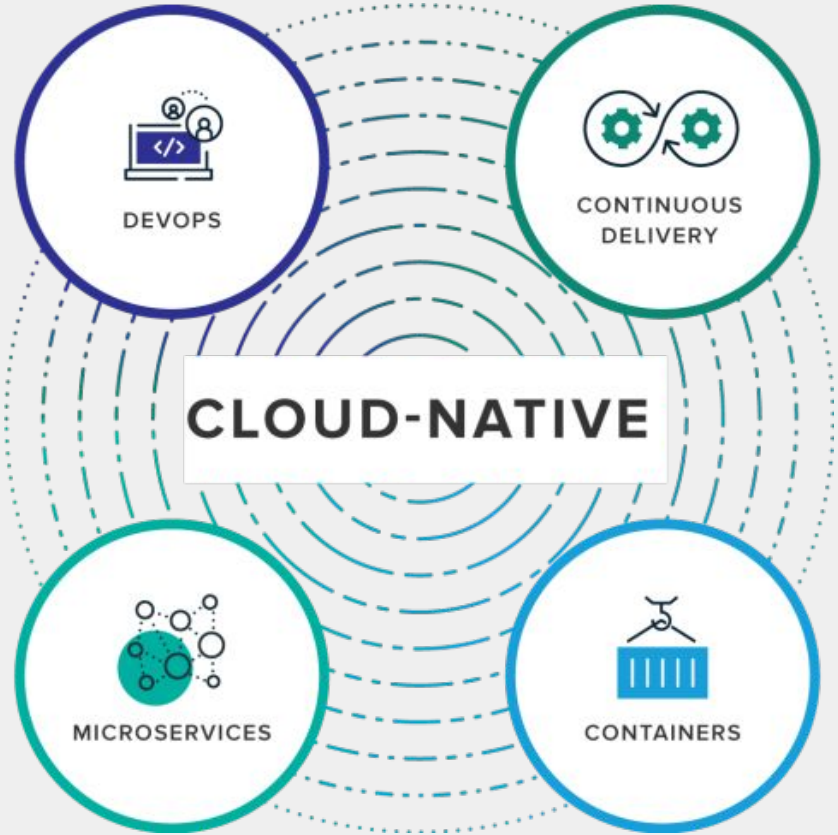
Cloud Native Application Development on Kubernetes

Cemal Ünal

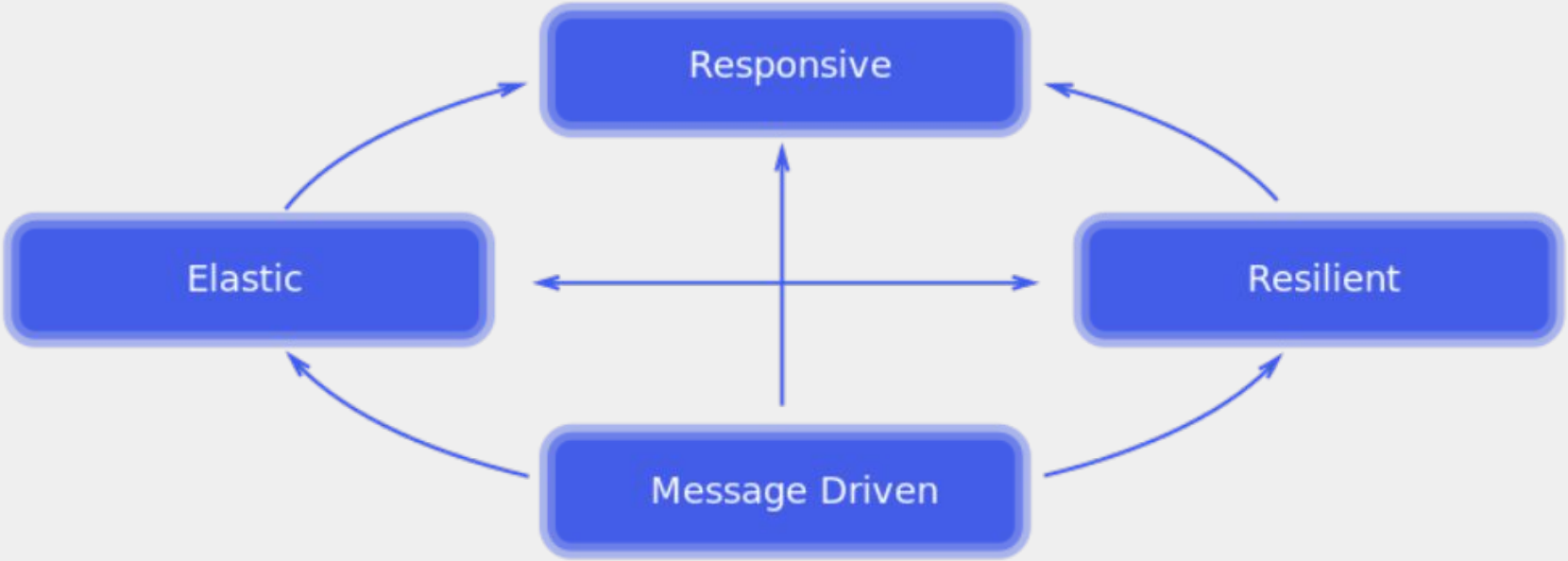
- Software Engineer @ Havelsan Inc.
  - Cloud Native Application Development / Delivery / Monitoring

- cemalunal@yahoo.com
- https://www.linkedin.com/in/cemalunal/

Cloud Native



Reactive Manifesto

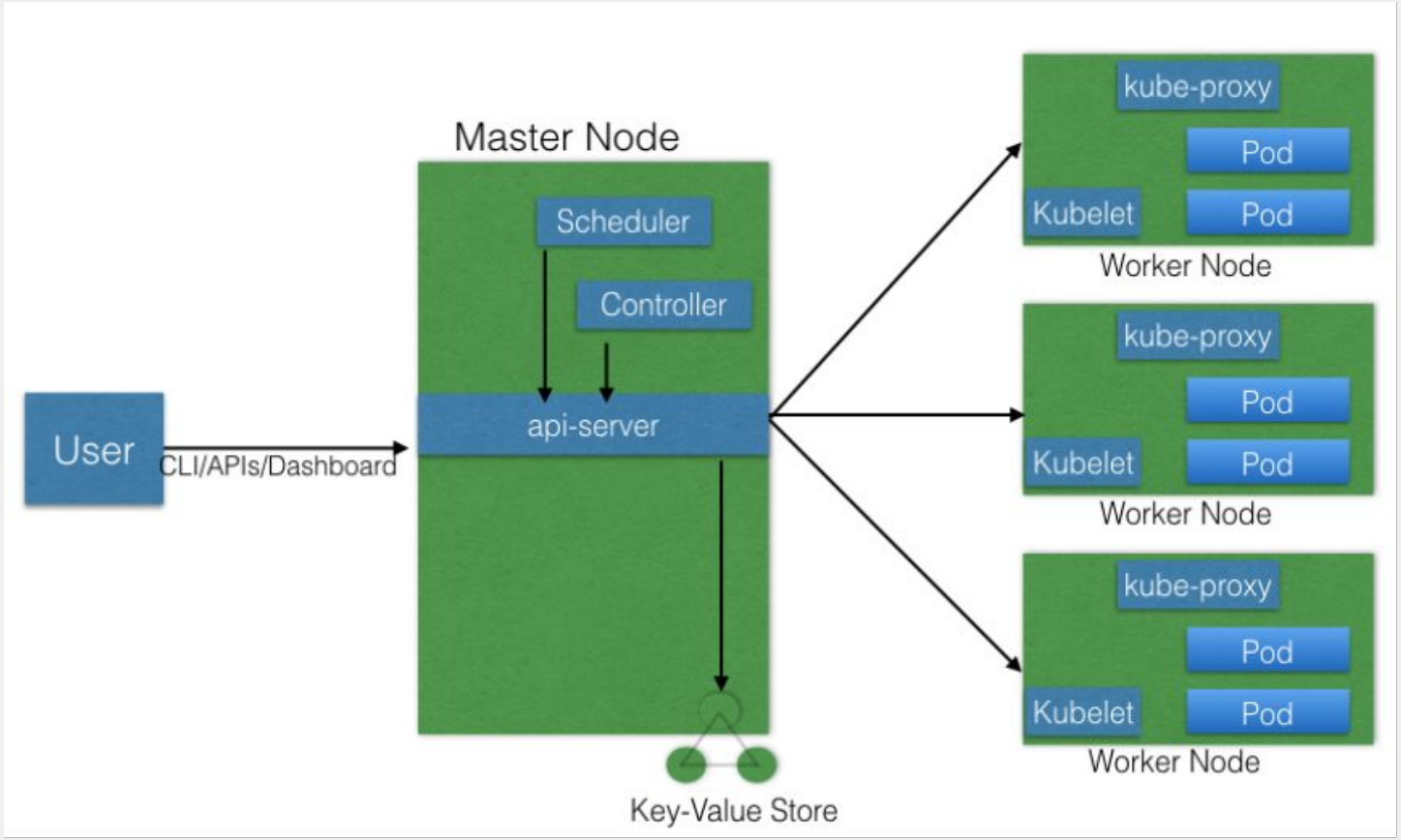


Container Orchestration

Requirements:

- Fault-tolerance
- On-demand scalability
- Optimal resource usage
- Self healing
- Highly available in case of any failures
- Auto-discovery to automatically discover and communicate with each other
- Seamless updates/rollbacks without any downtime

Kubernetes


Master Node
Scheduler
Controller
api-server
User
CLI/APIs/Dashboard
Key-Value Store
kube-proxy
Pod
Kubelet
Pod
Worker Node
kube-proxy
Pod
Kubelet
Pod
Worker Node
kube-proxy
Pod
Kubelet
Pod
Worker Node

Kubernetes Concepts - Pods

- Smallest and simplest Kubernetes object.
- Represents a single instance of the application.
- They do not have the capability to self-heal by themselves. (Use deployment instead)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.17.0
    ports:
    - containerPort: 80
```

Kubernetes Concepts - Deployment

- Makes sure that the current running number of pods always matches the desired state.
- Provide declarative updates to Pods.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx:1.16.0
        name: nginx
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 80
        terminationGracePeriodSeconds: 10
```

Kubernetes Concepts - Namespaces

- Use it to segment the services into manageable chunks.
- Namespaces can make Kubernetes a lot more manageable and gives us increased control, security and flexibility.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: elasticsearch-cluster
```

Kubernetes Concepts - Services

- Using Services, Pods can be grouped to provide common access points from the external world.

- Service types:
  - Headless
  - ClusterIP
  - NodePort
  - LoadBalancer

```yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx
  namespace: kube-monitoring
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
    - port: 9093
  selector:
    app: nginx
```

Kubernetes Concepts - ConfigMap

- Allow us to decouple the configuration details from the container image
- Using ConfigMaps, we can pass configuration details as key-value pairs, which can be later consumed by Pods

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: frontend-cm
  namespace: k8s-demo
data:
  REACT_APP_BACKEND_URI: "http://10.0.0.0"
```

```yaml
spec:
  containers:
    - image: cunal/demo-frontend:v0.0.1
      imagePullPolicy: IfNotPresent
      envFrom:
      - configMapRef:
          name: simple-frontend-cm
```

Horizontal Pod Autoscaler

- Automatically scales the number of pods in a deployment.
- Autoscale is based on observed CPU utilization or some custom metrics.