

# Introduction to Kubernetes

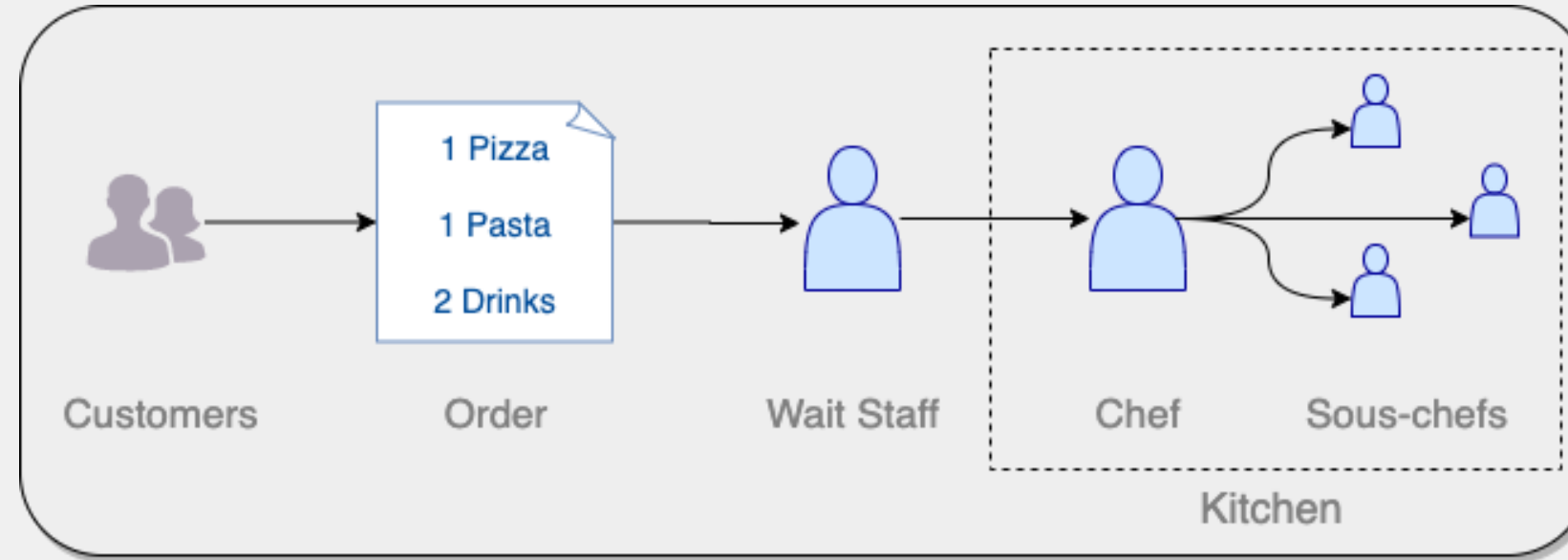
Juan Peredo

<https://www.linkedin.com/in/juanperedotech>

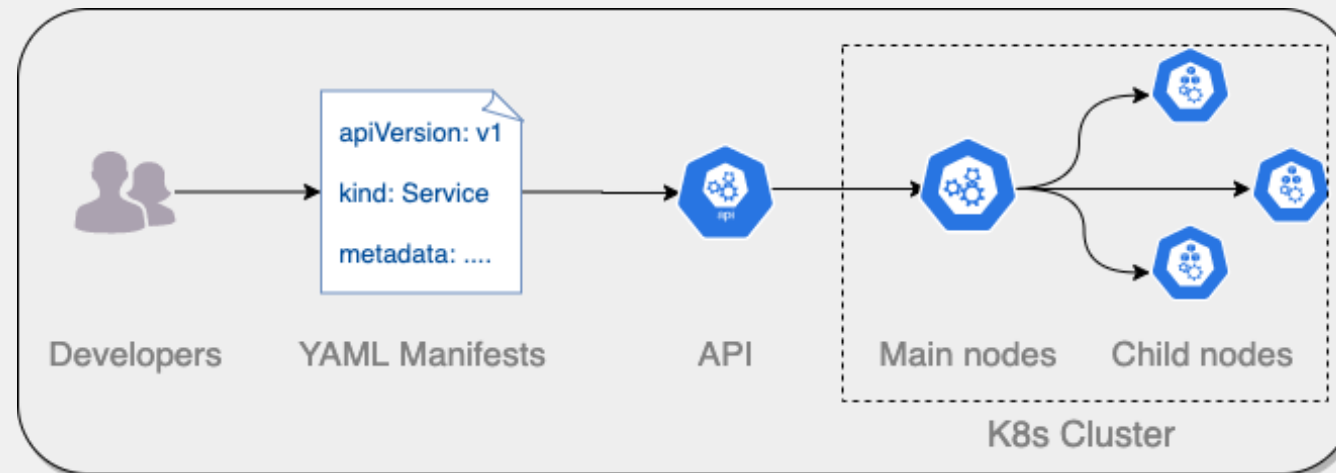
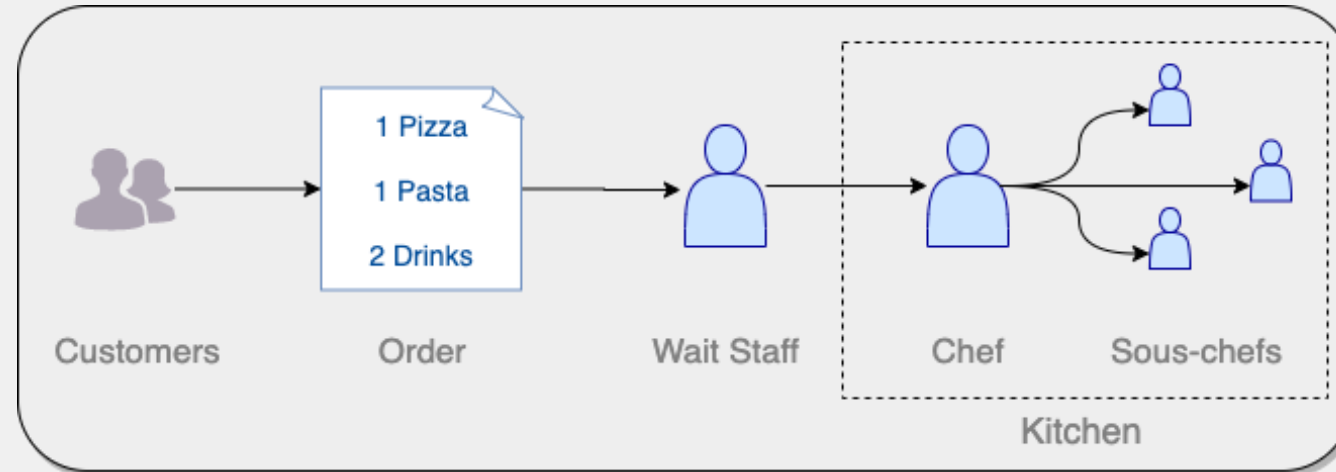
# A couple walks into a bar...



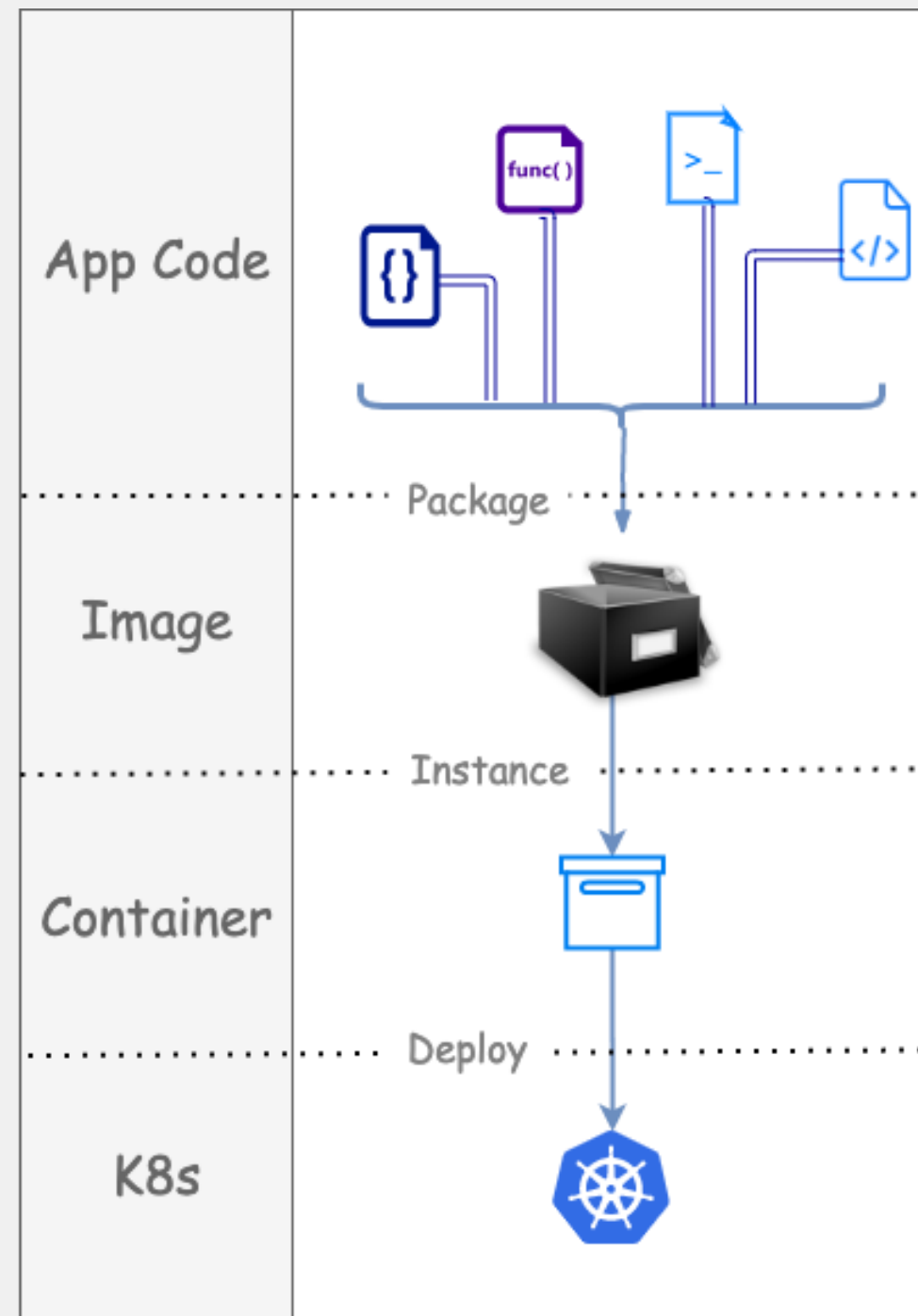
# The food ordering process



# Kubernetes



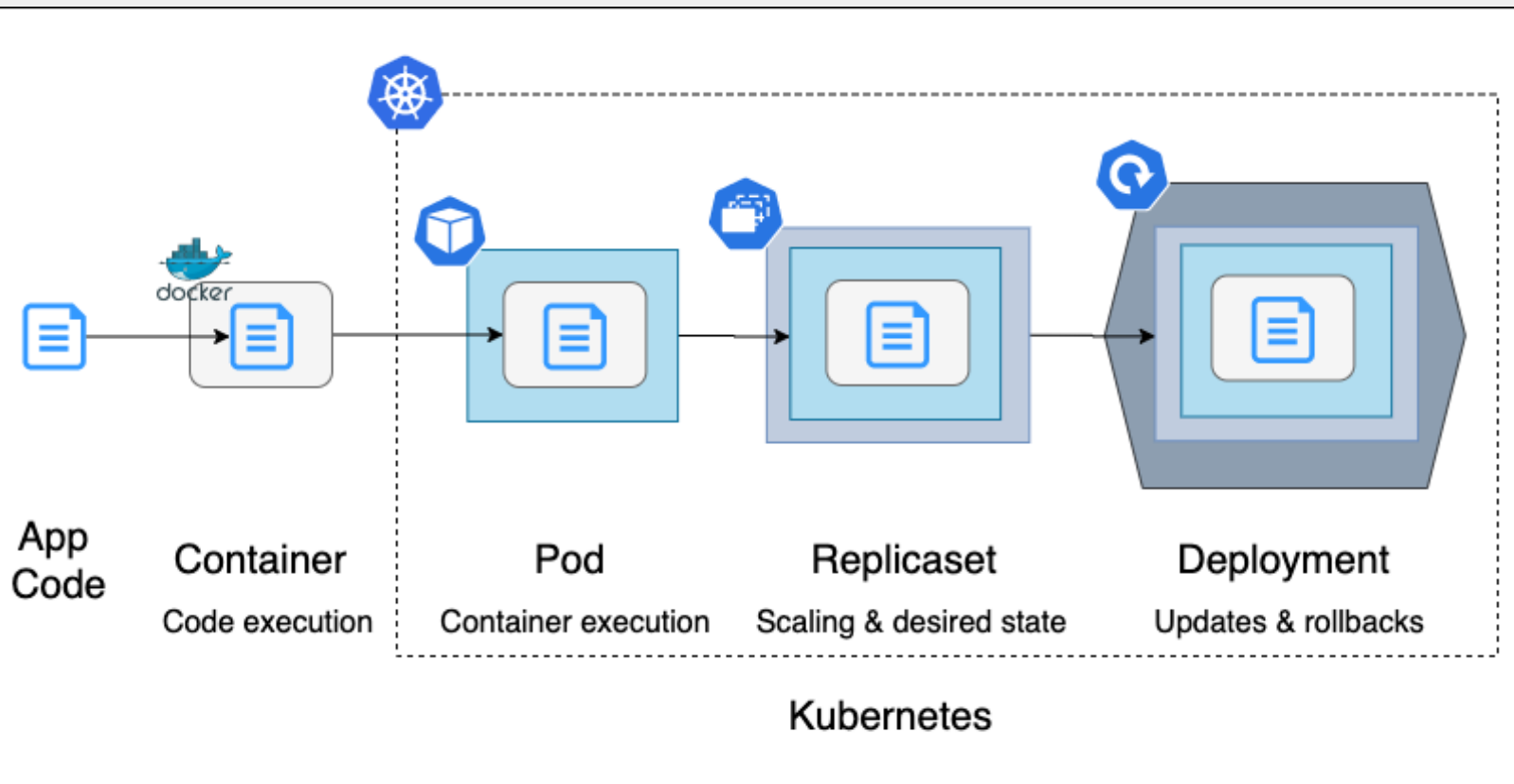
# Images and containers



- Images package apps and all their dependencies
- Images can be shared using container registries like Docker Hub
- Containers are ephemeral image instances
- Kubernetes manages container lifecycle automatically

# Moving to Kubernetes

- Pods: Smallest deployment unit
- Replicaset: Controls number of pod instances
- Deployment: Where all the magic happens

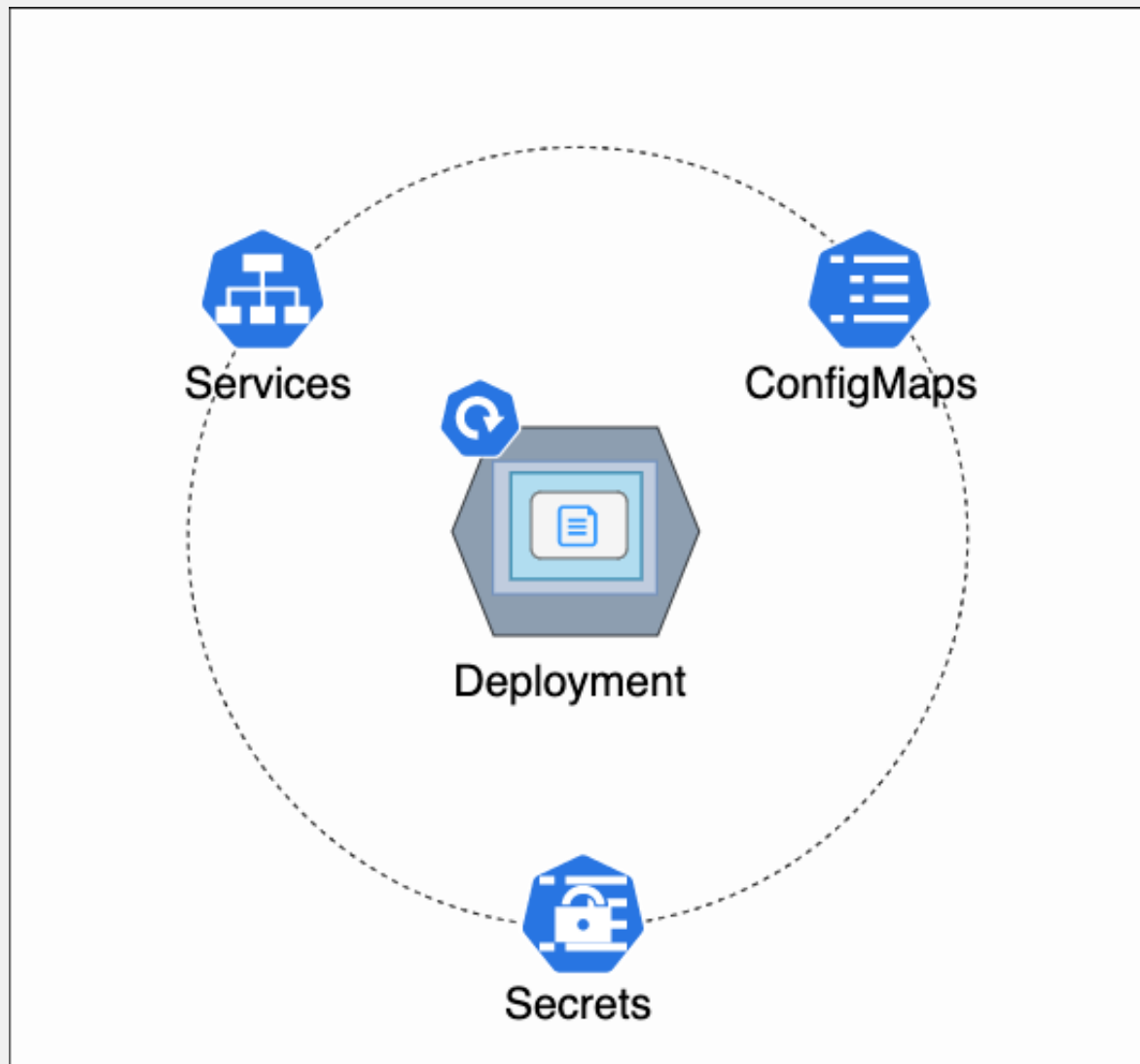


# Adding dessert












Satellite resources helpful when running an app:

- Services: Port definition
- ConfigMaps: App Configuration
- Secrets: Sensitive data (64bit encoded)
- And many more ...

All resources defined using YAML manifests



# Getting started

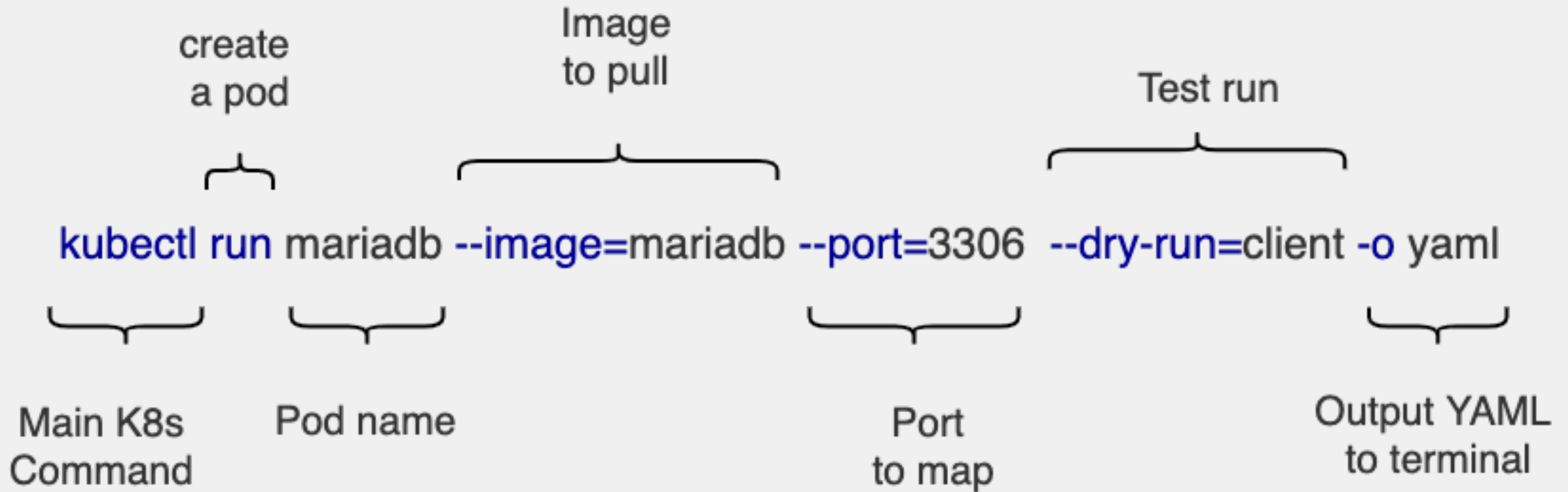
 <p>Local</p>	 <b>minikube</b> <ul style="list-style-type: none"><li>- Official release</li><li>- Latest version</li></ul>  <b>docker</b> <ul style="list-style-type: none"><li>- Included with Docker desktop</li><li>- A couple of versions behind</li></ul>	 <b>K3S</b> <ul style="list-style-type: none"><li>- Lightweight</li><li>- IOT, ARM, Edge</li></ul>  <b>kind</b> <ul style="list-style-type: none"><li>- K8s in Docker</li><li>- Multiple Nodes</li></ul>
 <p>Cloud</p>	 <p>Google Cloud</p>  <b>aws</b>  <b>Azure</b> <p>Free trials</p>	 <b>O'REILLY<sup>®</sup> Katacoda</b>  <b>Play with Docker</b> <p>Free playgrounds</p>



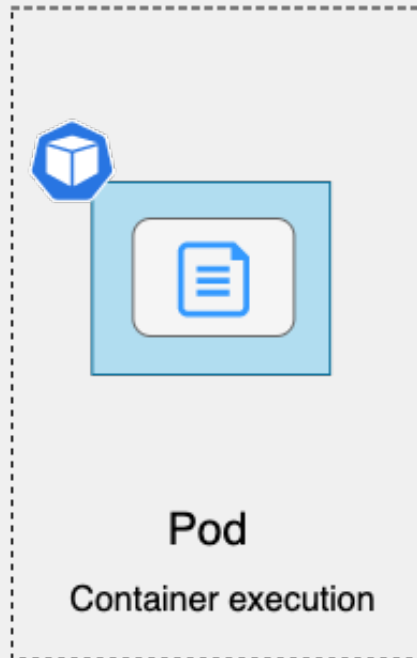
# Basic anatomy of a YAML manifest

```
apiVersion: v1
kind: [Service|Pod|Deployment|etc...]
metadata:
  annotations:
    foo: bar
  labels:
    app: mylabel
  name: myname
[spec|data]:
```

# Our first command

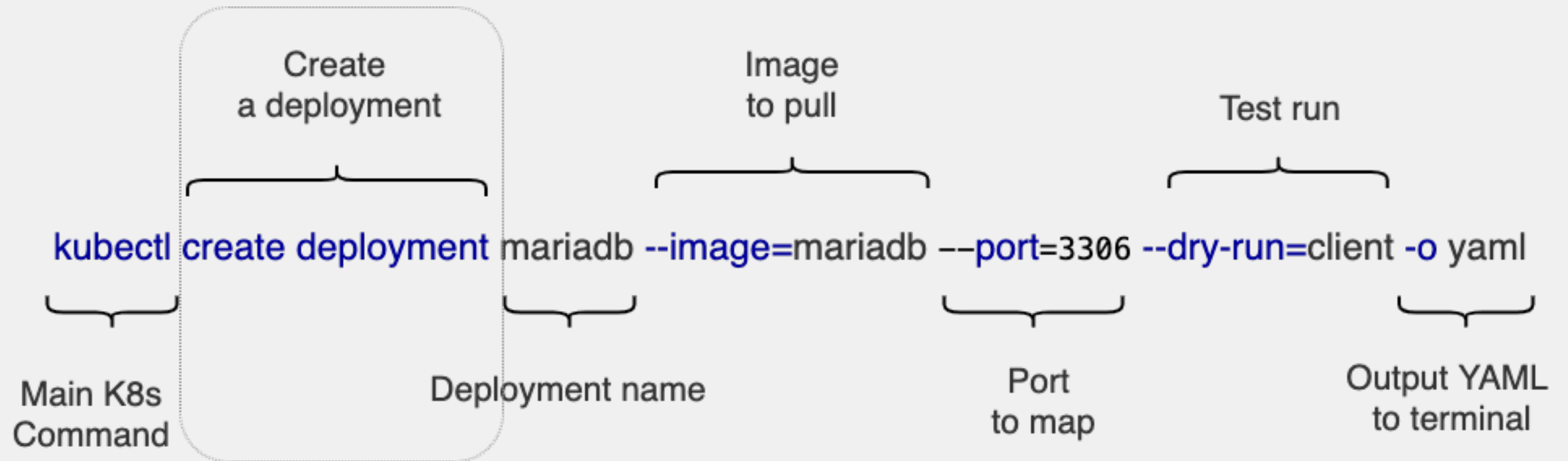


## Our First Pod

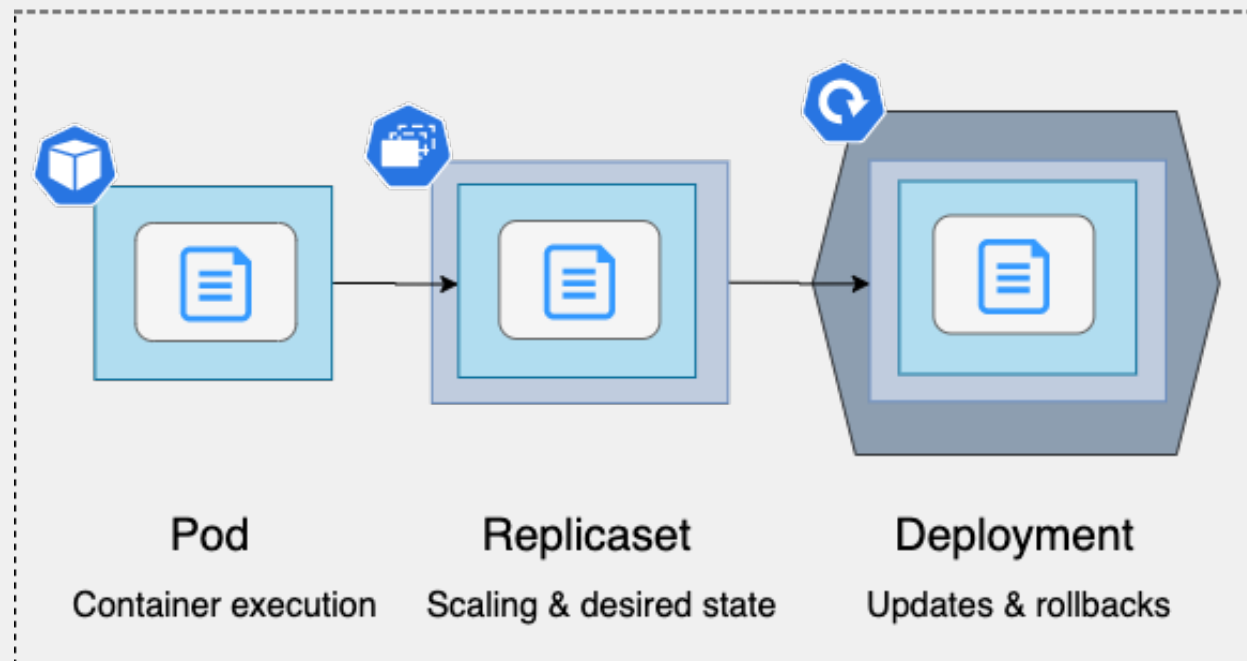


```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: mariadb
  name: mariadb
spec:
  containers:
  - image: mariadb
    name: mariadb
    ports:
    - containerPort: 3306
  ...
```

# Our second command



# Our First Deployment
















```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: mariadb
  name: mariadb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mariadb
  template:
    metadata:
      labels:
        app: mariadb
    spec:
      containers:
      - image: mariadb
        name: mariadb
        ports:
        - containerPort: 3306 ...
```

# Demo time

- Goals
  - Take an application all the way to deployment in K8s
  - Everything as code
- Repo:
  - [https://bitbucket.org/Bolbeck/cwit\\_2020](https://bitbucket.org/Bolbeck/cwit_2020)

```
57 t.appeared = false;
58 return;
59 }
60 //is the element inside the visible window?
61 var a = w.scrollLeft();
62 var b = w.scrollTop();
63 var o = t.offset();
64 var x = o.left;
65 var y = o.top;
66
67 var ax = settings.accX;
68 var ay = settings.accY;
69 var th = t.height();
70 var wh = w.height();
71 var tw = t.width();
72 var ww = w.width();
73
74 if (y + th + ay >= b &&
75     y <= b + wh + ay &&
76     x + tw + ax >= a &&
77     x <= a + ww + ax) {
78     //trigger the custom event
79     if (!t.appeared) t.trigger('appear', settings.data);
80 } else {
81     //it scrolled out of view
82     t.appeared = false;
83 }
84 };
85
86 //create a modified fn with some additional logic
87 var modifiedFn = function() {
88     //mark the element as visible
89     t.appeared = true;
90
91     //is this supposed to happen only once?
92     if (settings.one) {
93         //remove the check
94         w.unbind('scroll', check);
95         var i = $.inArray(check, $.fn.appear.checks);
96         if (i >= 0) $.fn.appear.checks.splice(i, 1);
97     }
98 }
```

# Demo roadmap

 App	   
 Write	 YAML Manifests
 Deploy	  minikube
 Sidecar	
 Multi Environment	  Dev Test

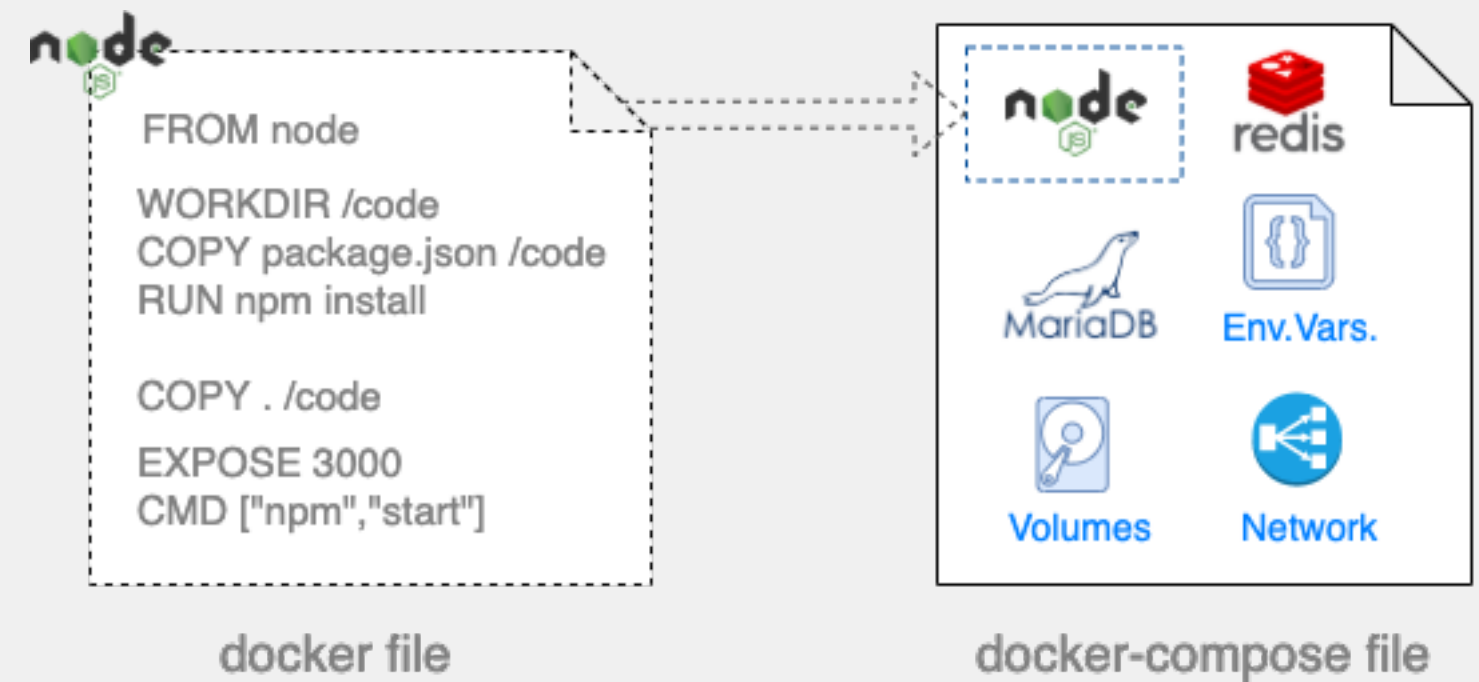
# Our application







# Running our app

- Build custom Node app image with Docker
- Run with Docker-compose
  - Contains everything needed for our app
  - Brings up our containers
  - Creates the appropriate network
  - Checked in with the rest of our code

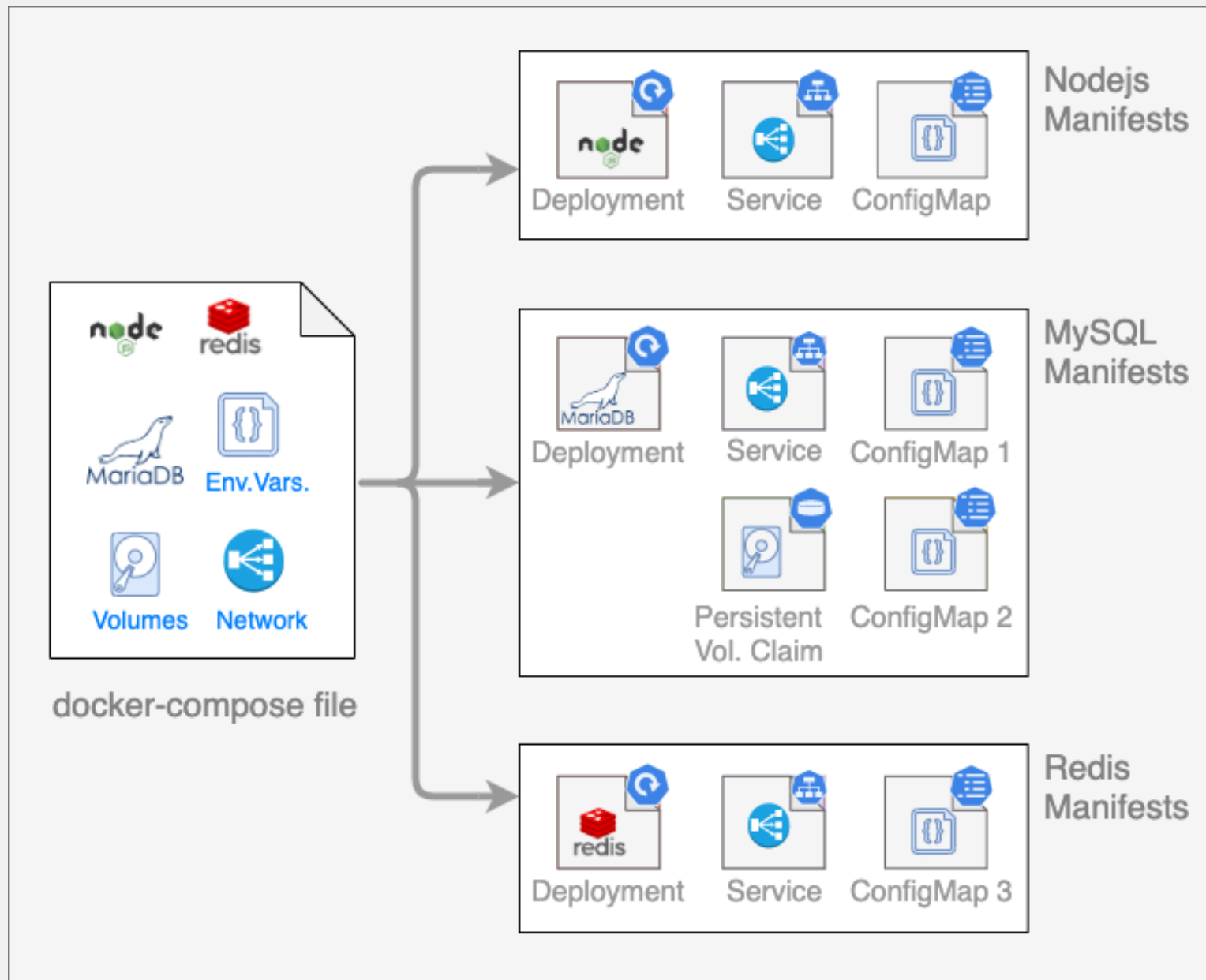


# Demo roadmap

 App	
 Write	 YAML Manifests
 Deploy	 Kubernetes minikube
 Sidecar	
 Multi Environment	 Dev Test

- ✓ Run app with Docker
- ➔ Write Kubernetes manifests
- ➔ Deploy to Kubernetes in Minikube
- Sidecar deployments
- Multi-environment manifests

# Kubernetes manifests

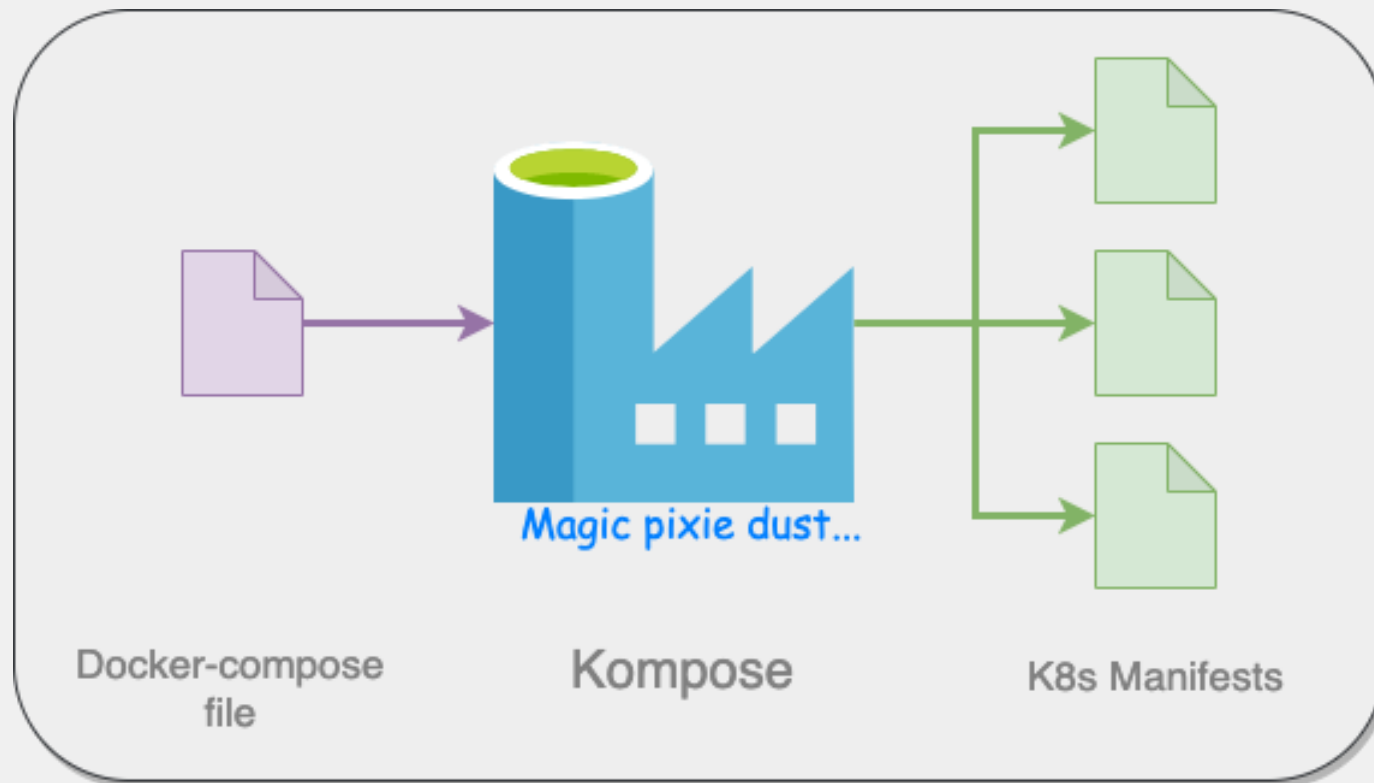


- Docker-compose defined our app with 3 services
- For K8s, we need to map those to multiple resource manifests:
  - Deployments
  - Services
  - Persistent Volumes
  - ConfigMaps

# Manual manifest creation

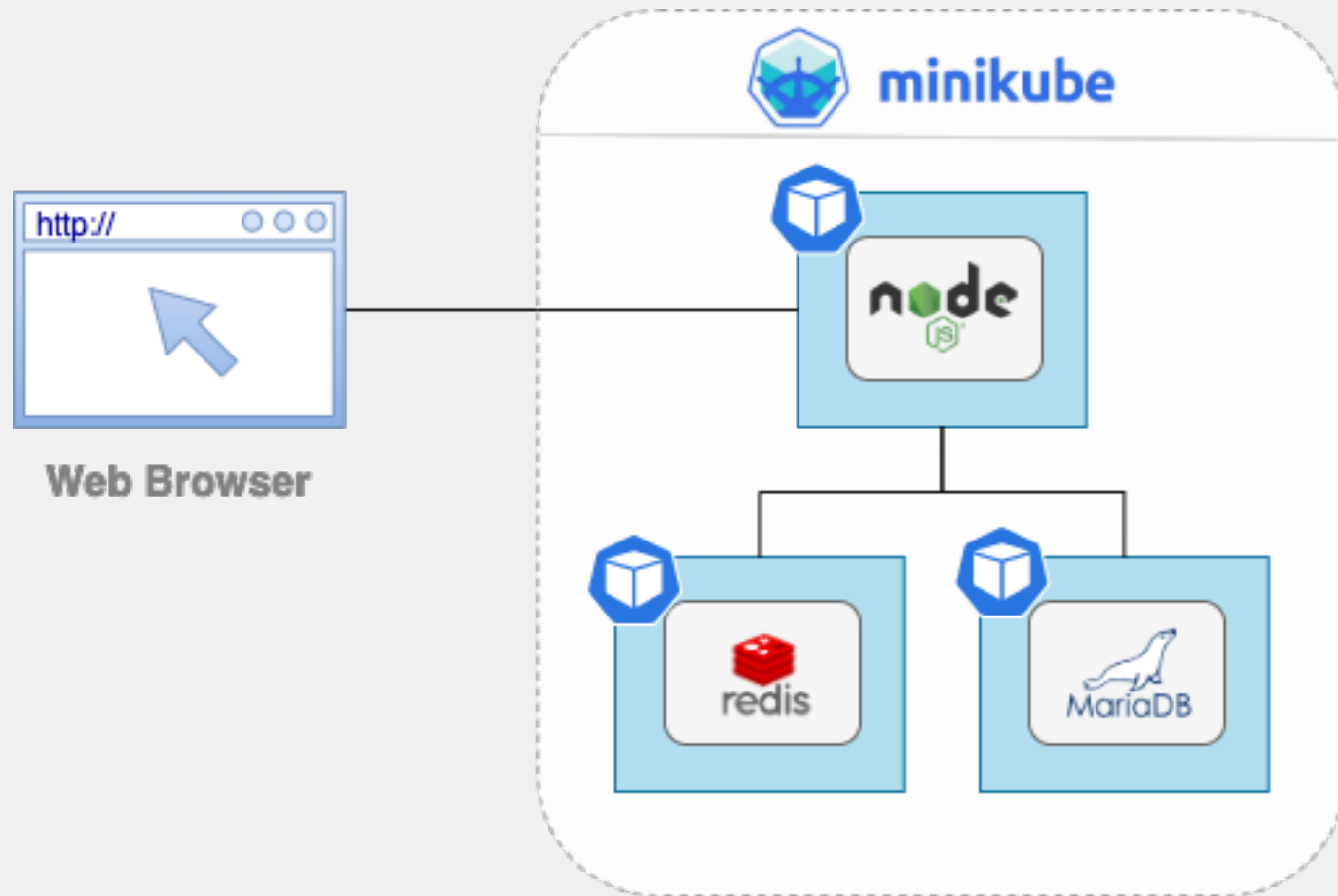


# Generating Kubernetes manifests



- Kompose generates manifests automagically! 😎
- Based on dockercompose file
- Out of the box, generation gets you 90% there
- Run: `kompose -f <path_to_dockercompose_file> convert`
- Get Kompose at [kompose.io](https://kompose.io)

# Our app in K8s



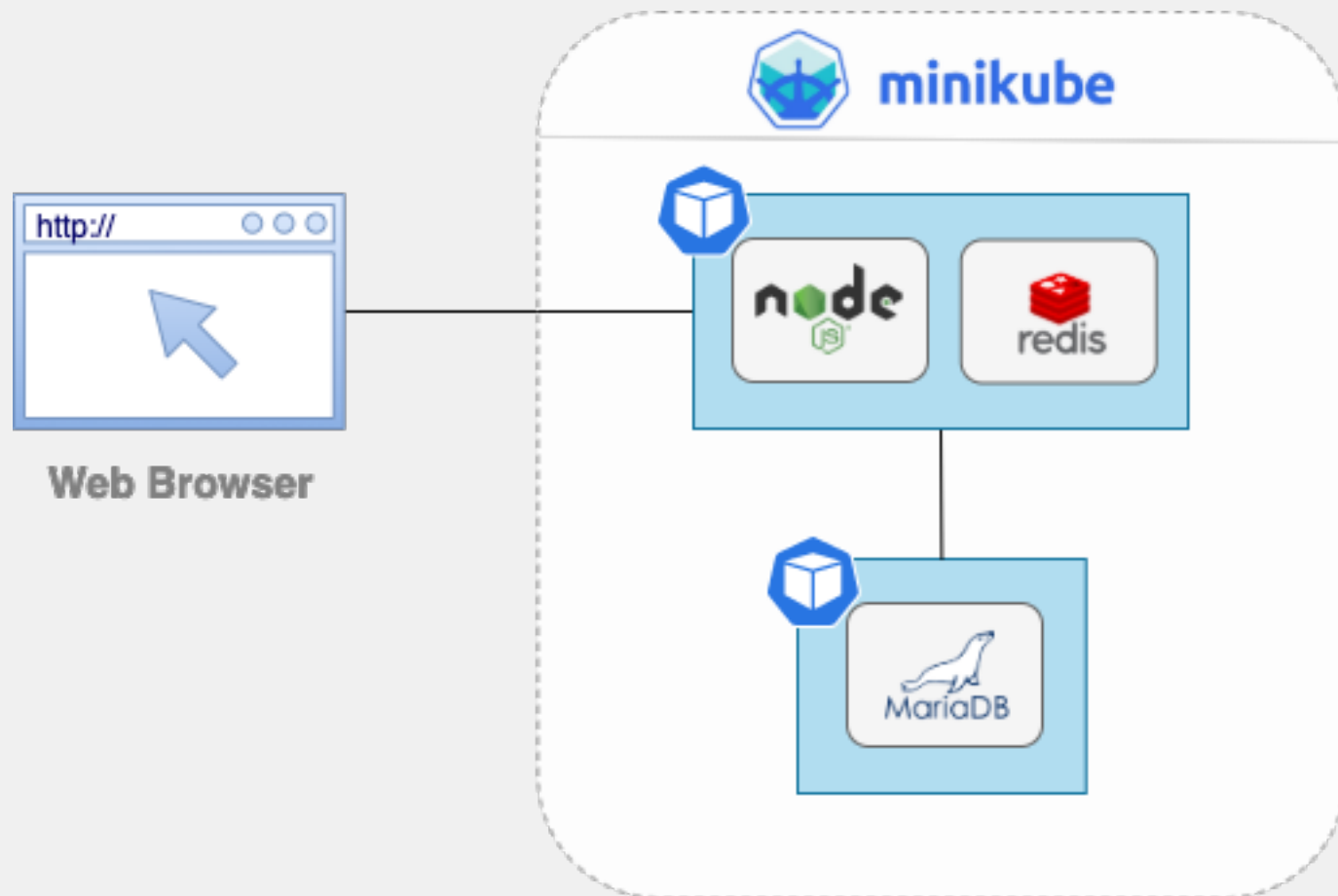
- Application running:
  - One container per Pod
  - Network communication
  - Accessible via browser
  - Scaled up and down
- Is that where we want our Redis cache ?

# Demo roadmap

 App	
 Write	 YAML Manifests
 Deploy	 Kubernetes minikube
 Sidecar	
 Multi Environment	 Dev Test

- ✓ Run app with Docker
- ✓ Write Kubernetes manifests
- ✓ Deploy to Kubernetes in Minikube
- ➔ Sidecar deployments
- Multi-environment manifests


# Caching as a sidecar



- Sidecar: provides additional functionality to main container
- Advantage: Treated as a local resource
- Disadvantage: Containers cannot be individually managed
- What we'll do:
  - Remove our Redis deployment & service
  - Add Redis to our Node deployment
  - Change how Node discovers Redis



# Demo roadmap

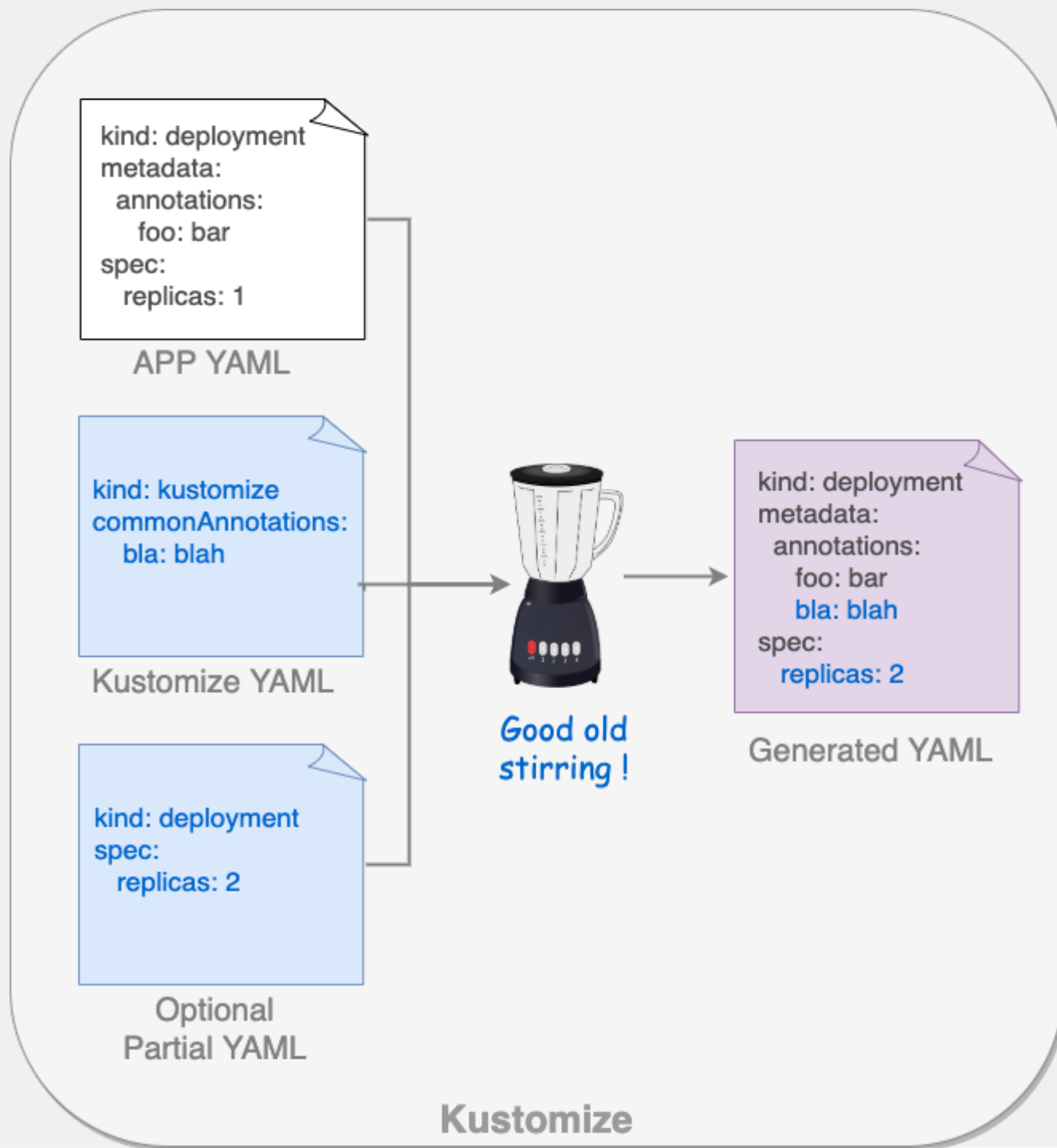
 App	
 Write	 YAML Manifests
 Deploy	 Kubernetes minikube
 Sidecar	
 Multi Environment	 Dev Test

- ✓ Run app with Docker
- ✓ Write Kubernetes manifests
- ✓ Deploy to Kubernetes in Minikube
- ✓ Sidecar deployments
- ➔ Multi-environment manifests

# Maintain manifest copies















# Kustomize



- YAML customization
- Merges YAML files
- Eases multi-configuration manifests maintenance
- Usage, either:
  - Built-in as `kubectl apply -k` and `kubectl kustomize`
  - Latest binary from [Kustomize.io](https://kustomize.io)

# Demo roadmap

 App	
 Write	 YAML Manifests
 Deploy	
 Sidecar	
 Secure	
 Multi Environment	


- ✓ Run app with Docker
- ✓ Write Kubernetes manifests
- ✓ Deploy to Kubernetes in Minikube
- ✓ Sidecar deployments
- ✓ Using secrets
- ✓ Multi-environment manifests


# K8s resources at a glance


## K8s Resources


 Container deployment using Pods


 Replicasets for scaling


 Pod management using Deployments


 Services for port definition


 Ingress for external access and proxy


 Application isolation via namespaces


 Storage with Persistent Volumes

 Jobs to run scheduled tasks

 K8s extensions with custom resources

 Role based security with RBAC

 Secrets for sensitive information

 Cronjob for scheduling

And so many more ....

# Rich ecosystem

The image displays a grid of 16 categories of Kubernetes ecosystem tools, each with its own icon and logo. The categories are arranged in a 4x2 grid:

- Manage:** Kompose, HELM, Kustomize, OperatorHub
- Secure:** SealedSecrets, Vault, Open Policy Agent, aqua
- Deploy:** Jenkins X, SKAFFOLD, Tekton, argo
- Test:** chaoskube, Telepresence, SONOBUOY, kube-monkey
- Diagnose:** Grafana, Prometheus, DATADOG, Kubewatch
- Network:** linkerd, Istio, Consul, CoreDNS
- Providers:** Google Cloud, Linode, aws, Azure
- Infrastructure management:** Terraform, Anthos, RANCHER, pulum

# Key takeaways



# Questions ?



Thanks to the CWIT Conference & all the sponsors





# Appendix

# Photos

-  Photo by [DarkmoonArt\\_de](#) at [Pixabay](#)
-  Photo by [Markus Spiske](#) on [Unsplash](#)
-  Photo by [Wiktor Karkocha](#) on [Unsplash](#)

# Photos

-  Photo by [Daniel Cheung](#) on [Unsplash](#)
-  Photo by [Marcos Paulo Prado](#) on [Unsplash](#)