Kubernetes için Temel Yetkinlikler

- Linux
- Docker (build)
- Temel Ağlar, DNS
- HTTP
- YAML

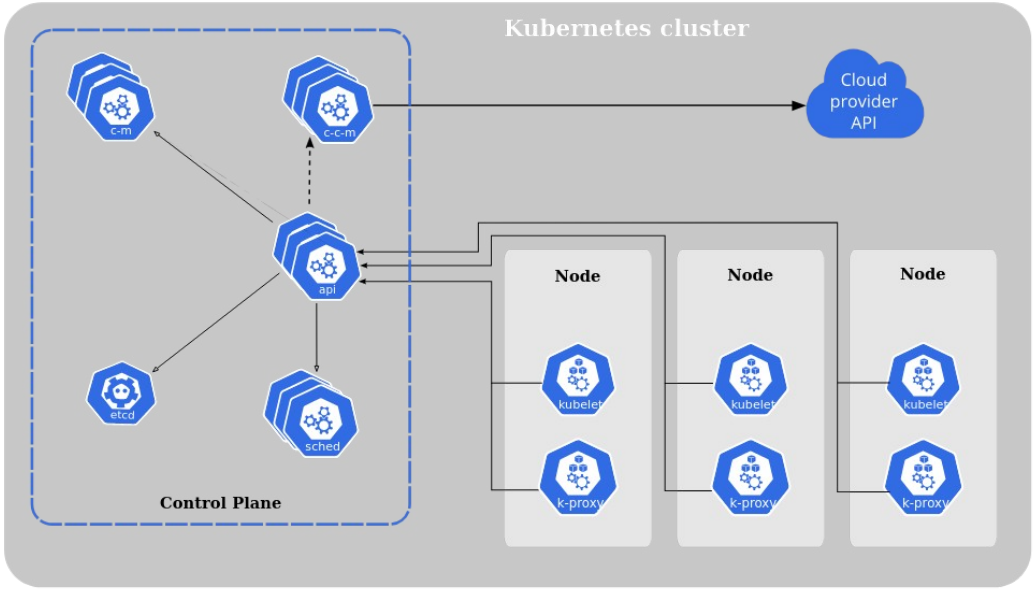
Kubernetes Yapıtaşları

Controlplane (Sadece masterlarda)

- api-server
- controller-manager
- Scheduler
- etcd

Dataplane (Hepsinde var)

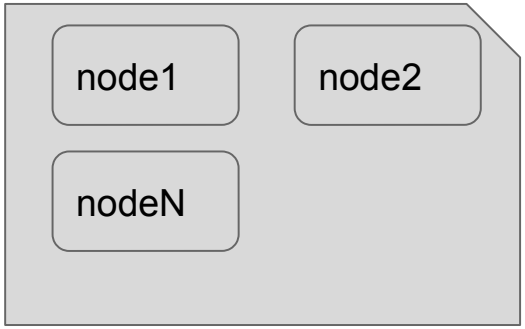
- Kubelet
- kube-proxy
- **DNS - coreDNS**
- **CRI** (Container Runtime Interface) - containerd
- **CNI** (Container Network Interface) - calico



★ Her biri bir/çok servistir. Genellikle container/pod olarak çalışırlar. Nadir sorun yaşanır ve sorun çıkmadığı sürece işimiz olmaz.

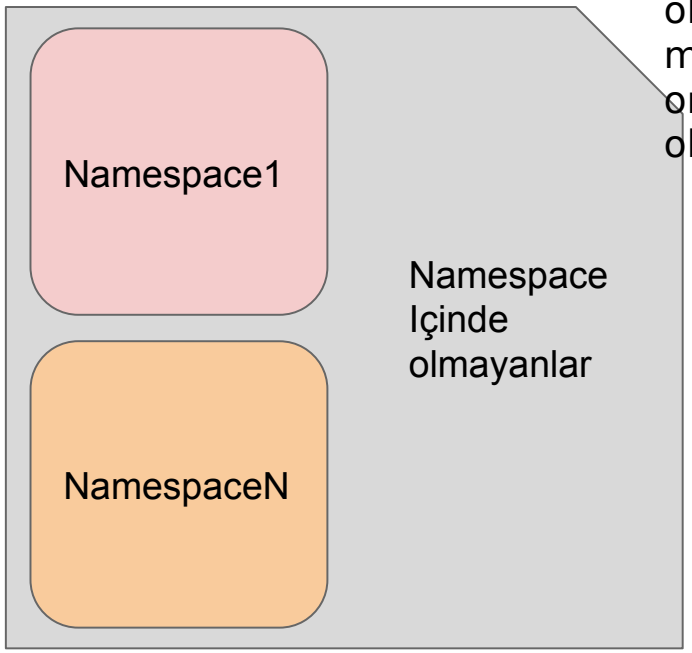
Kubernetes

Küme (Fiziksel)



★ Her biri bir havuzdaki kaynaktan ibarettir. Bizi ilgilendiren kısmı havuzun büyüklüğüdür.

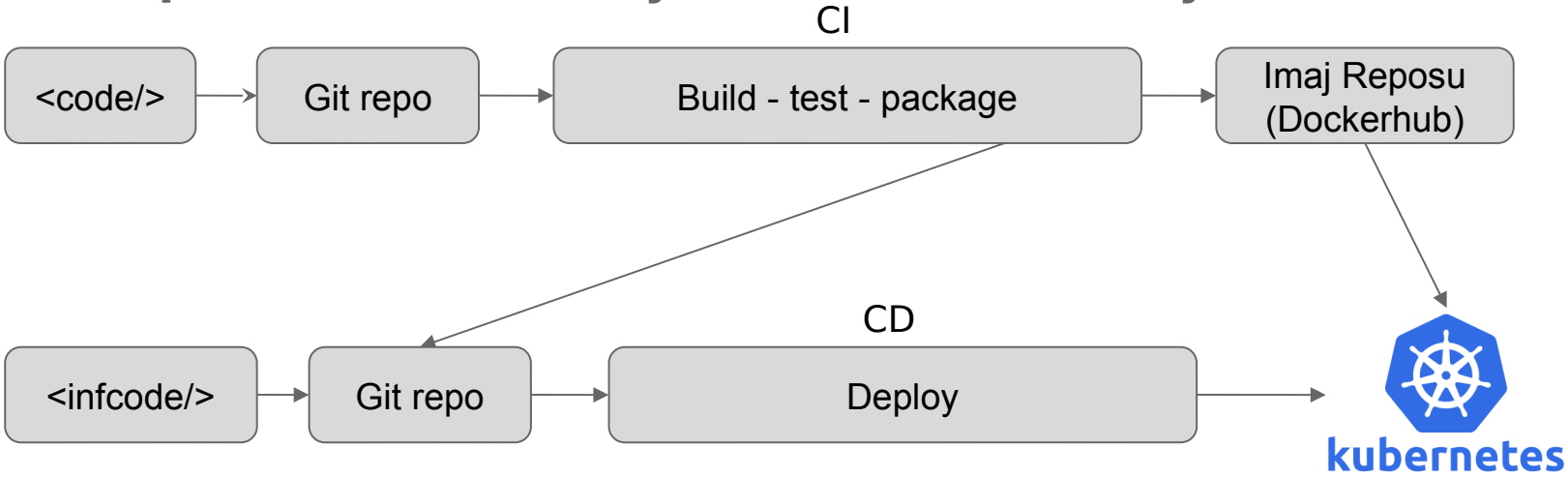
Küme (Mantıksal)



★ İnsanların günlük olarak bu mantıksal ortamla işi olur.

Kubernetes İşyükleri Akışı

- **Gitops (Kurumsal düzeyde kodtan hizmete yol)**



<infcode/> - tek derdimiz

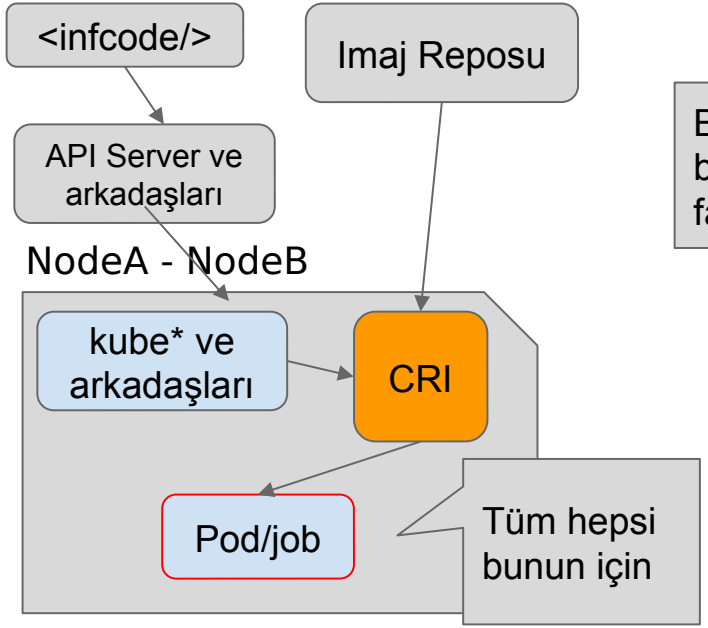
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
##########
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
##########
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service.myspace.cluster-
adi.local:8000
  namespace: myspace
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8000
      TargetPort: 80
```

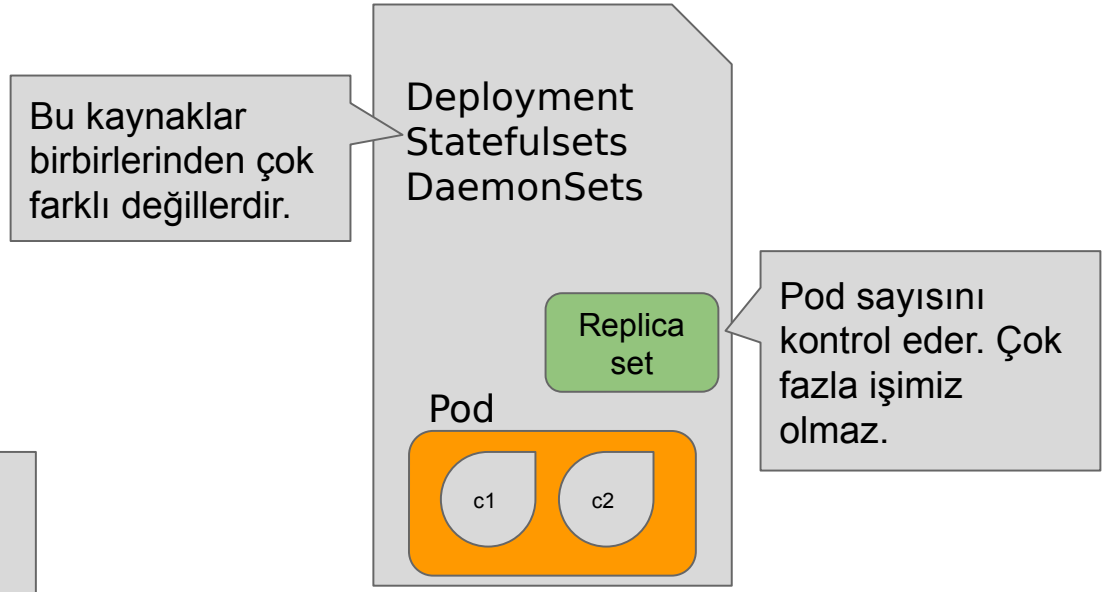
★ Geliştiricilerin ve altyapı yönetenlerin en çok görecekleri altyapı kodları **Deployment** ve **Service**'tir

Kubernetese Hizmetimiz Nasıl Kurulur?

- **Kurulum(Deployment) Bakışı**



- **Mantıksal İlişki Bakışı**



https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

<infcode/> - tek derdimiz

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
##########
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
##########
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

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

Service'in hangi porta trafiği yönlendireceğini söyler.

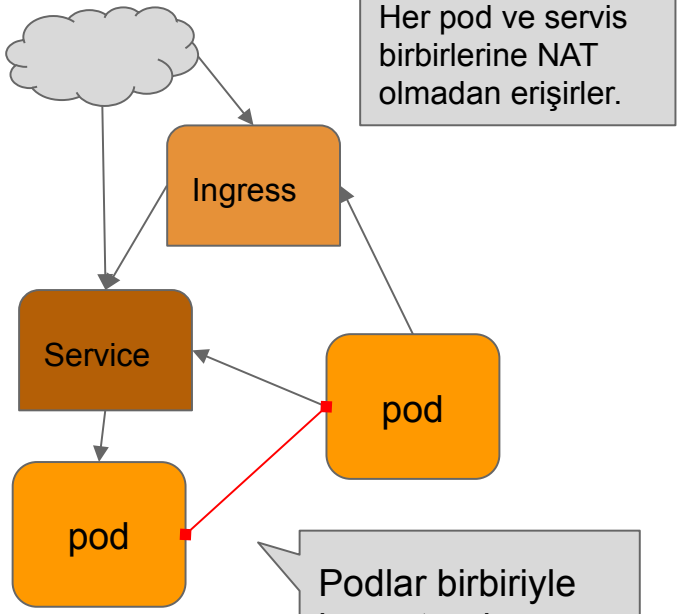
Key: value pod ile aynı olmak zorunda. Yoksa bulamaz.

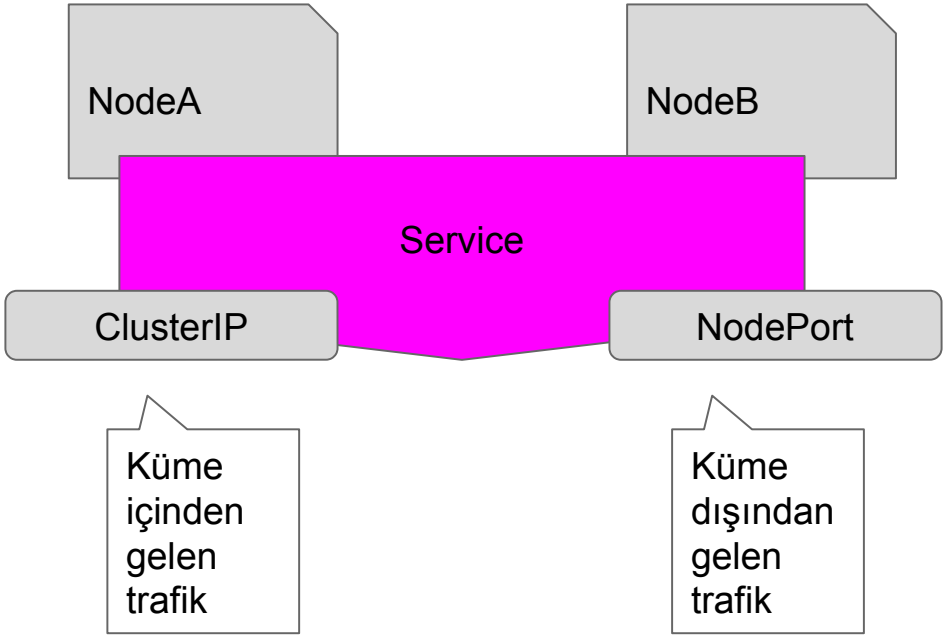
★ Etiket herşeydir.

Kubernetes Nasıl Çalışır?

- **Network Bakışı**
  - **Overlay Network**

Her pod ve servis birbirlerine NAT olmadan erişirler.





Dikkat çekelim

- **Pod dışındaki her şey mantıksaldır.**
- **Ağ işlemleri node kernelde gerçekleşir. (iptables, ipvs)**
- **Kümede DNS hizmeti vardır. Ipleri hatırlamaya gerek yok.**
  - **Service discovery**
  - **Load balancing**

Ortamın ayarları ortamda kalır! 12factor.net

- Configmaps
- Secrets

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: myconfigmap
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # file-like keys
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    configMap:
      name: myconfigmap
```

Bir Uygulamanın Temel İhtiyaçları

- İhtiyaç Kadar Büyük Olmalı
- Hızlı Hizmete Alınabilmeli
- Geri Alınabilmeli
- İzlenmeli
- Otomatik Kurtarılmalı
- Otomatik Ölçeklenmeli
- Ağ Erişimi Sabit Olmalı
- Kalıcı Diske Yazmalı
- Kaynakları Sınırlanmalı

- Container / Pod
- Deployment
- Replication Set
- HealthChecks
- Deployment
- Autoscaler / HPA
- Service / Nodeport / DNS
- Volumes
- Resources / Limits

Uygulamanın Ek İhtiyaçları

- Çok ekip aynı ortam üzerinde çalışabilmeli - **NAMESPACES**
- Ayarları Kodtan ayırabilmeli - **CONFIGMAPS / SECRETS**
- Yetki mekanizmaları olmalı - **RBAC**
- Uygulamaların birbirlerine erişimleri sınırlanmalı - **NETWORKPOLICY**
- Uygulamaların kümeye girişi ve çalışması kurallı olmalı - **POLICY MANAGEMENT, POD SECURITY**
- Hizmet bazında izlenebilmeli - **PROMETHEUS/GRAFANA**
- Servisler arası trafik yönetilebilmeli – **SERVICEMESH (circuit braker, timeouts)**
- Dışarıdan başka araç olmadan erişilebilmeli – **METALLB-INGRESS**
- Başka yeteneklerin eklenmesine izin verilmeli **- CRD**

★ Altçizgili olanlar kuberneteste varsayılan olarak geliyor.