Kubernetes? Easy!

Nicolas Perraut - @mafzst

Kubernetes?

- "Platform for automating deployment, scaling, and operations of application containers across clusters of hosts"
- Backed by Google & Cloud Native Computing Foundation
- Mainly used with Docker containers but supports also RKT and others

Kubernetes?

- Cluster = 1 master and n nodes (workers)
- All can be done with `kubectl` cli & YAML files
- A REST API is available (inside and outside of the cluster)

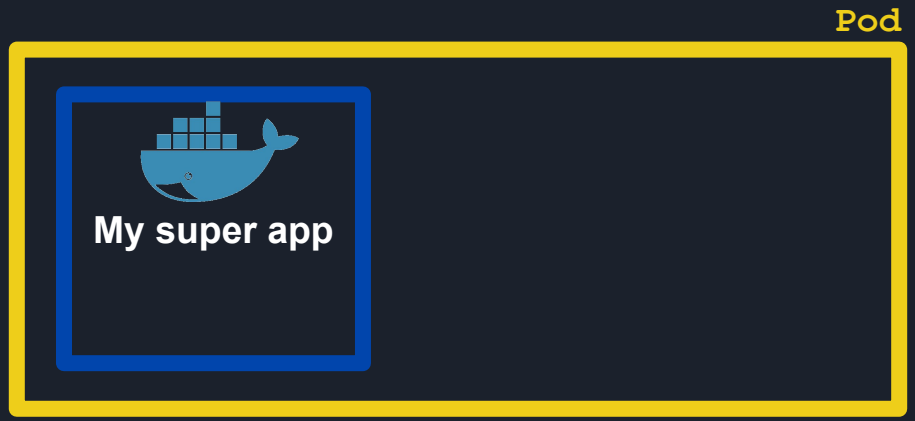
Easy!

Let’s start with a simple Docker container

Easy!

Let's start wrap it with a `Pod`

- **1 application** container
- n sidecars (proxy, loggers, backups, updater,...)
- 1 unique IP for the whole

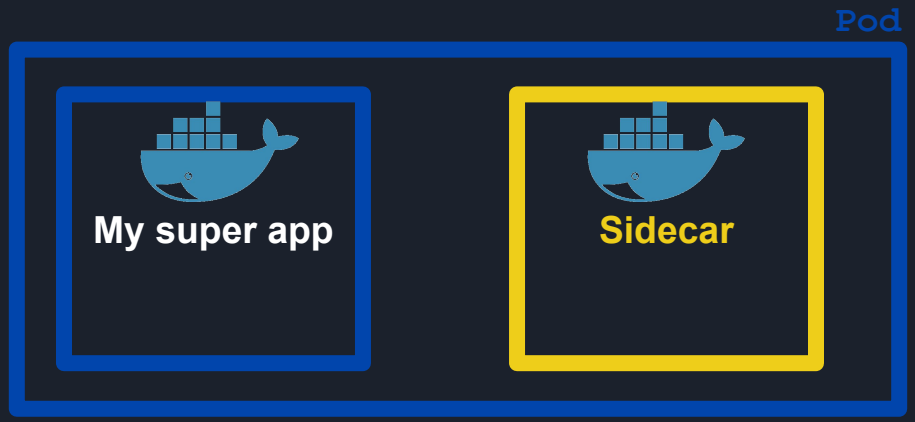


`Pods` are the smallest deployable units of computing that can be created and managed.

Easy!

Let’s add a sidecar

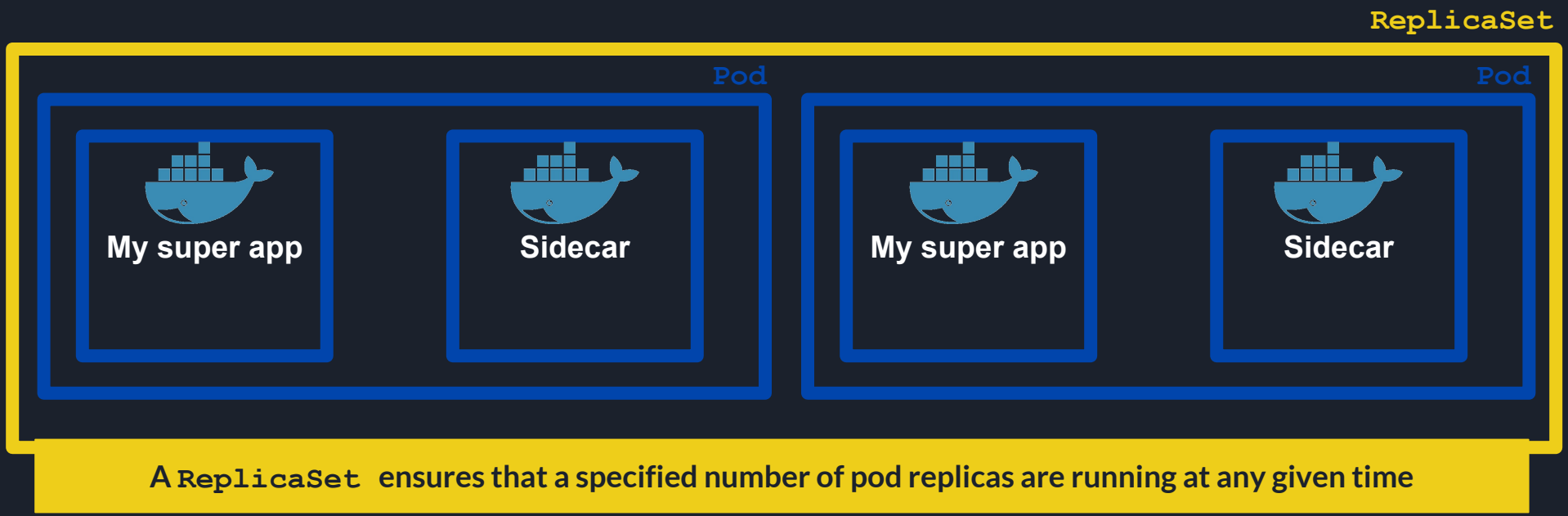
- **1 application** container
- n sidecars (proxy, loggers, backups, updater,...)
- 1 unique IP for the whole



Pods are the smallest deployable units of computing that can be created and managed.

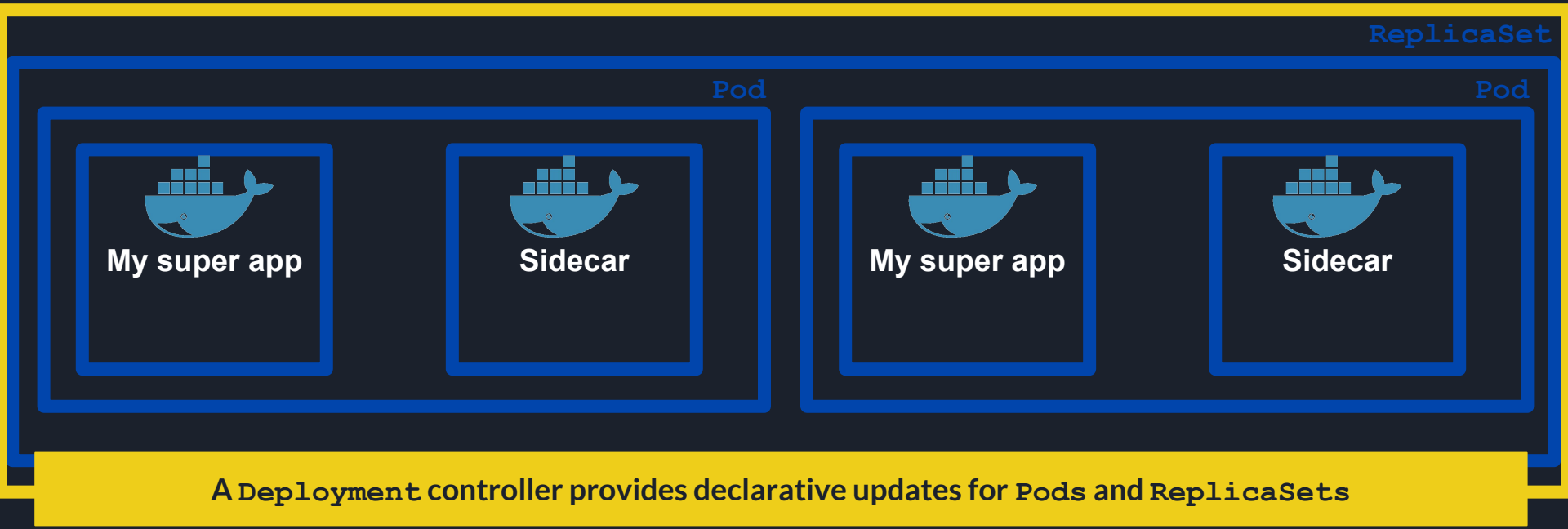
Easy!

Let's manage this with a **`Replica Set`**



Easy!

And now deploy it with a... `Deployment`!



A `Deployment` controller provides declarative updates for `Pods` and `ReplicaSets`

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: my-super-app
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: myapp:1.0.12
      - name: proxy
        image: nginx:1.7.9
        ports:
        - containerPort:8000
```

Deployment

ReplicatSet

Pod

Containers

Easy!

Let’s do some networking stuff!

3 network types:

- Pod network: Intra node network. Pods have IPs in it
- Cluster network: Shared between node. For services
- Outside world network

Easy!

Let's do some networking stuff!

3 network types:

- Pod network: Intra node network. Pods have IPs in it
- Cluster network: Shared between node. For services
- Outside world network

Pods CAN communicate with others on the SAME node but CANNOT with others on OTHER node

Easy!

Let's do some networking stuff!

3 network types:

- Pod network: Intra node network. Pods have IPs in it
- Cluster network: Shared between node. For services
- Outside world network

Pods CAN communicate with others on the SAME node but CANNOT with others on OTHER node

Easy!

Services

- Unique endpoint for all selected pods
- An unique IP in the cluster network
- Can act as load balancer

A `Service` is an abstraction which defines a logical set of `Pods` and a policy by which to access them

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: my-super-app
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: myapp:1.0.12
      - name: proxy
        image: nginx:1.7.9
        ports:
        - containerPort: 8000
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
```

Easy!

Let's connect to the wild outside world!

- Ingresses expose services to the outside
- It's like a Vhost or an Nginx server file
- An Ingress manager or an external load balancer is needed

An `Ingress` is a collection of rules that allow inbound connections to reach the cluster `Services`.

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: s1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: s2
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: s3
          servicePort: 80
```

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: my-super-app
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: myapp:1.0.12
      - name: proxy
        image: nginx:1.7.9
        ports:
        - containerPort: 8000
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-super-app
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: my-service
          servicePort: 80
```

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: my-super-app
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: myapp:1.0.12
      - name: proxy
        image: nginx:1.7.9
        ports:
        - containerPort: 8000
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-super-app
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: my-service
          servicePort: 80
```