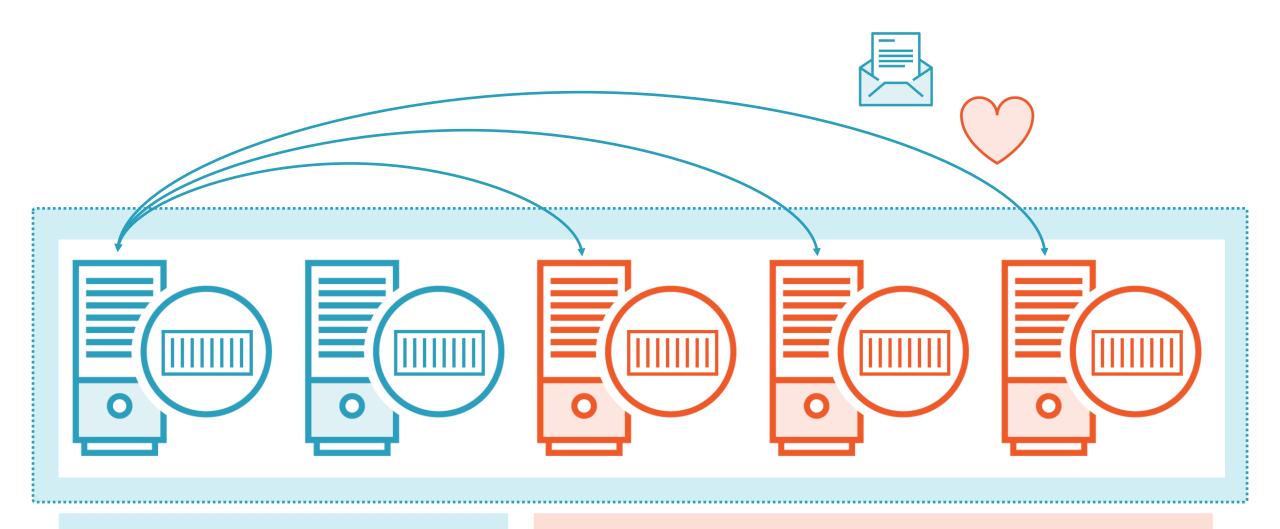
Understanding Kubernetes



Elton Stoneman CONSULTANT & TRAINER

@EltonStoneman | blog.sixeyed.com



Control Plane

Workers



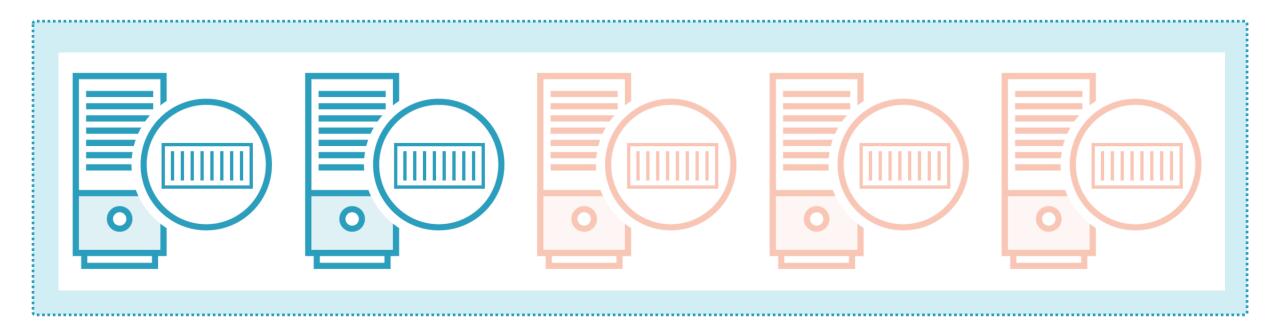




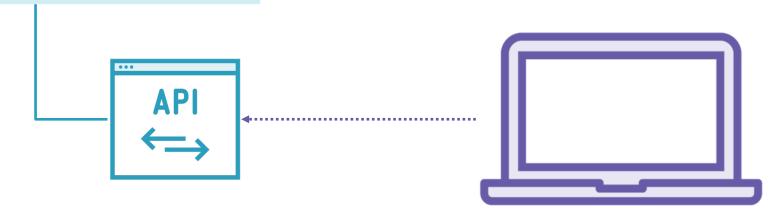


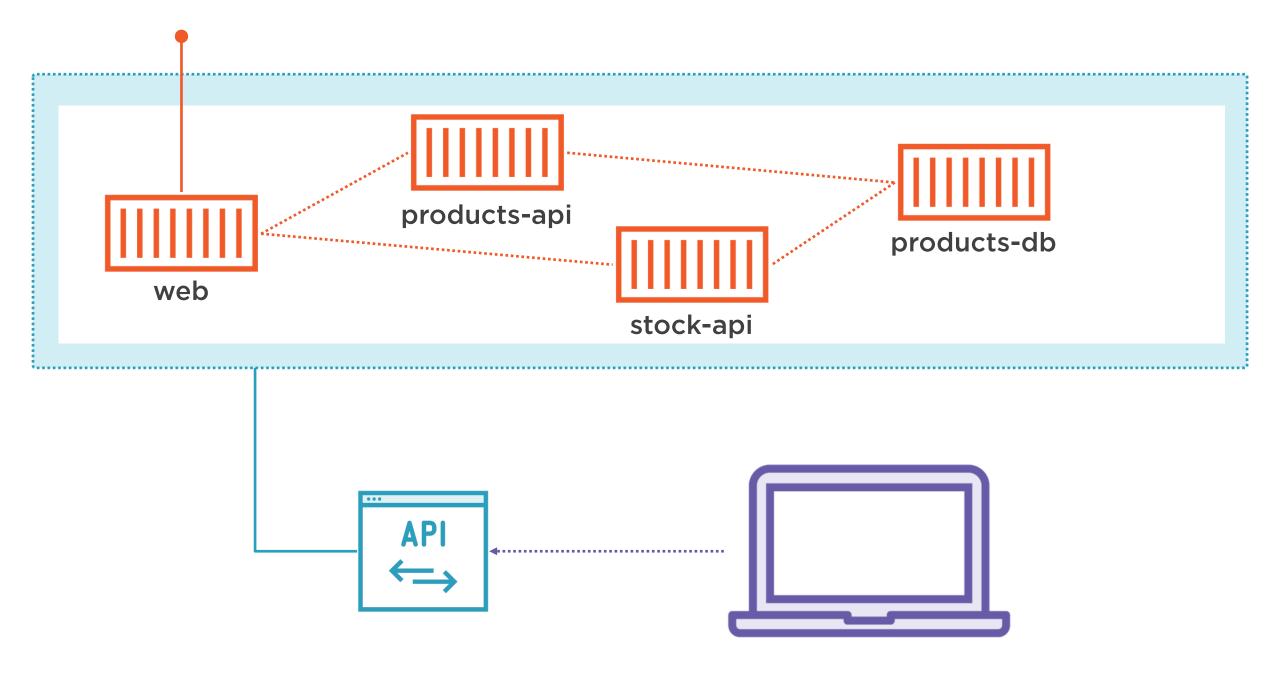


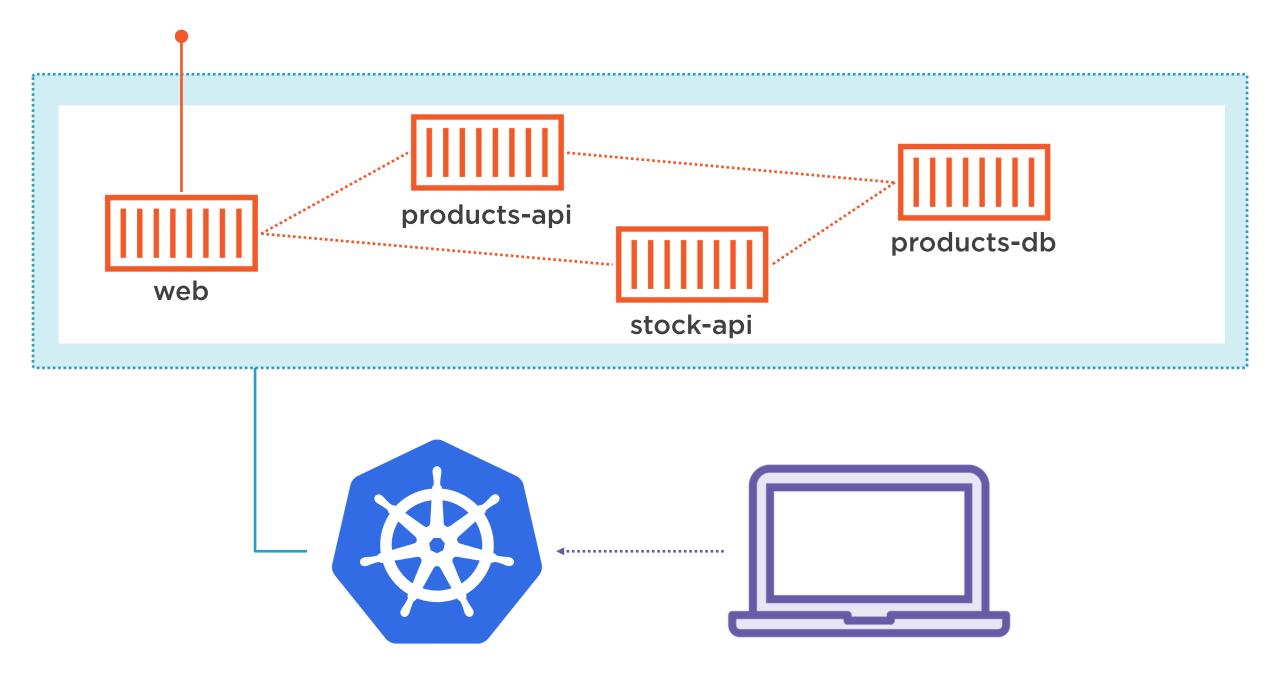


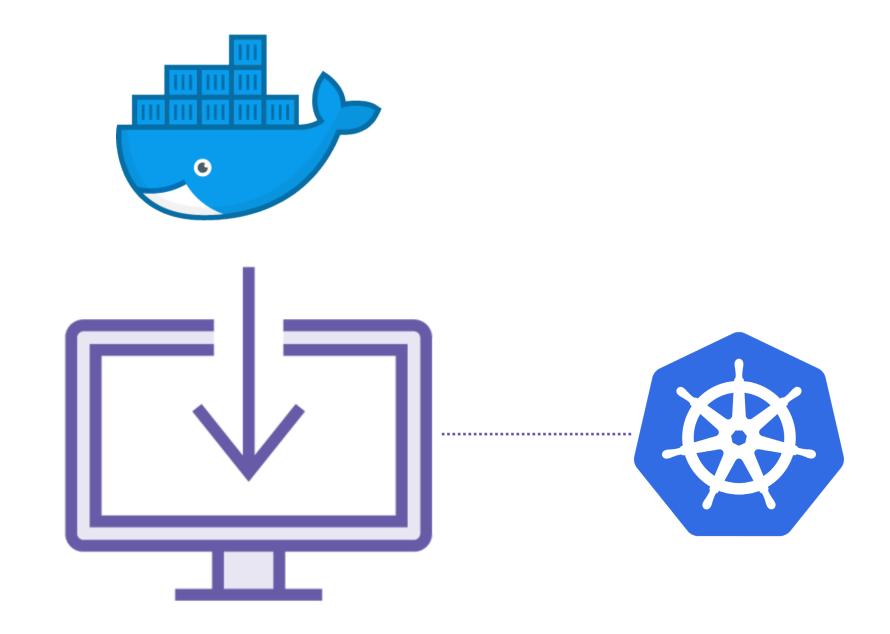


Control Plane





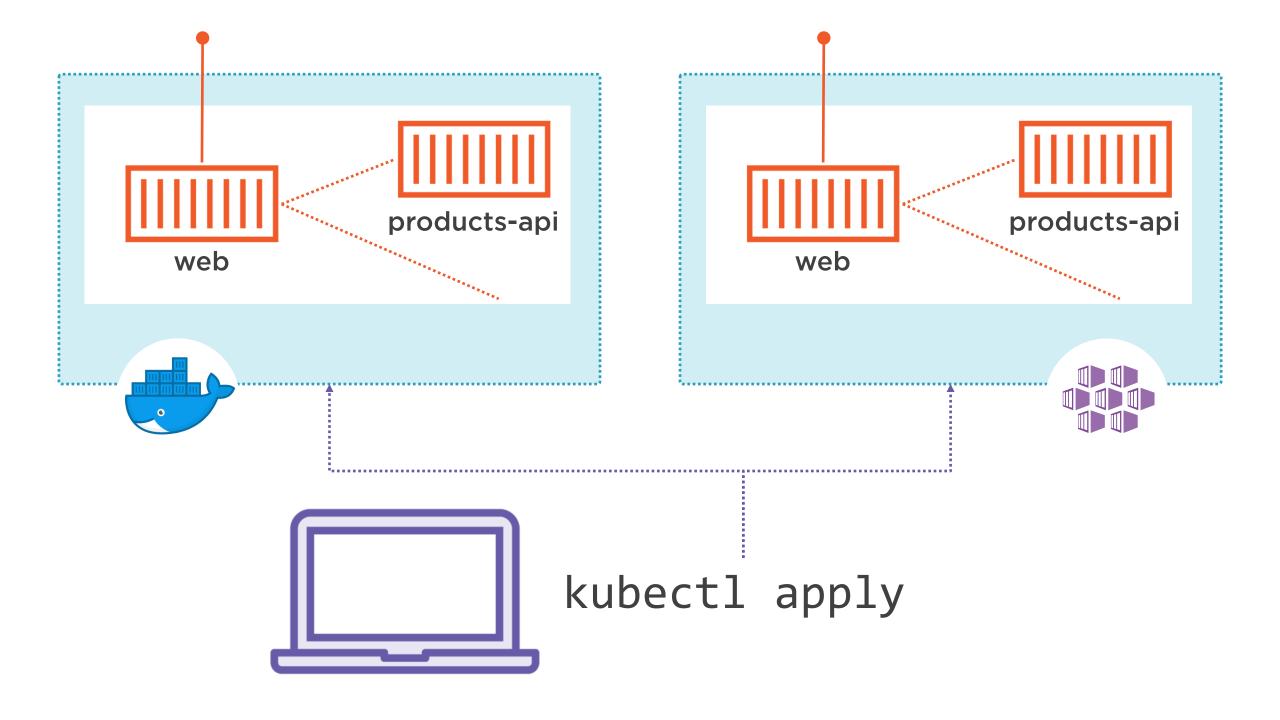




```
az aks create `
  -n 'demo-cluster' -g 'ps-demos' `
  -node-count 100
```

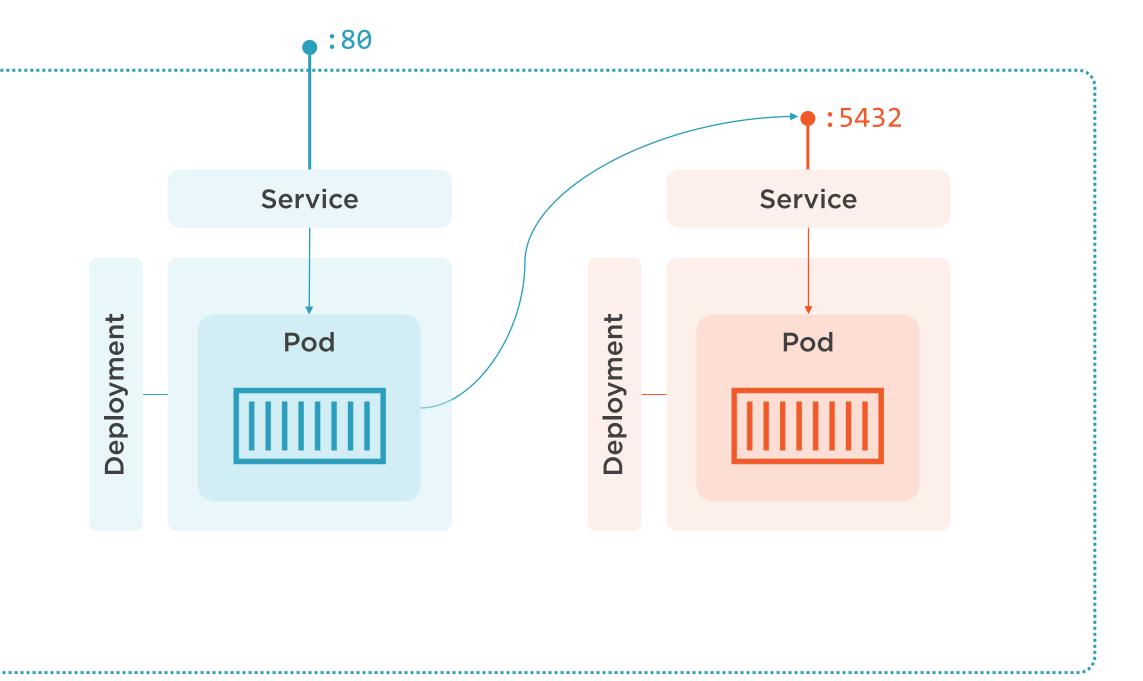
Azure Kubernetes Service

Managed cluster with pay-per-node model



Modelling Applications with Kubernetes

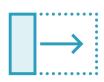




Core Kubernetes Resources



||||||||| Pod: manages containers & container environment



Deployment: manages Pods & rolling upgrades



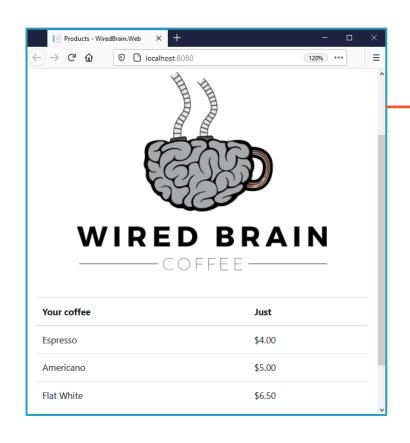
Service: manages network routing & DNS names

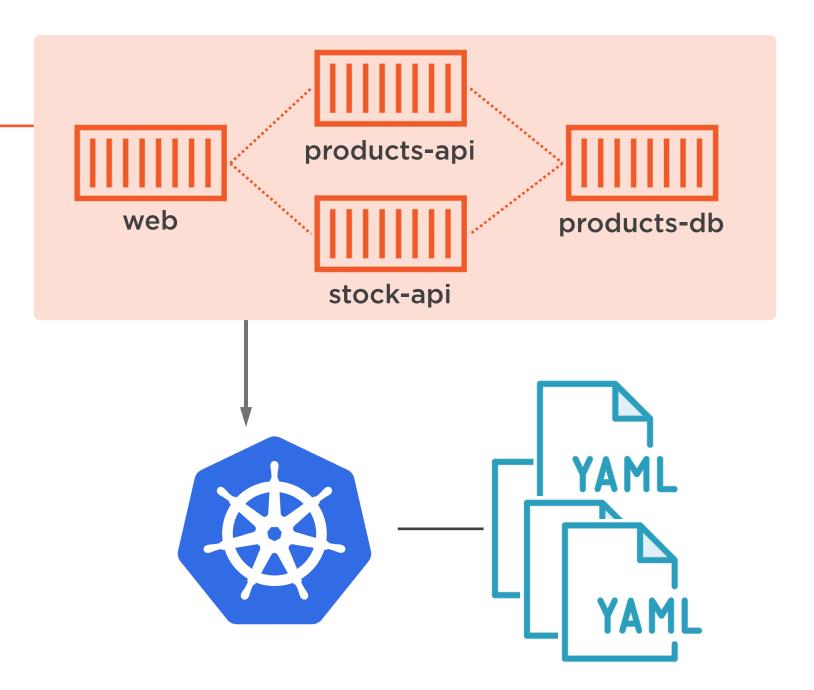
Demo



Deploying apps to Kubernetes

- Creating Services
- Creating Deployments
- Managing Kubernetes resources





kubectl get nodes
kubectl describe node

Cluster Management
Initial deployment with kubeadm

db-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: products-db
spec:
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: products-db
  type: ClusterIP
```

db-deployment.yaml

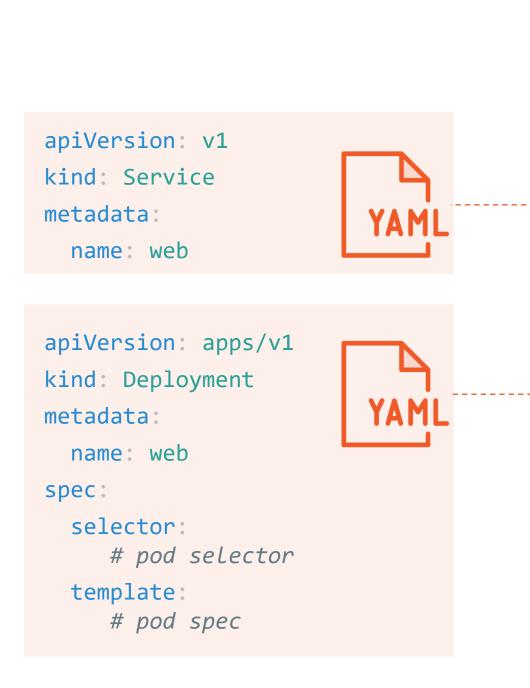
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products-db
spec:
  selector:
    matchLabels:
      app: products-db
  template:
    metadata:
      labels:
        app: products-db
    spec:
      containers:
        - image: products-db
```

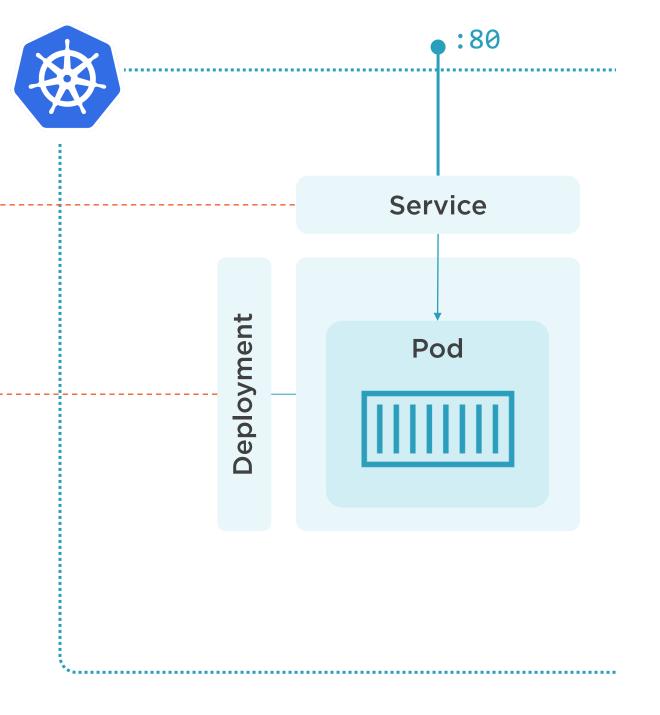
web.yaml

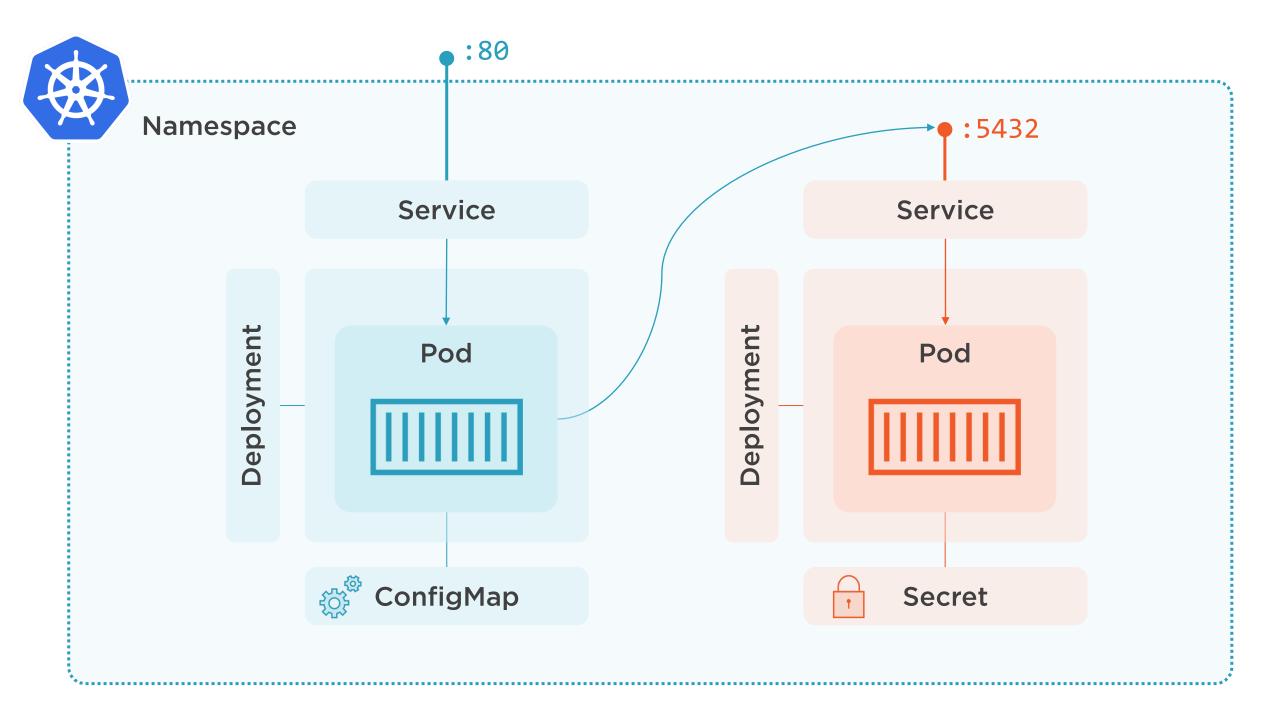
```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  ports:
    - port: 8080
      targetPort: 80
  selector:
    app: web
  type: LoadBalancer
```

web.yaml (continued)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - image: psdockerrun/web
```





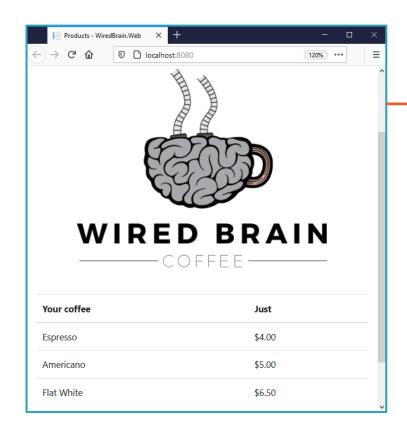


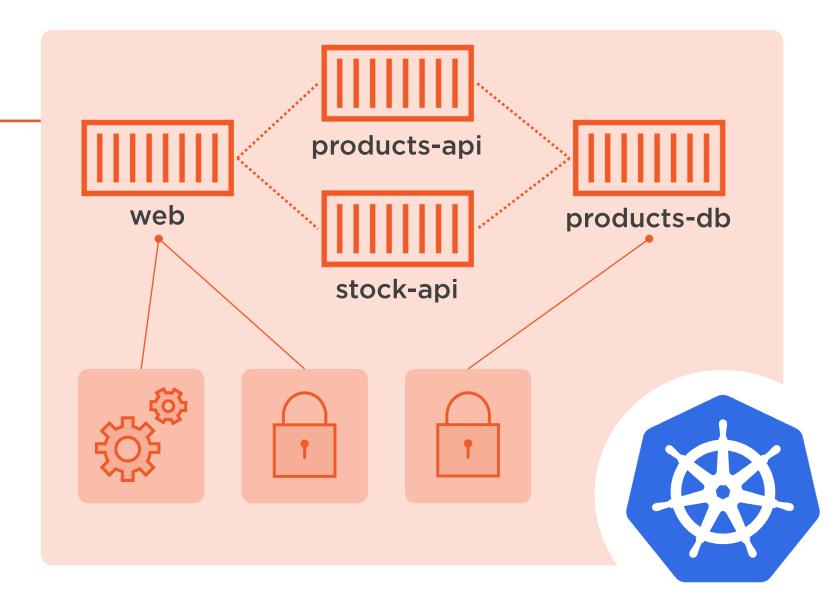
Demo



Configuring apps with Kubernetes

- Creating ConfigMaps
- Creating Secrets
- Modelling apps in Namespaces





kubectl apply -f configMaps/
kubectl create secret --from-file

Storing App Config in Kubernetes

YAML model or imperative commands

api-properties.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: products-api-properties
  namespace: wb-test
data:
  application.properties: -
   logging.level=DEBUG
   management.endpoints=prometheus
    server.port=80
    spring.jpa.show-sql=true
    spring.jpa.generate-ddl=true
```

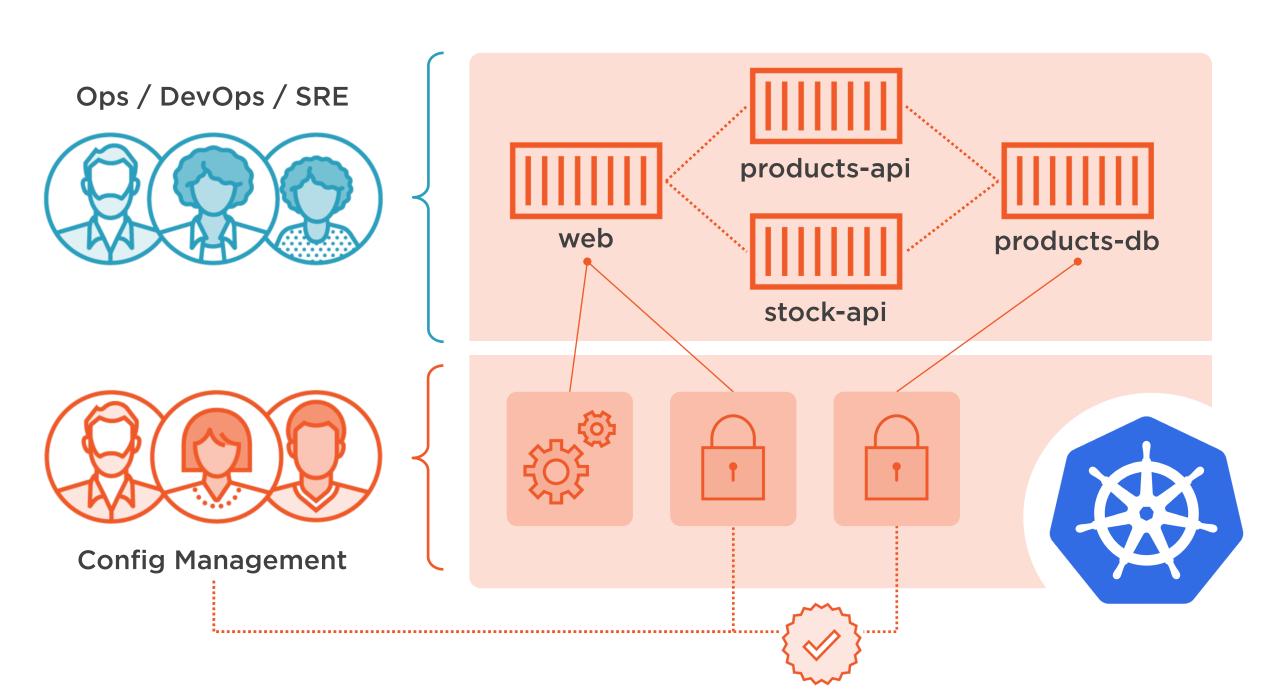
web-logging.yaml

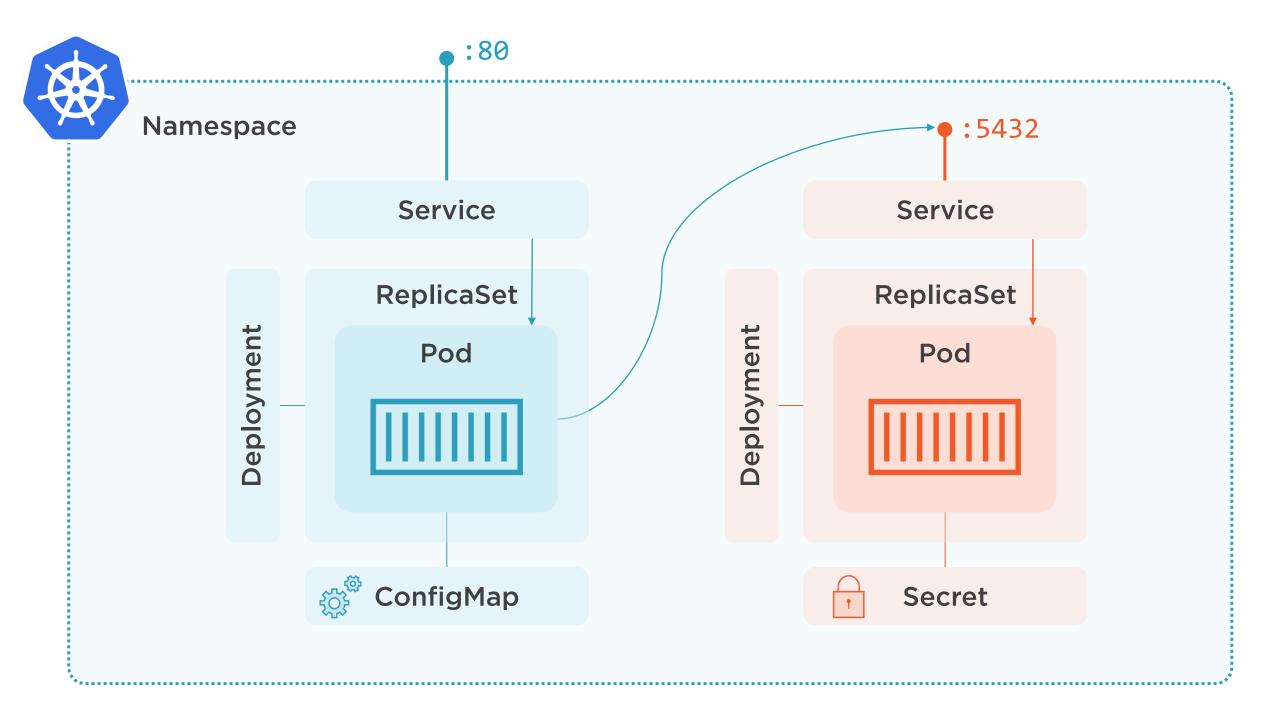
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: web-logging
  namespace: wb-test
data:
  logging.json: -
      "Logging": {
        "LogLevel": {
          "Default": "Warning"
```

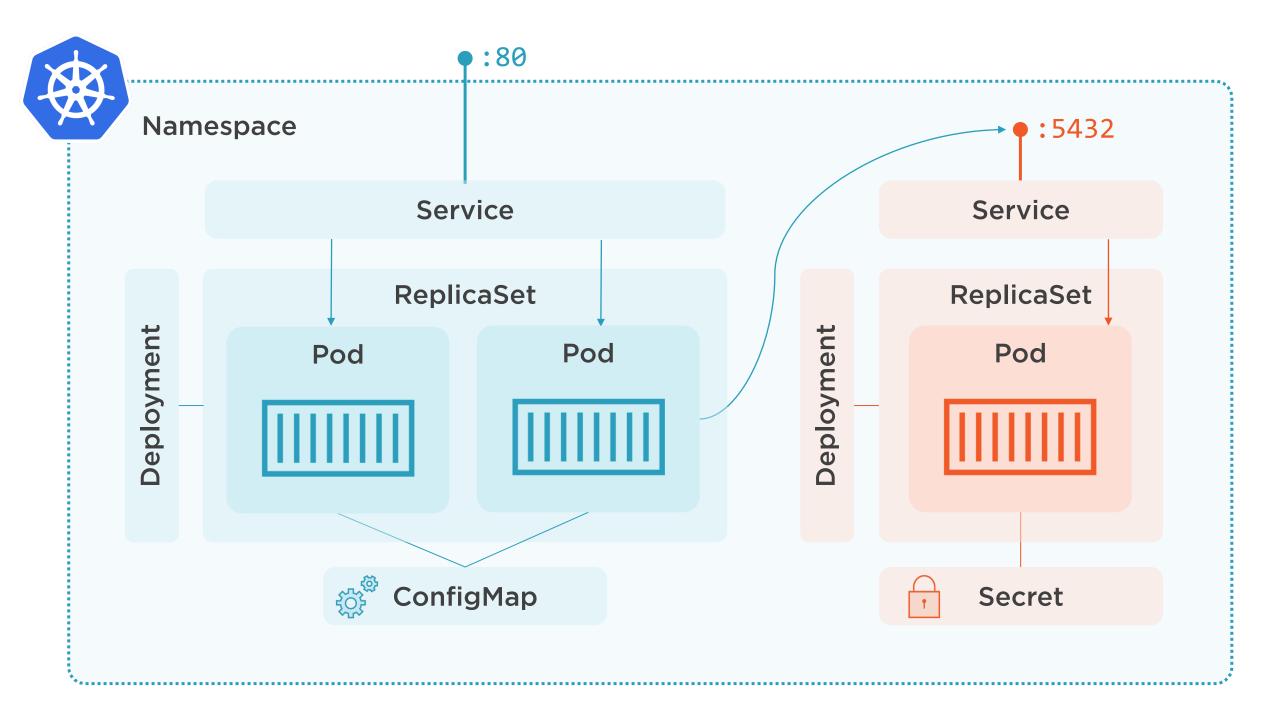
db-password.yaml

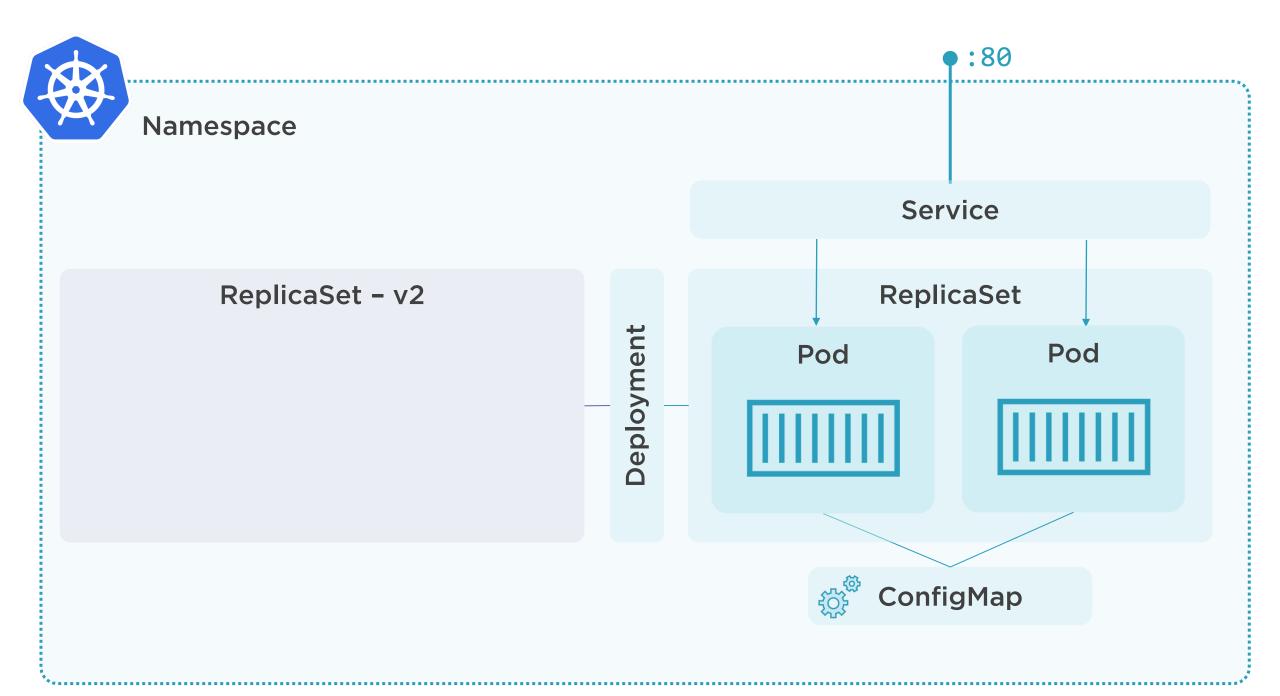
```
apiVersion: v1
kind: Secret
metadata:
  name: products-db-password
  namespace: wb-test
type: Opaque
stringData:
  pg-password: |-
    wiredtestm3
```

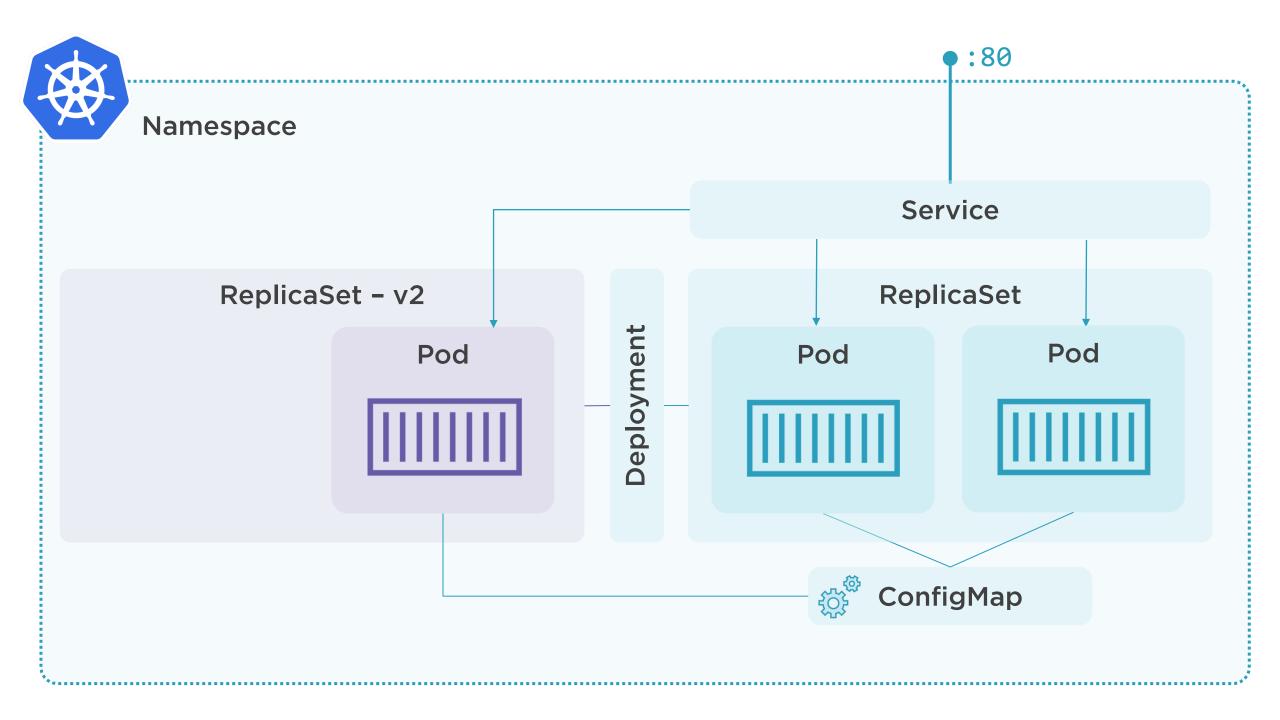
stock-api-connection.yaml

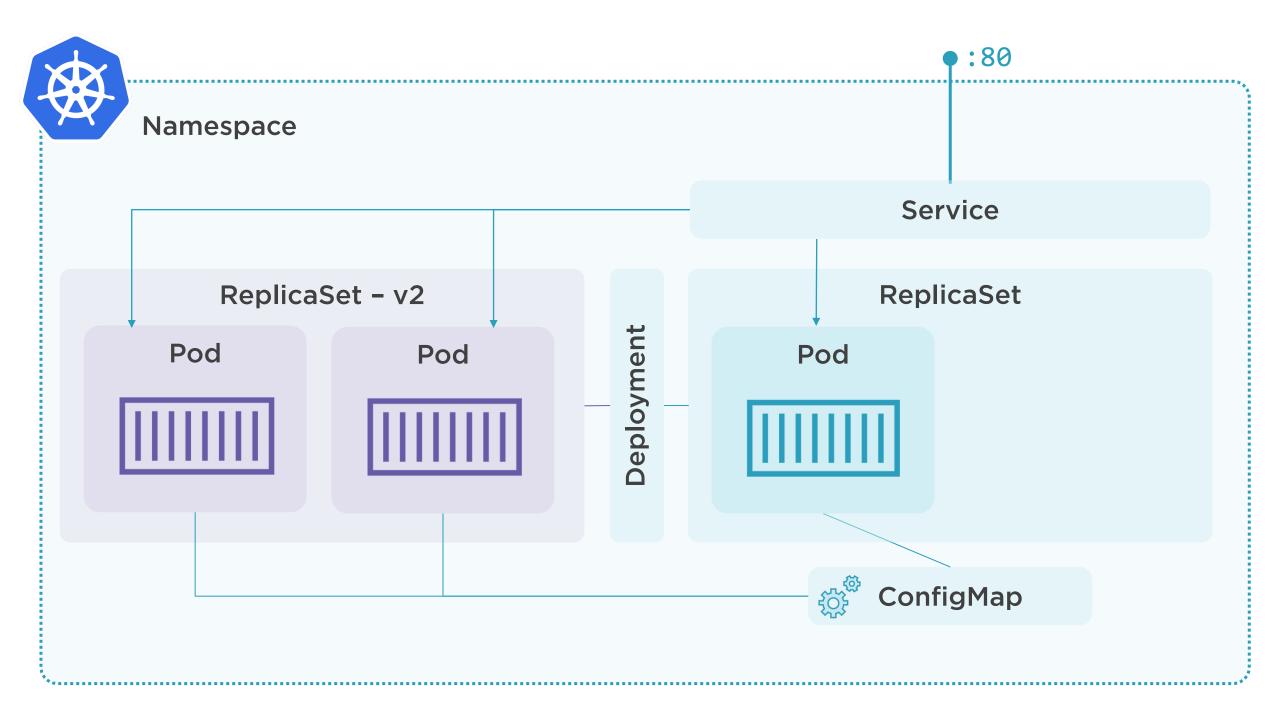


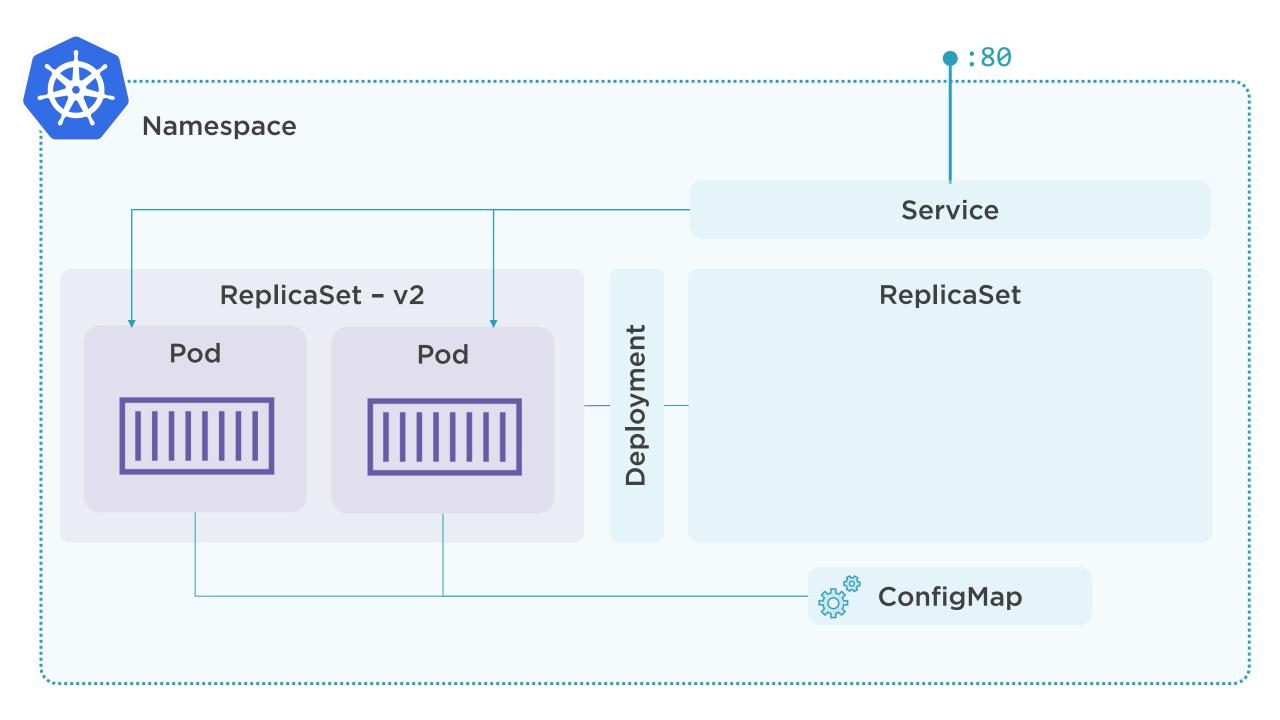










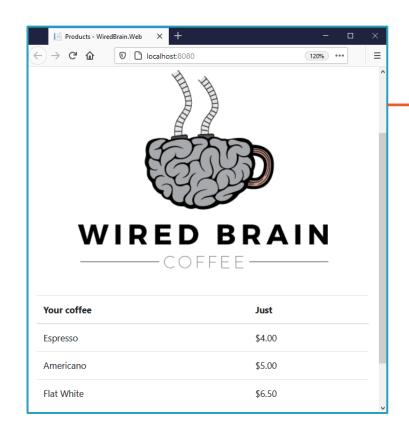


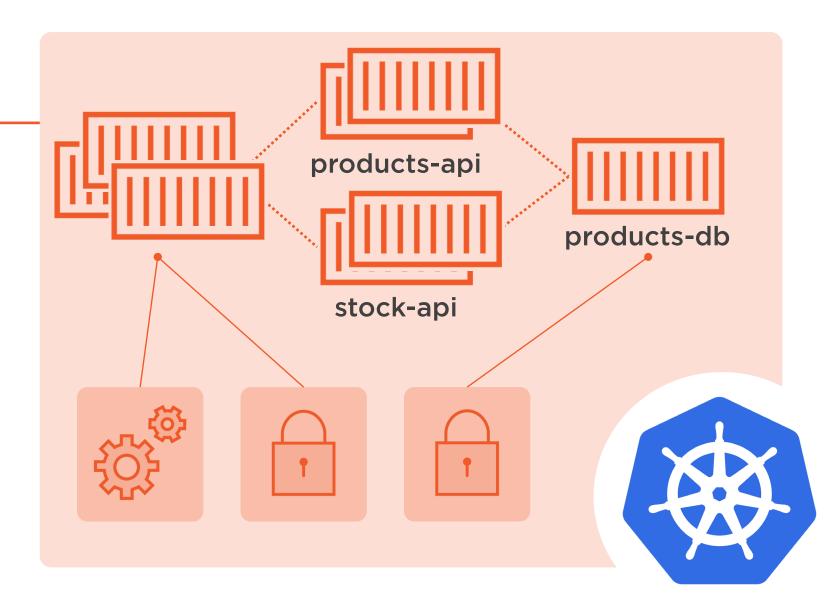
Demo

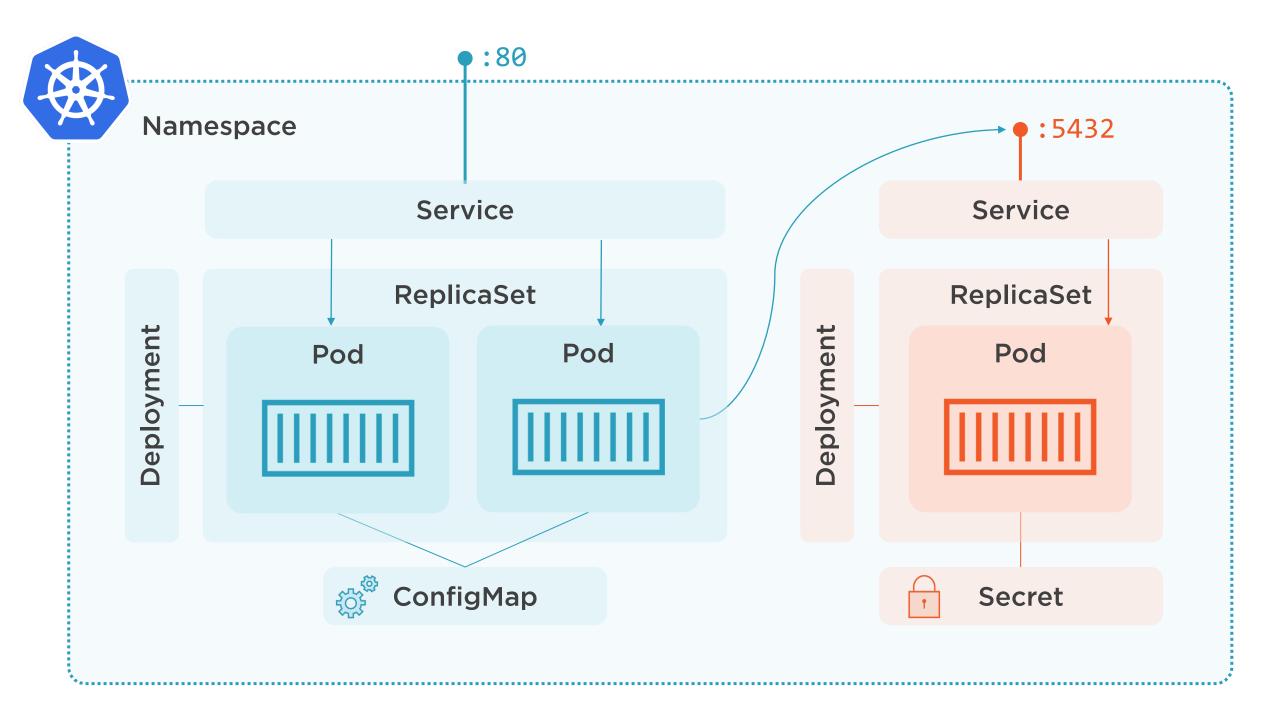


Scale and Reliability in Kubernetes

- Restarting failed containers
- Scaling ReplicaSets
- Safe rolling updates







web.yaml

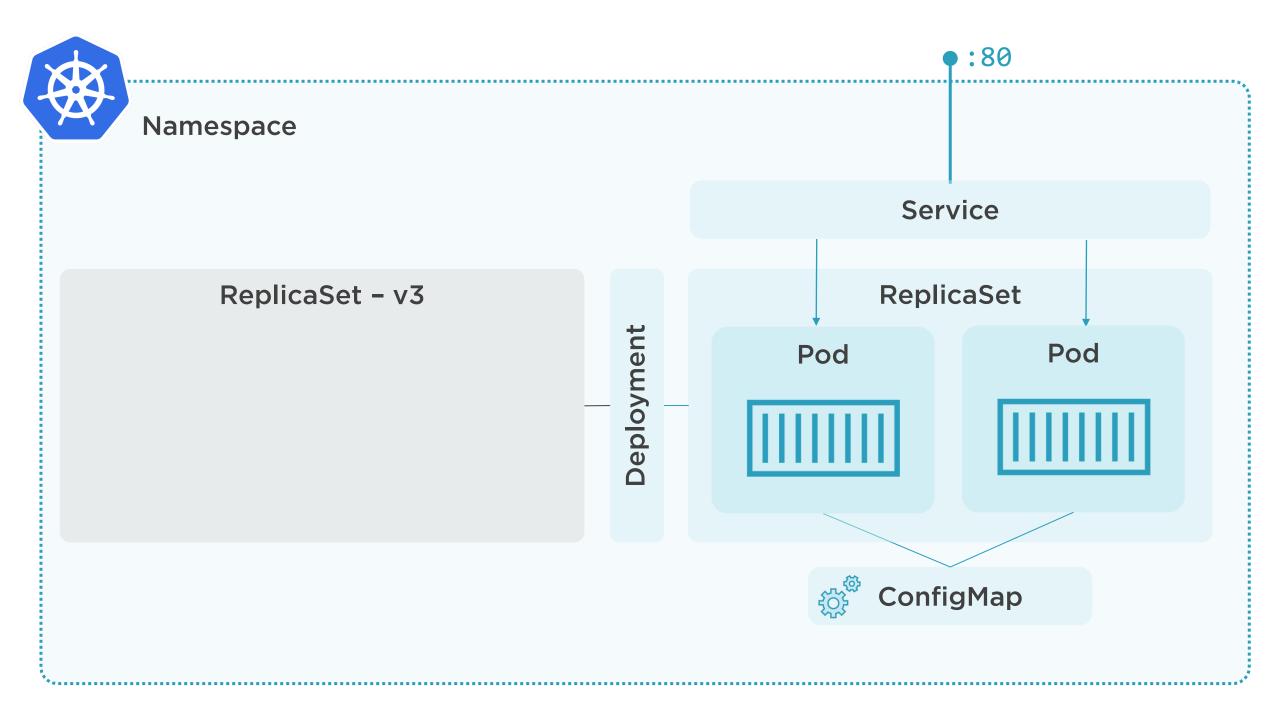
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  namespace: wb-test-2
spec:
  replicas: 3 # managed by replicaset
  selector:
    matchLabels:
      app: web
  template:
    # pod spec follows
```

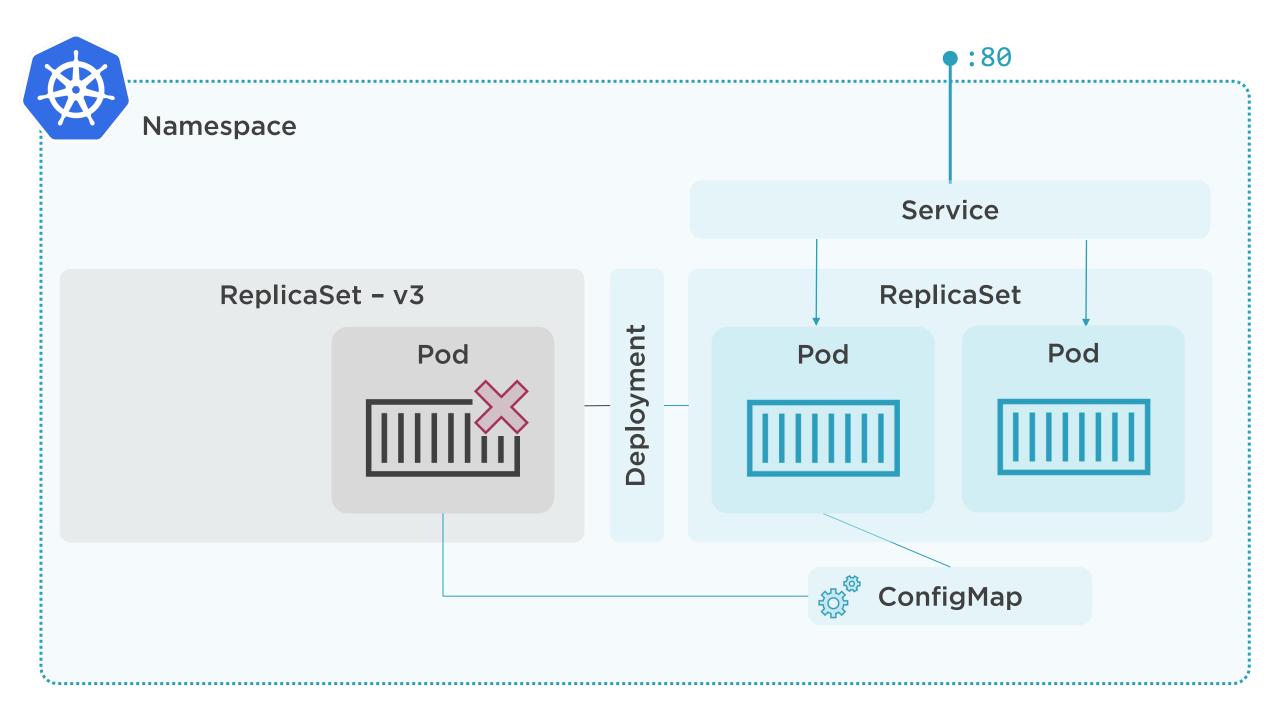
web.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
  namespace: wb-test-2
spec:
  replicas: 3 # managed by replicaset
  selector:
   matchLabels:
      app: web
  template:
   # pod spec follows
```

web-v3.yaml

```
template:
  metadata:
    labels:
      app: web
      version: v3
  spec:
    containers:
      - name: api
        image: psdockerrun/web:v3
        env:
          - name: Environment
            value: TEST
          - name: Debug ShowHost
            value: "true"
```







Course into	
Rating	★★★★★ (110)
Level	Intermediate 👊
Updated	Feb 7, 2020
Duration	3h 34m - ©

Description

Istio is a service mesh - a component which lets you take control of the network communication between your application services. You can manage traffic routing, security, and telemetry centrally without changing code or configuration. In this course, Managing Apps on Kubernetes with Istio, you will learn what you can do with a service mesh. First, you will explore blue/green and canary deployments. Next, you will learn about authentication, authorization, and how to view the health and status of your services. Finally, you will discover how to work with Istio in a local environment, and what you need to know for running Istio in production. When you are finished with the course, you will have the skills to deploy Istio and run new and old applications in the service mesh.

What Is a Service Mesh? 6m Understanding Istio's 4m Features Demo: Installing Istio on 6m **Kubernetes with Docker** Desktop Examining Istio's 6m Architecture and Running Costs Demo: Running the BookInfo 6m App with Istio Using a VirtualService to 5_m Manage Traffic Demo: Adding Fault 6m Tolerance with Istio 2m Managing Service Traffic Understanding 6m VirtualServices, DestinationRules, and Subsets Demo: A Dark Launch for a 8m New Feature Using Gateways with 5m VirtualServices to Mange **External Traffic** Demo: A Blue/Green 6m Deployment

https://is.gd/itined

Summary



Understanding Kubernetes

- Container managment
- Standardized distribution
- API for app modelling

Cluster Management

- Managed cloud services
- Datacenter deployment

Kubernetes API Abstractions

- Pods, ReplicaSets, Deployments
- Services and Pod IP addresses
- Many more storage, stateful apps

Up Next: Using Cloud Container Services