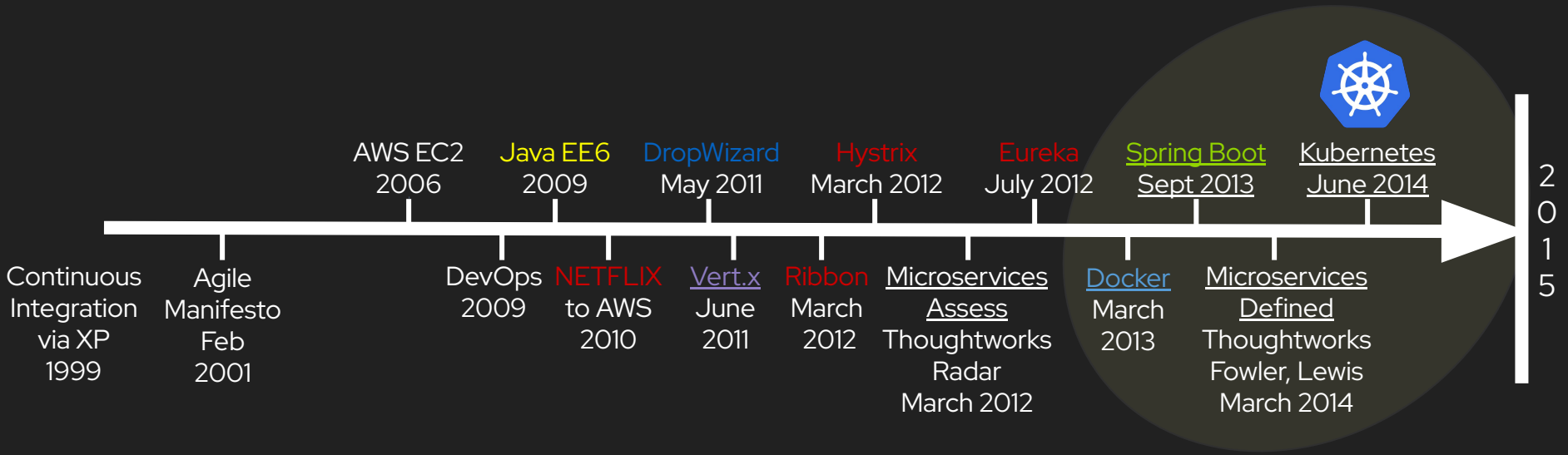
Red Hat Developer | DevNation

Learn more

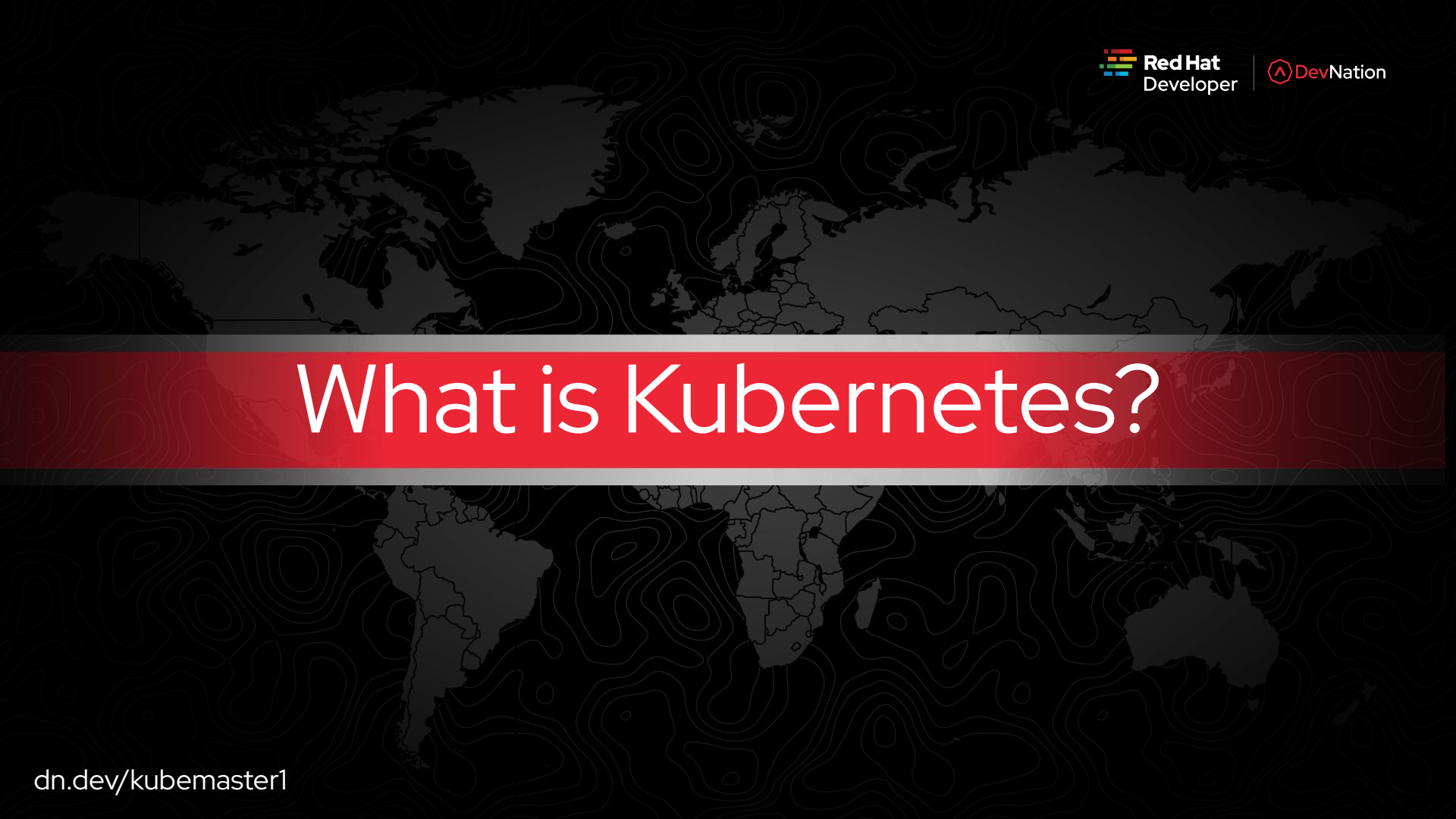
dn.dev/kubemaster1

History of Microservices

Red Hat Developer | DevNation



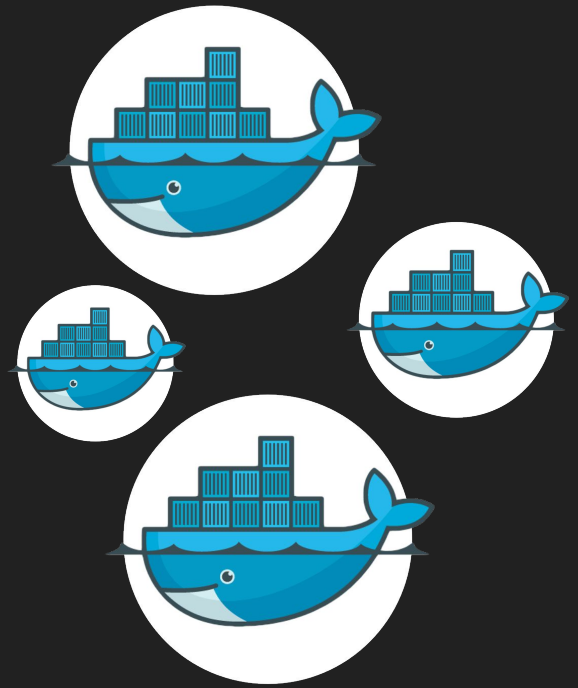
dn.dev/kubemaster1


Red Hat
Developer
DevNation
What is Kubernetes?
dn.dev/kubemaster1

Pods

A group of whales is commonly referred to as a pod and a pod usually consists a group of whales that have bonded together either because of biological reasons or through friendships developed between two or more whales.

In many cases a typical whale pod consists of anywhere from 2 to 30 whales or more.*



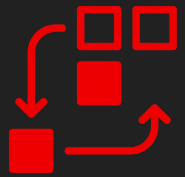
*http://www.whalefacts.org/what-is-a-group-of-whales-called/

Red Hat Developer | DevNation

Pod

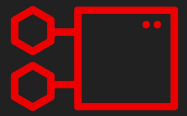
- ✓ 1+ containers
- ✓ Shared IP
- ✓ Shared storage (ephemeral)
- ✓ Shared resources
- ✓ Shared life cycle

ReplicaSet/ Deployment



- ✓ The desired state - replicas, pod template: health checks, resources, image

Service



- ✓ Grouping of pods (acting as one) has stable virtual IP and DNS name

Persistent Volume

- ✓ Network available storage
- ✓ PVs and PVCs

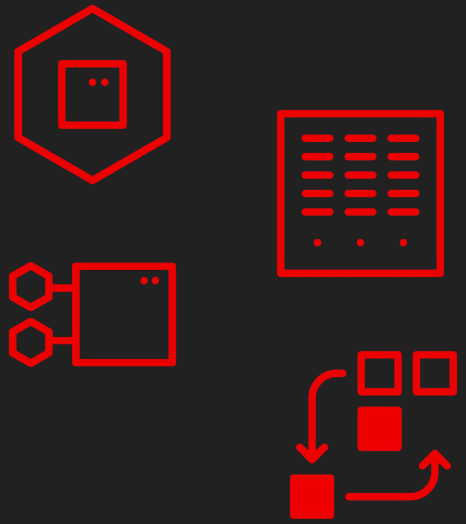
Label

- ✓ Key/value pairs associated with Kubernetes objects (env=production)

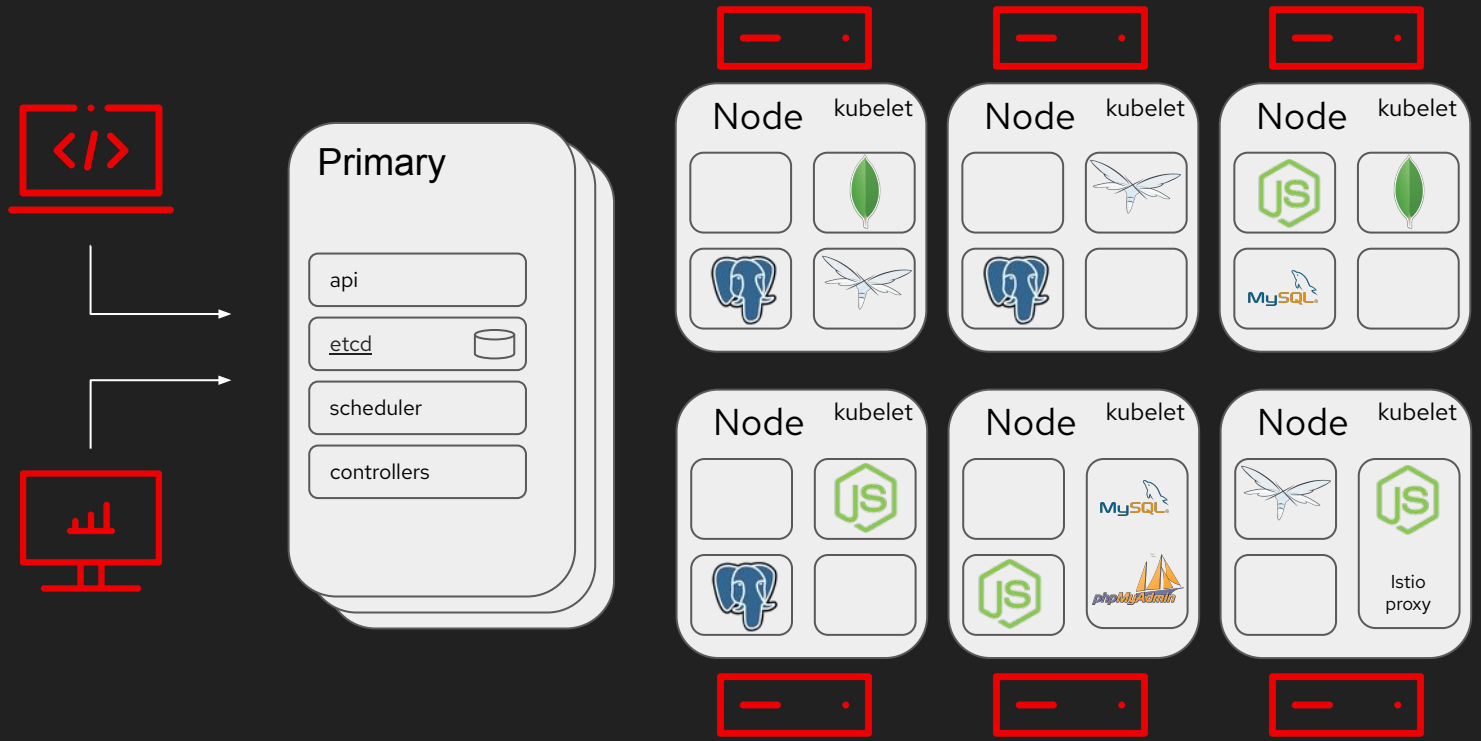
dn.dev/kubemaster1

Namespaces

- ✓ A way to divide cluster resources between multiple users
- ✓ Support attachment of authorization and usage policies

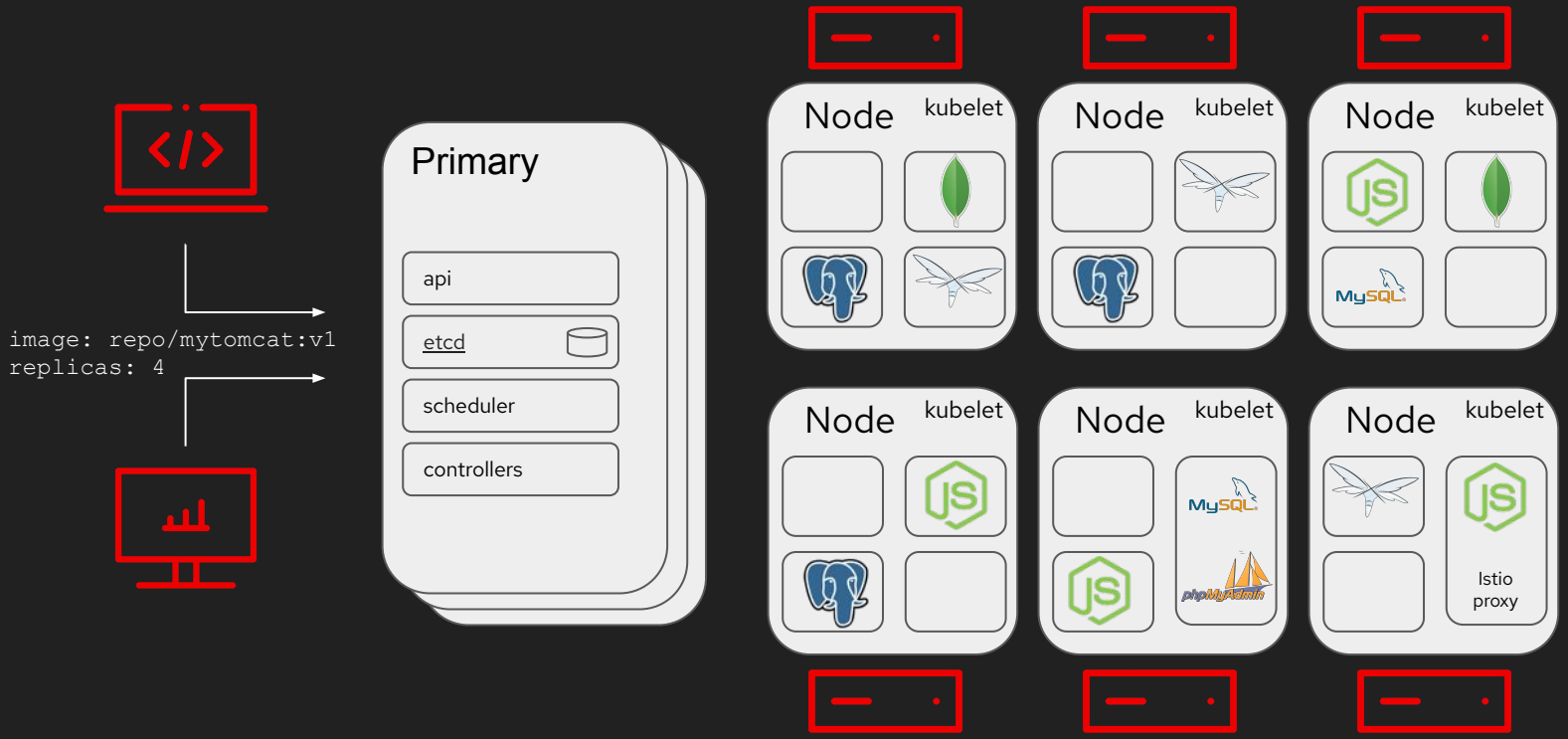


Kubernetes Cluster - Nodes



dn.dev/kubemaster1

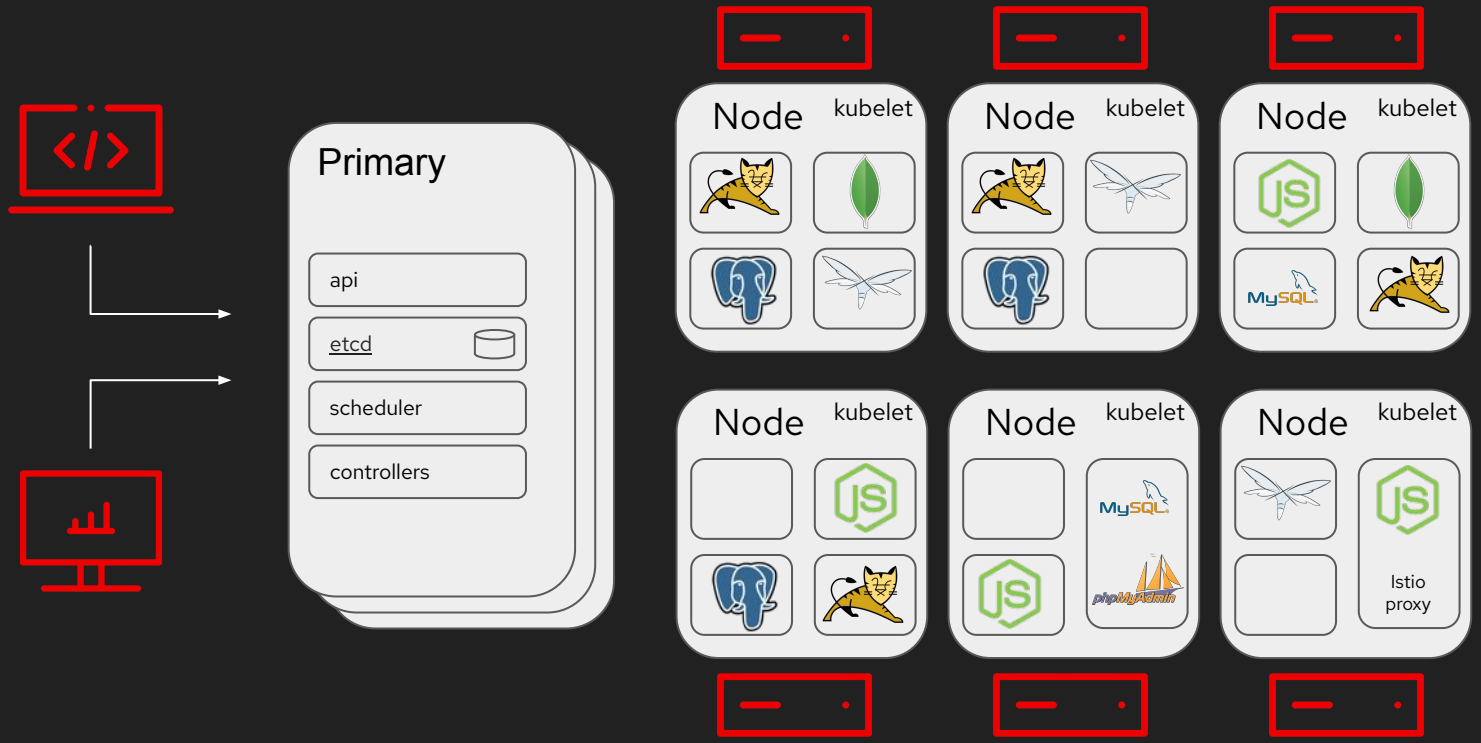
Kubernetes Cluster - Declarative API



dn.dev/kubemaster1

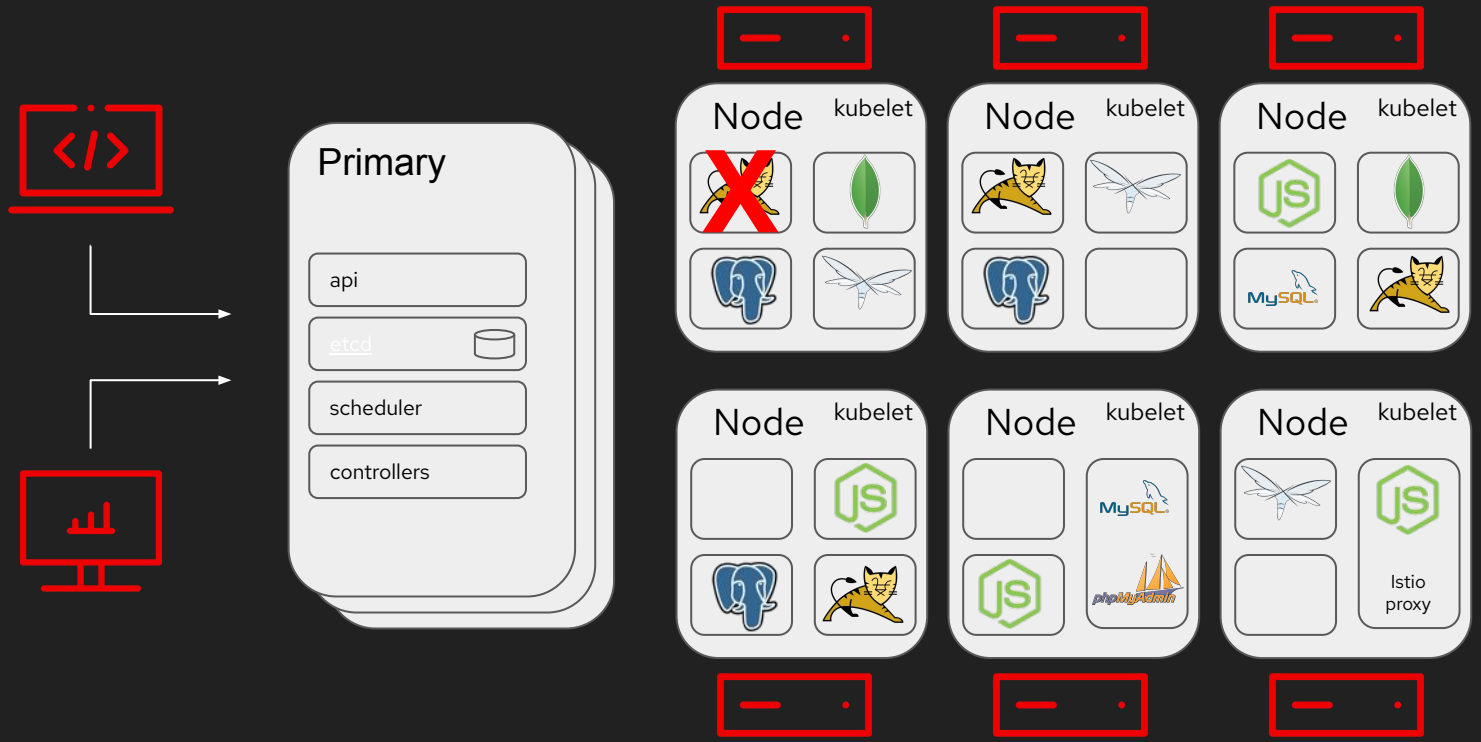
Red Hat Developer | DevNation

Kubernetes Cluster - 4 Tomcat Instances



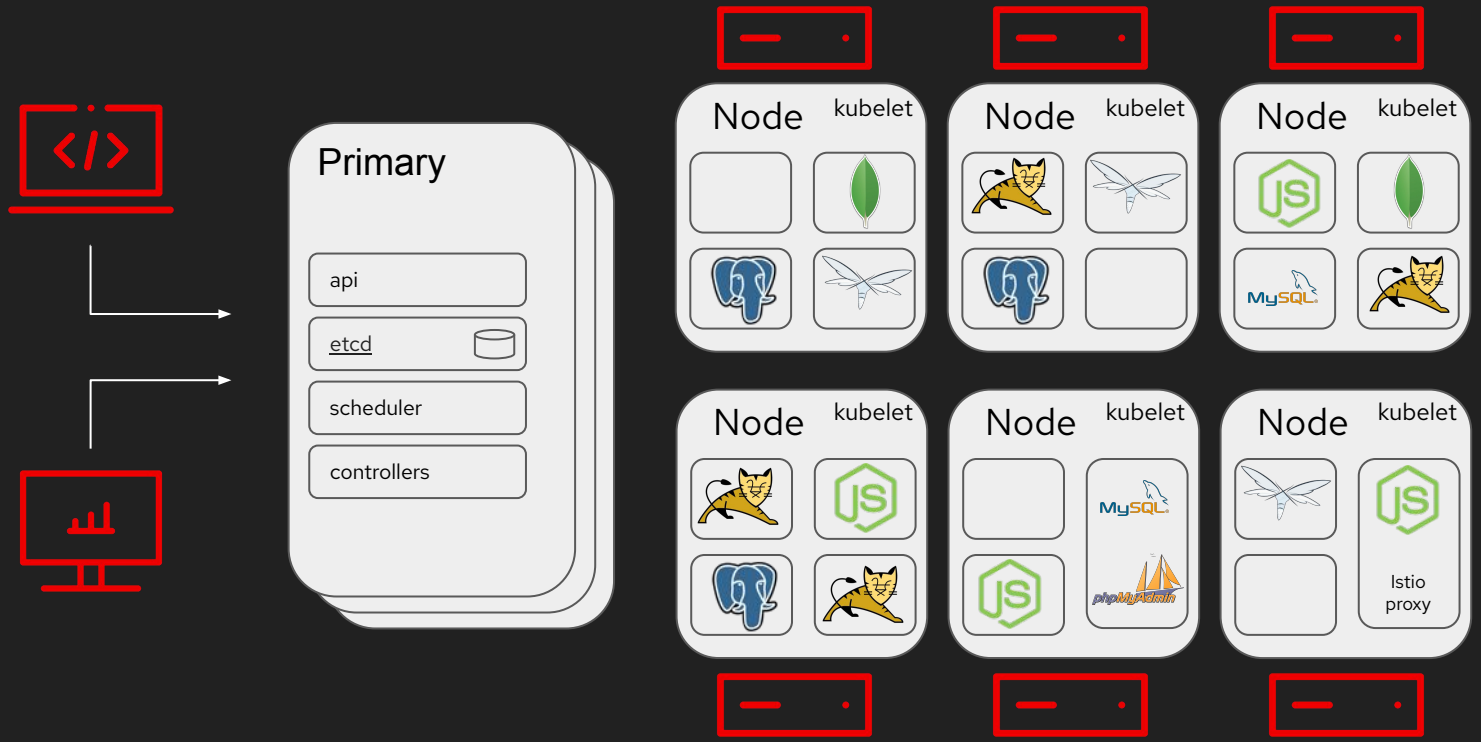
dn.dev/kubemaster1

Kubernetes Cluster - Pod Failure



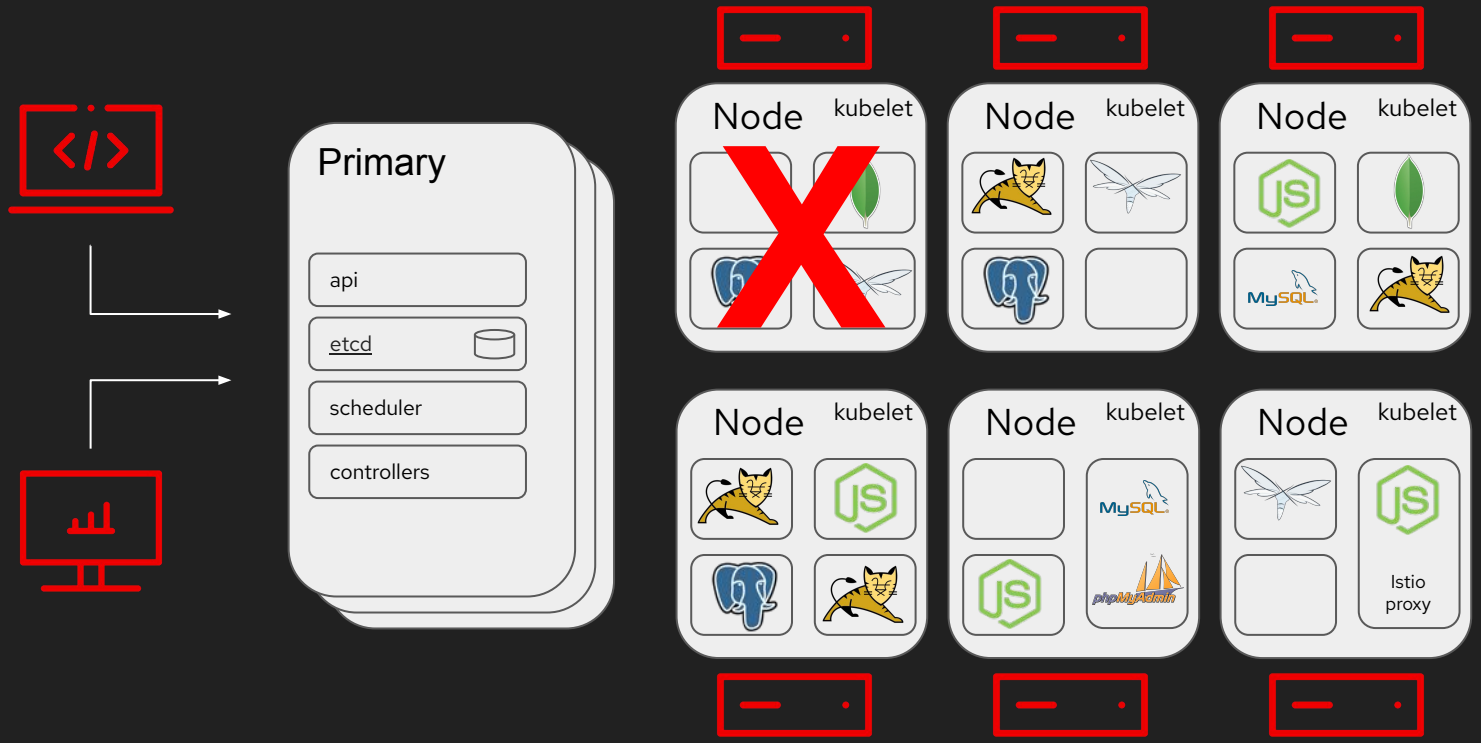
dn.dev/kubemaster1

Kubernetes Cluster - Recovery



dn.dev/kubemaster1

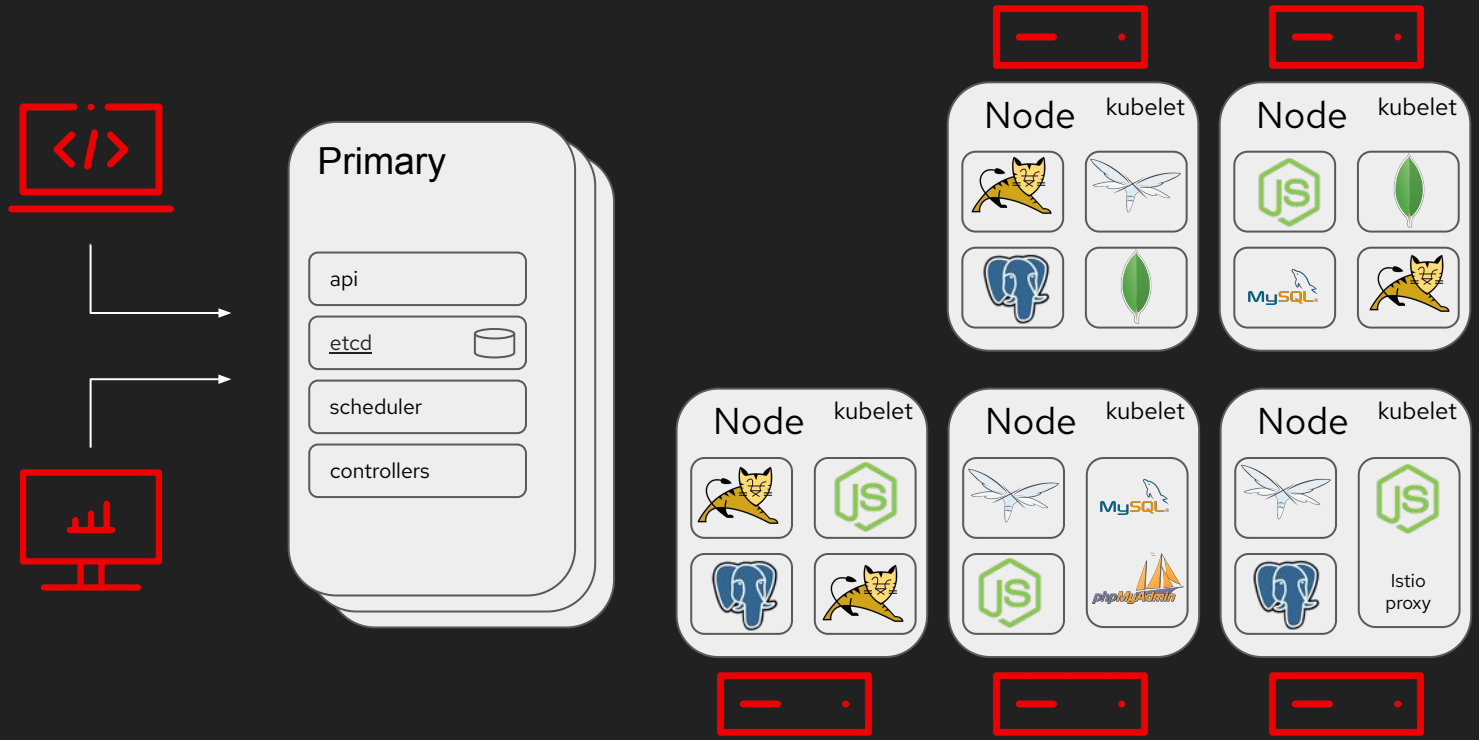
Kubernetes Cluster - Node Failure



dn.dev/kubemaster1

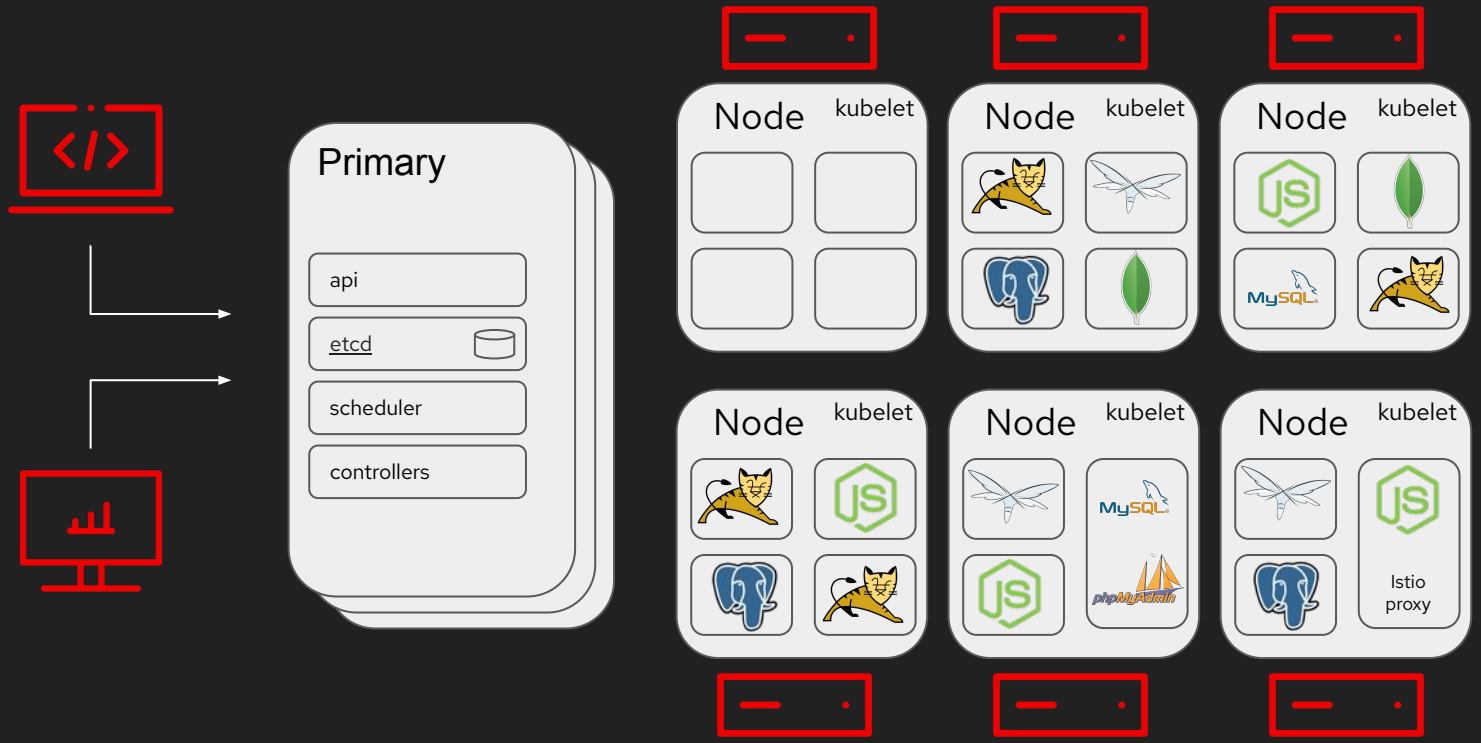
Red Hat Developer | DevNation

Kubernetes Cluster - Pods Replaced



dn.dev/kubemaster1

Kubernetes Cluster - New Node



dn.dev/kubemaster1


Red Hat
Developer
DevNation
Pod, ReplicaSet, Deployment
dn.dev/kubemaster1

Pod

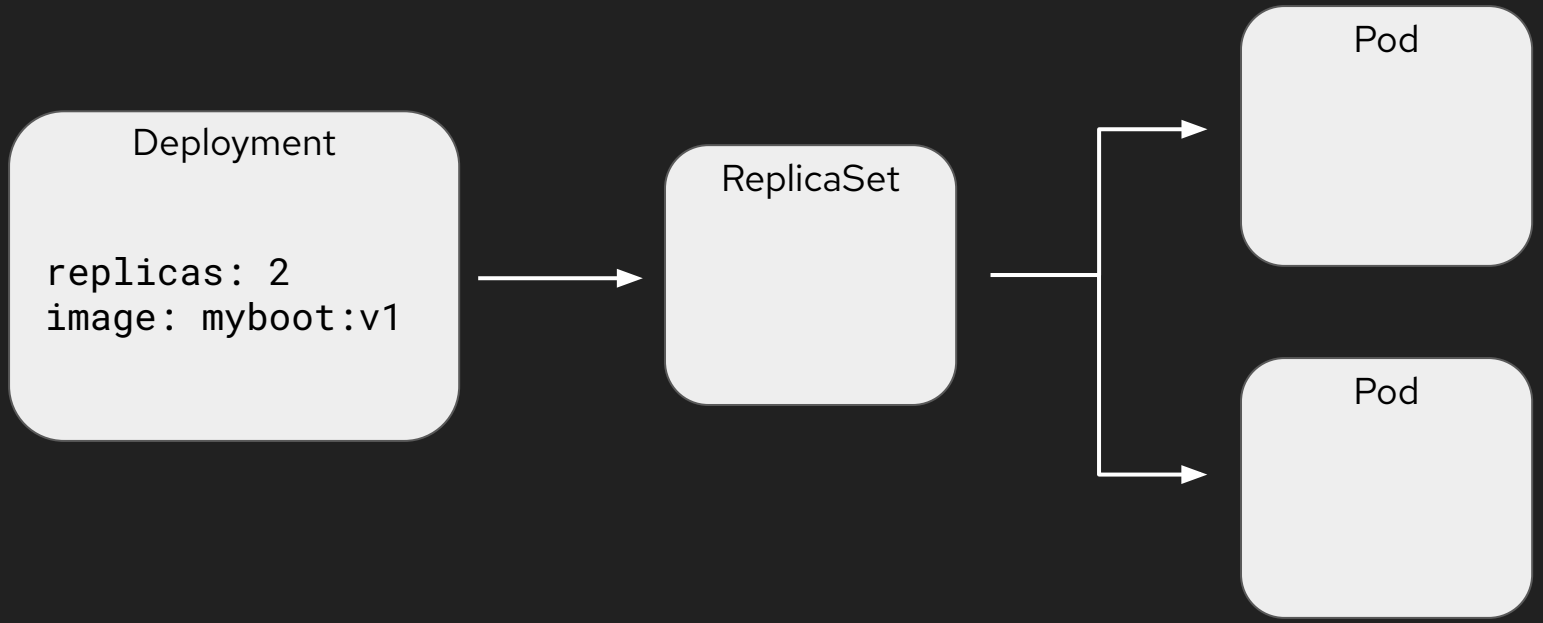
```
apiVersion: v1
kind: Pod
metadata:
  name: myboot-demo
spec:
  containers:
  - name: myboot-demo
    image: quay.io/rhdevelopers/myboot:v4
```

Red Hat Developer | DevNation

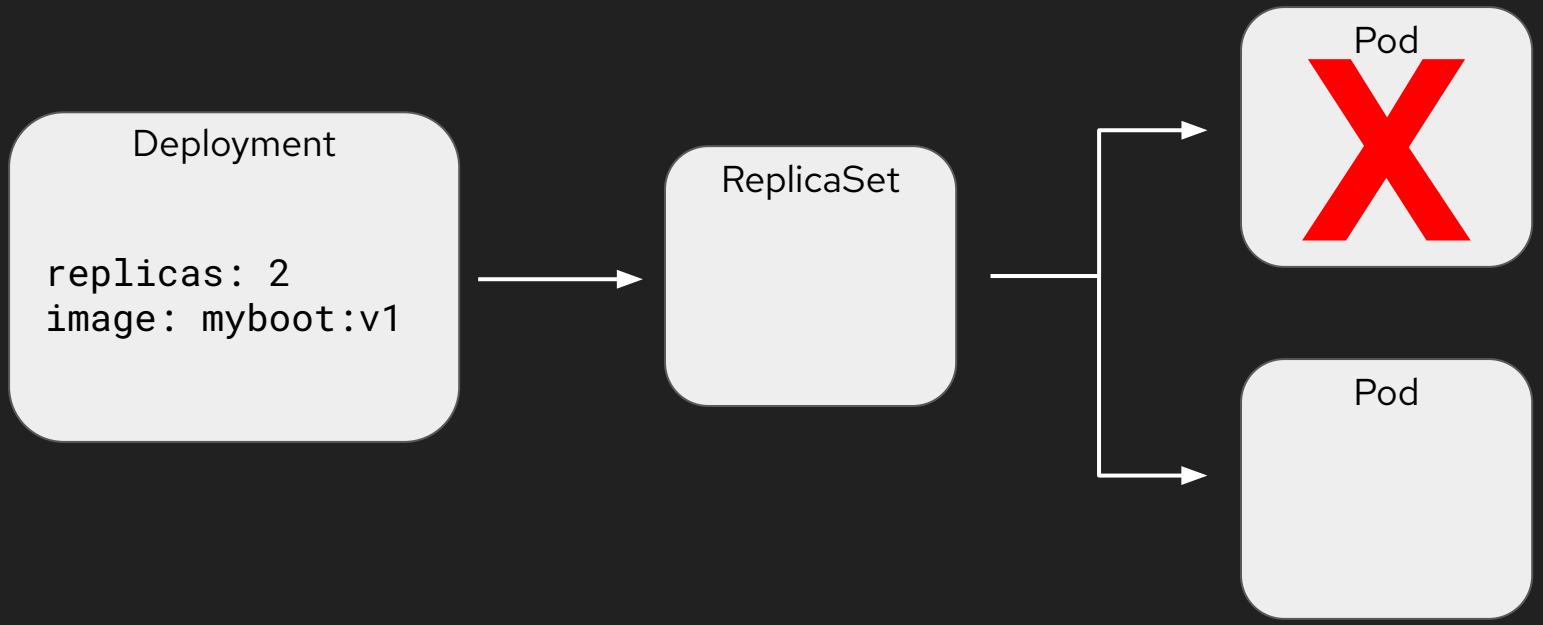
Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myboot
spec:
  replicas: 1
  selector:
    matchLabels:
      app: myboot
  template:
    spec:
      containers:
      - name: myboot
        image: quay.io/rhdevelopers/myboot:v1
        ports:
        - containerPort: 8080
```

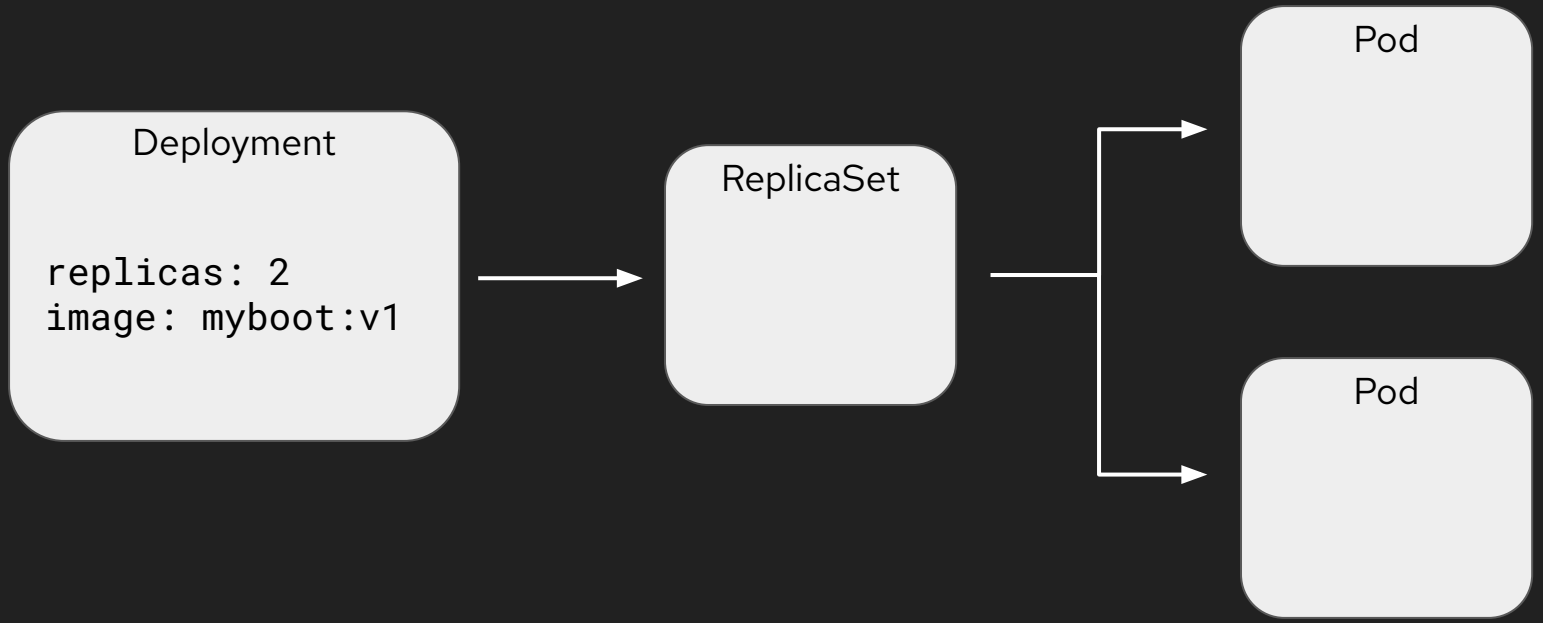
Deployments



Deployments



Deployments




Red Hat
Developer
DevNation
Rolling Updates
dn.dev/kubemaster2

"Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones."

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

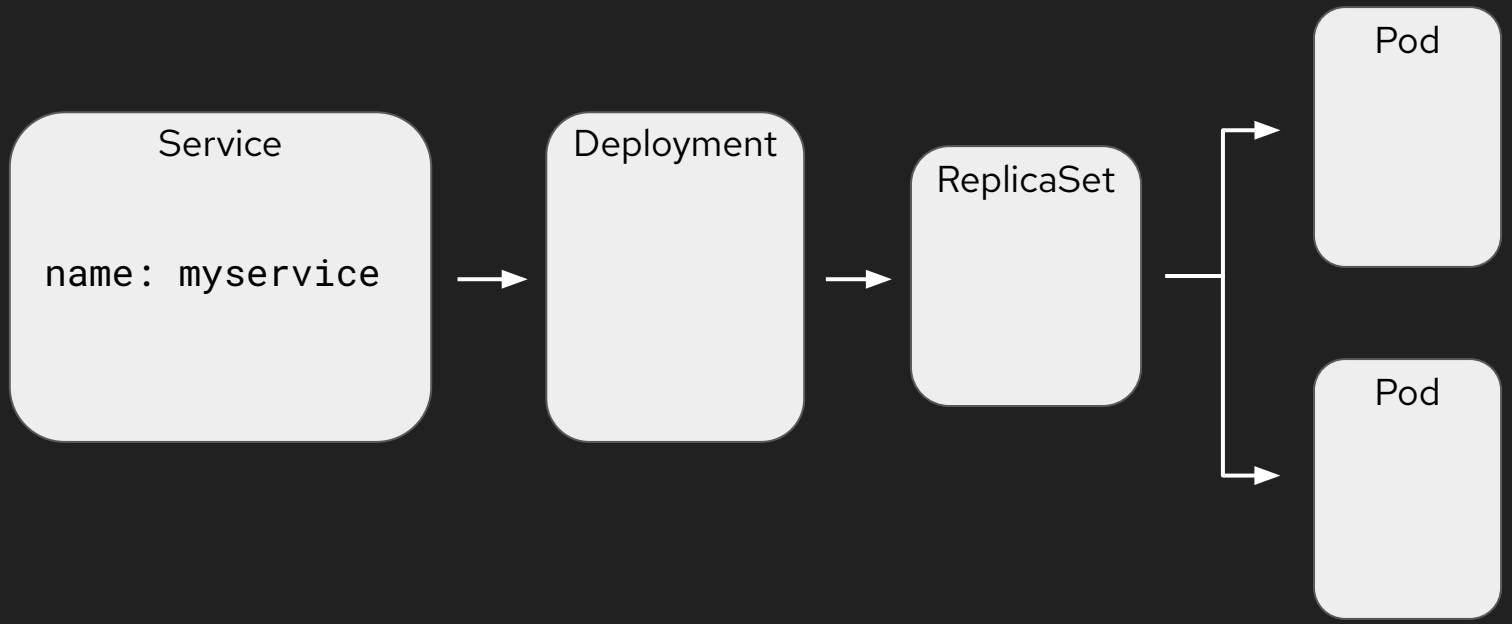
Service

```
apiVersion: v1
kind: Service
metadata:
  name: myboot
spec:
  ports:
  - name: http
    port: 8080
  selector:
    app: myboot
  type: LoadBalancer
```


Red Hat
Developer
DevNation
Service
dn.dev/kubemaster1

Red Hat Developer | DevNation

Service




Red Hat
Developer
DevNation
kubectl
dn.dev/kubemaster1

Red Hat Developer | DevNation

kubectl commands

https://kubernetes.io/docs/user-guide/kubectl/
https://kubernetes.io/docs/reference/kubectl/cheatsheet

```
kubectl get namespaces

kubectl get pods -n mynamespace

kubectl run myboot --image=quay.io/burrsutter/myboot:v1 --port=8080

kubectl logs myboot-75f94d76fc-pl2l7

kubectl expose deployment --port=8080 myboot --type=LoadBalancer

kubectl scale deployment myboot --replicas=3

kubectl set image deployment myboot myboot=quay.io/burrsutter/myboot:v2
```


Red Hat
Developer
DevNation
Logs
dn.dev/kubemaster1

logs

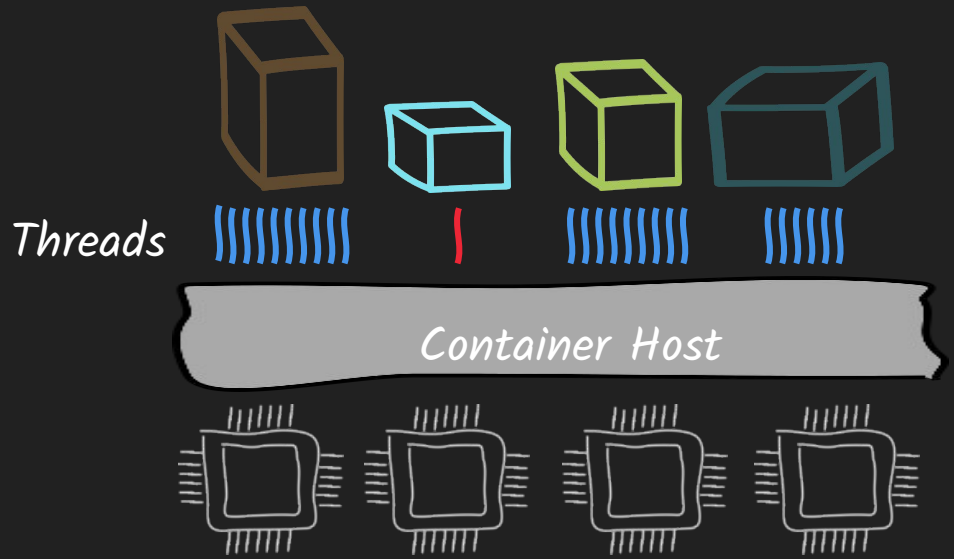
Red Hat Developer | DevNation

```
System.out.println("Java, where am I?");
Or console.log("Node logs");

kubectl get pods
kubectl logs microspringboot1-2-nz8f8
kubectl logs microspringboot1-2-nz8f8 -p # last failed pod

OR stern (brew install stern)
https://github.com/wercker/stern
stern myboot

OR kail (https://github.com/boz/kail)
kail -d myboot
```


Red Hat
Developer
DevNation
Resource Limits
dn.dev/kubemaster2

The hidden truth of containers

Containers are about **sharing** and **deployment density**



costs memory

cost lots of memory

cost CPU cycles

\+ quotas = BOOM

Resource Requests/Limits

Red Hat Developer | DevNation

```
resources:
  requests:
    memory: "300Mi"
    cpu: "250m" # 1/4 core
  limits:
    memory: "400Mi"
    cpu: "1000m" # 1 core
```


Red Hat
Developer
DevNation
Liveness & Readiness
dn.dev/kubemaster2

"The kubelet uses liveness probes to know when to restart a container."

"The kubelet uses readiness probes to know when a container is ready to start accepting traffic."

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

Example

Red Hat Developer | DevNation

```
kubectl create -f Deployment.yml
```

```
resources:
  requests:
    memory: "300Mi"
    cpu: "250m" # 1/4 core
  limits:
    memory: "400Mi"
    cpu: "1000m" # 1 core
livenessProbe:
  httpGet:
      port: 8080
      path: /
  initialDelaySeconds: 10
  periodSeconds: 5
  timeoutSeconds: 2
readinessProbe:
  httpGet:
    path: /health
    port: 8080
  initialDelaySeconds: 10
  periodSeconds: 3
```


Red Hat
Developer
DevNation
Environment & ConfigMap
dn.dev/kubemaster2

env vars & configmaps

Red Hat Developer | DevNation

An app's config is everything that is likely to vary between deploys (staging, production, developer environments, etc). *12 Factor Apps*

```
kubectl set env deployment/myboot DBCONN="jdbc:sqlserver://45.91.12.123:1443;user=MyUserName;password=*****;"

kubectl create cm my-config --from-env-file=config/some.properties
```

ConfigMaps

Red Hat Developer | DevNation

```
kubectl create cm my-config --from-env-file=config/some.properties
```

some.properties

```
GREETING=jambo

LOVE=Amour
```

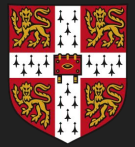
Partial deployment.yml

```
spec:
 containers:
 - name: myboot
   image: 9stepsawesome/myboot:v1
   ports:
     - containerPort: 8080
   envFrom:
   - configMapRef:
       name: my-config
```

Partial .java

```
@Autowired

private Environment environment;

String greeting =
environment.getProperty("GREETING","Default");

String love =
environment.getProperty("LOVE","Default");
```

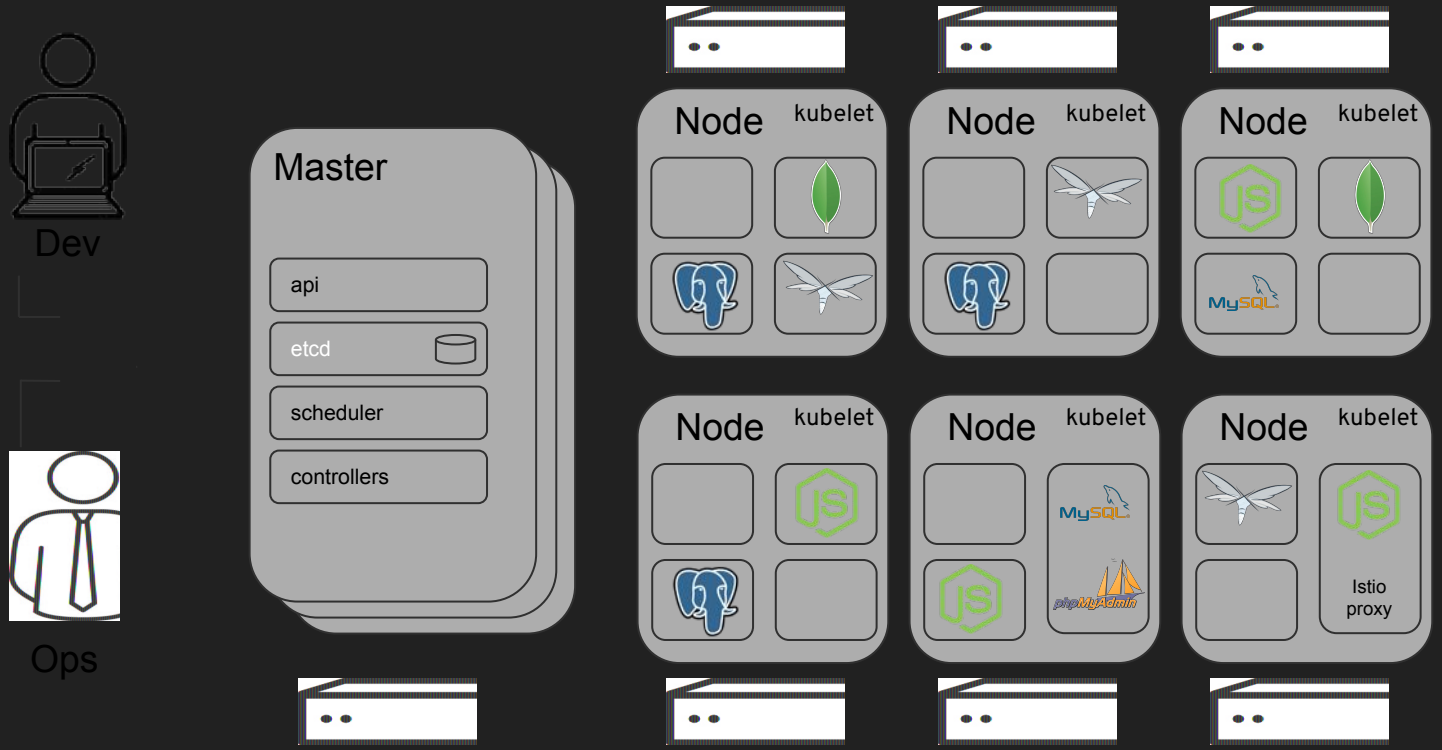

Red Hat
Developer
DevNation
Secrets
dn.dev/kubemaster3



Cambridge Dictionary

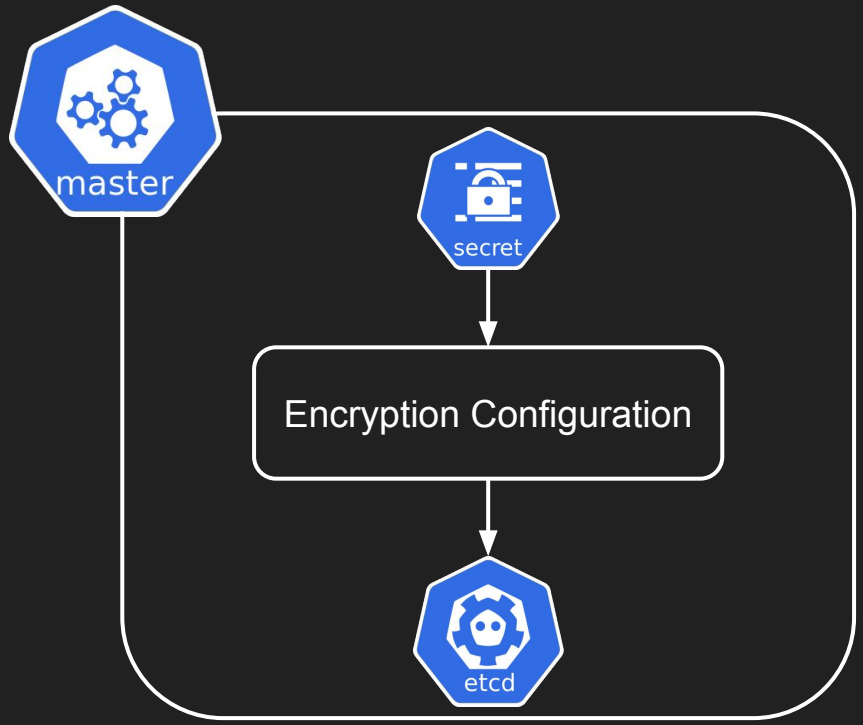
Secret: a piece of information that is only known by one person or a few people and should not be told to others.

Kubernetes Cluster - Nodes

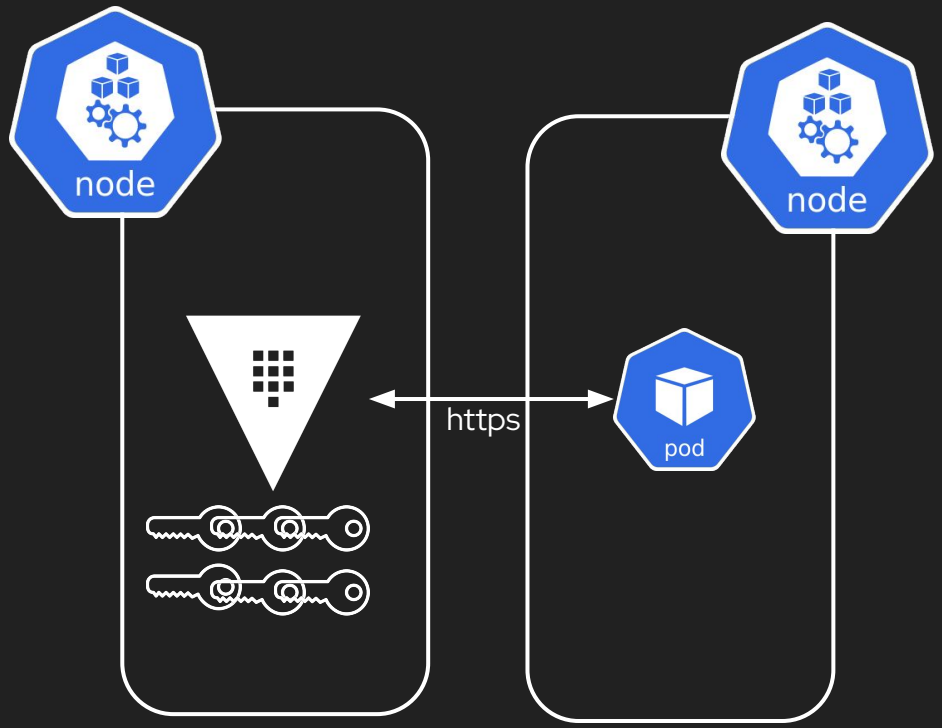


dn.dev/kubemaster3

- Kubernetes object that contains a small amount of sensitive data.
- Injected as volume or environment variable.


master
secret
Encryption Configuration
etcd

Vaults provide full encryption



dn.dev/kubemaster3


Red Hat
Developer
DevNation
Volumes
dn.dev/kubemaster3

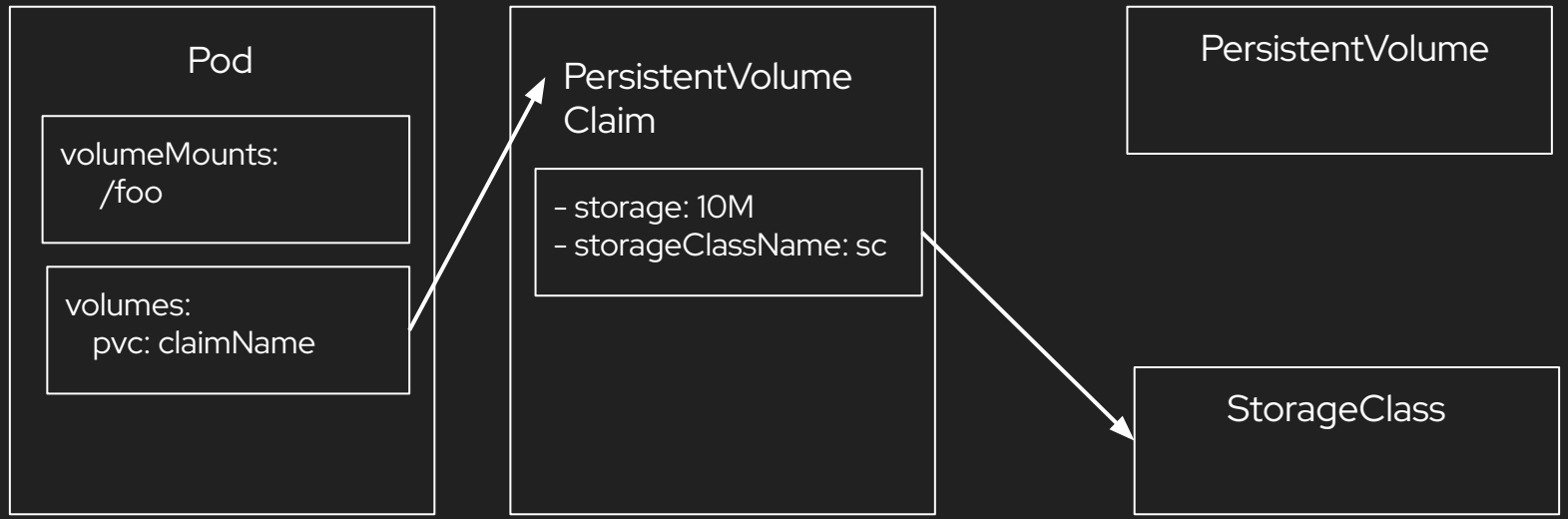
The data in volumes is preserved across container restarts. Lifecycle tied to a pod.

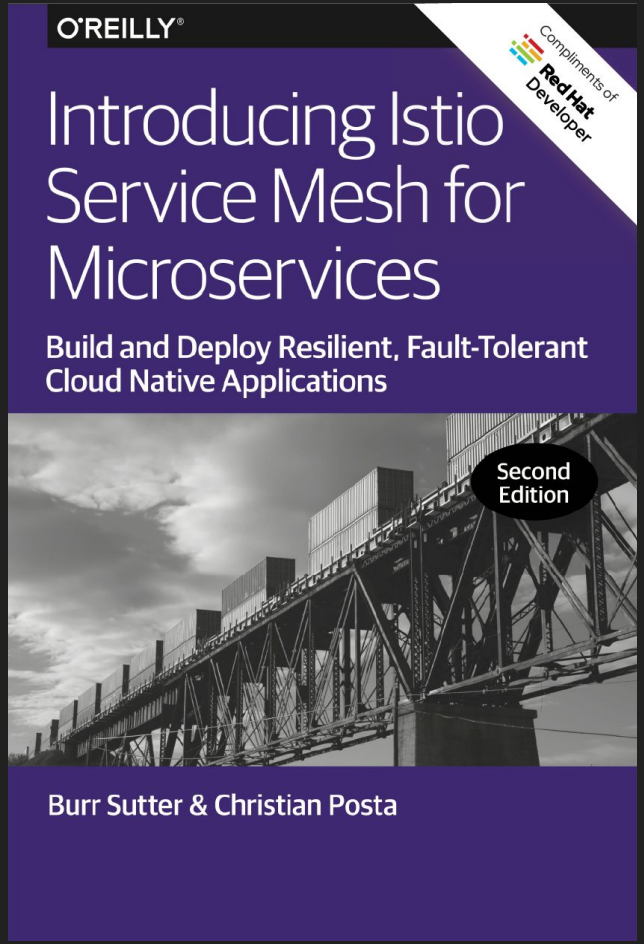
Networked storage. Data is persisted beyond the lifetime of a pod.

Storage needs to be claimed.

Storage Class for dynamic provisioning of storage.

https://kubernetes.io/docs/concepts/storage/volumes/


Pod
volumeMounts:
/foo
volumes:
pvc: claimName
PersistentVolume Claim
- storage: 10M
- storageClassName: sc
PersistentVolume
StorageClass

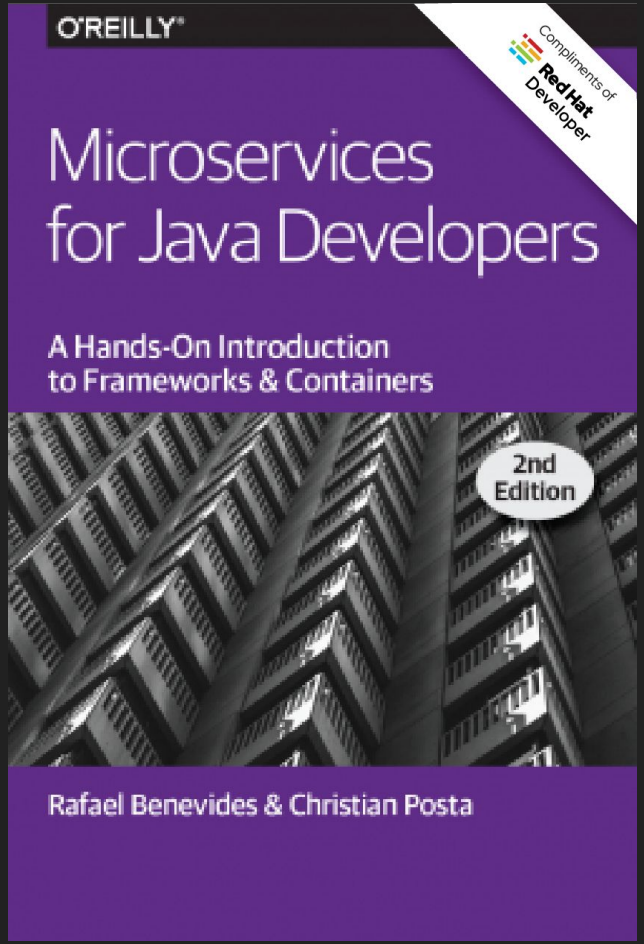


Red Hat Developer | DevNation

Download

https://dn.dev/istiobook

dn.dev/kubemaster1

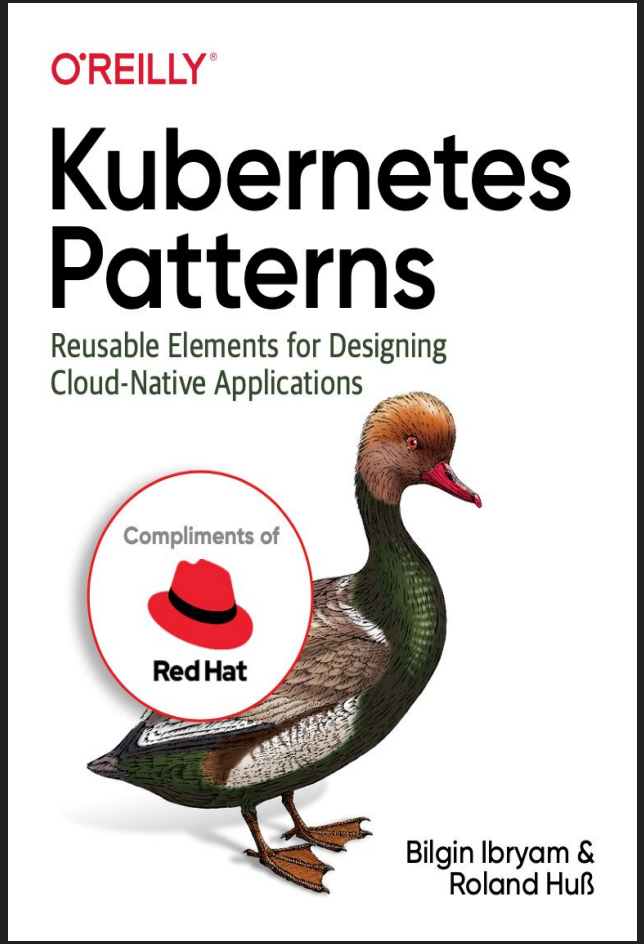


Red Hat Developer | DevNation

Download

https://dn.dev/javamsbook

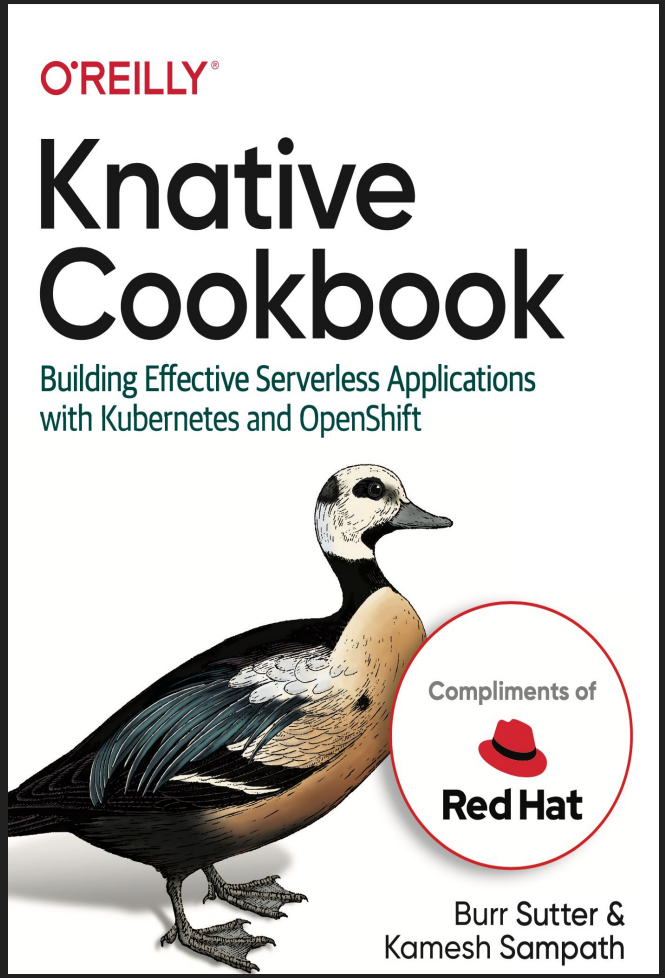
dn.dev/kubemaster1



Red Hat Developer | DevNation

Download

https://dn.dev/k8s-patterns

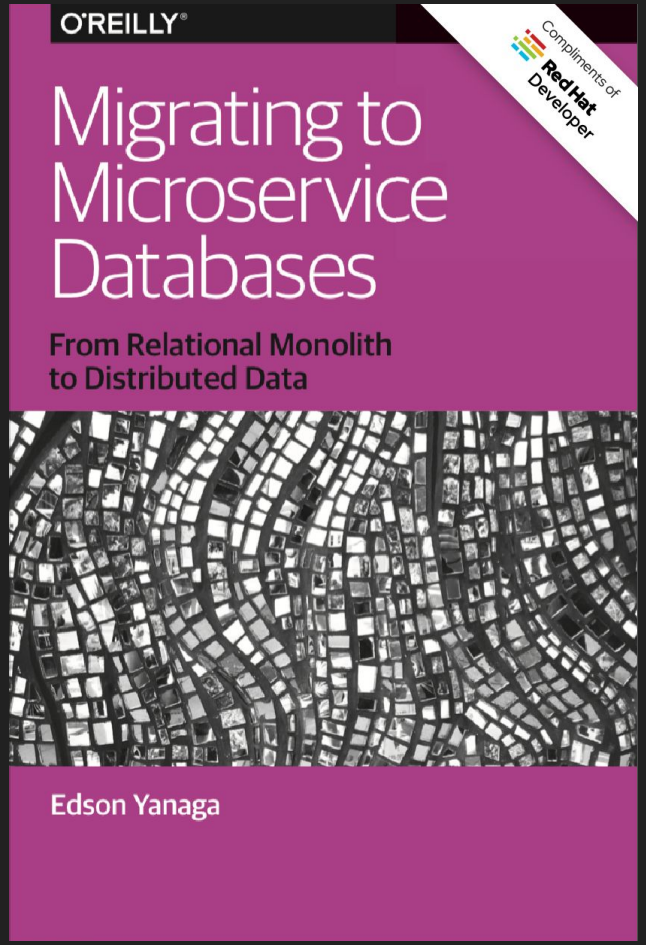


Red Hat Developer | DevNation

Download

dn.dev/knative-cookbook

dn.dev/kubemaster1



Red Hat Developer | DevNation

Download

bit.ly/mono2microdb

dn.dev/kubemaster1