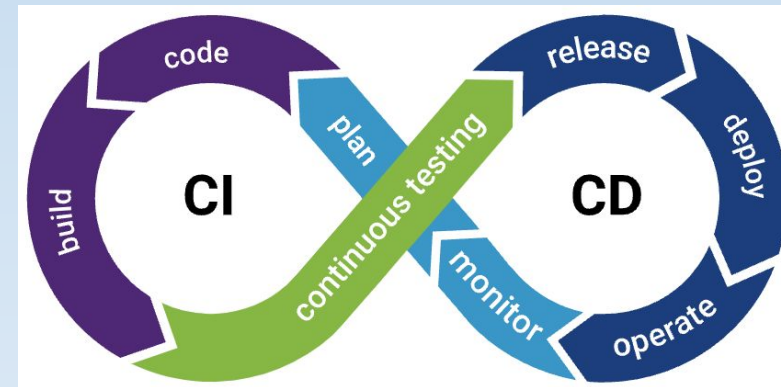




DevOps Atölyesi

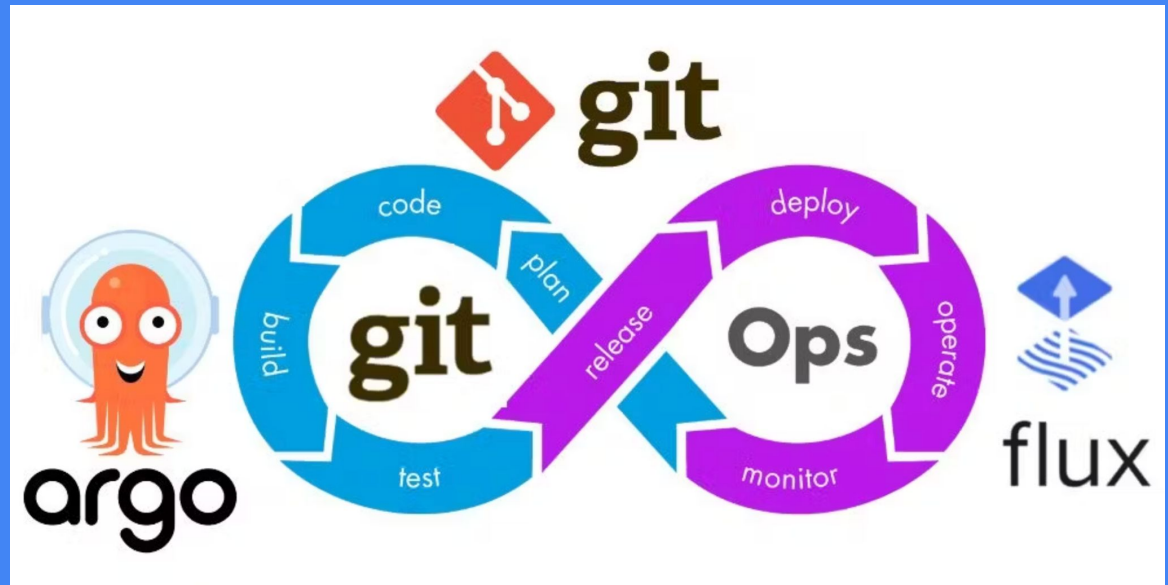


Hakan BAYRAKTAR

GitHub Actions ve ArgoCD ile GitOps Workflow (Node.js - GKE)

- Gitops Nedir?
- ArgoCD
- KUBERNETES
- GKE (Google Kubernetes Engine)
- **Uygulamalı Proje:** GitHub Actions ve ArgoCD ile Node.js Uygulamasının GKE Cluster'a Otomatik Olarak Deploy Edilmesi

GitOps



GitOPS Nedir?

GitOps, Kubernetes için tüm konfigürasyonların Git ile yönetildiği ve otomatik olarak dağıtıldığı bir yazılım geliştirme yaklaşımıdır



Bileşenler:

- **Git deposu:** Versiyon kontrolü ve sistemin tek doğruluk kaynağı
- **CD Aracı:** Otomatikleştirme (Argo CD, FluxCD)
- **Kubernetes:** Hedef sistem (deployment ortamı)



Avantajları:

- Versiyon kontrolü
- Geri alınabilir yapı
- Otomatik ve güvenli dağıtım
- Audit log ve işbirliği

GitOps: From Principles to Practice

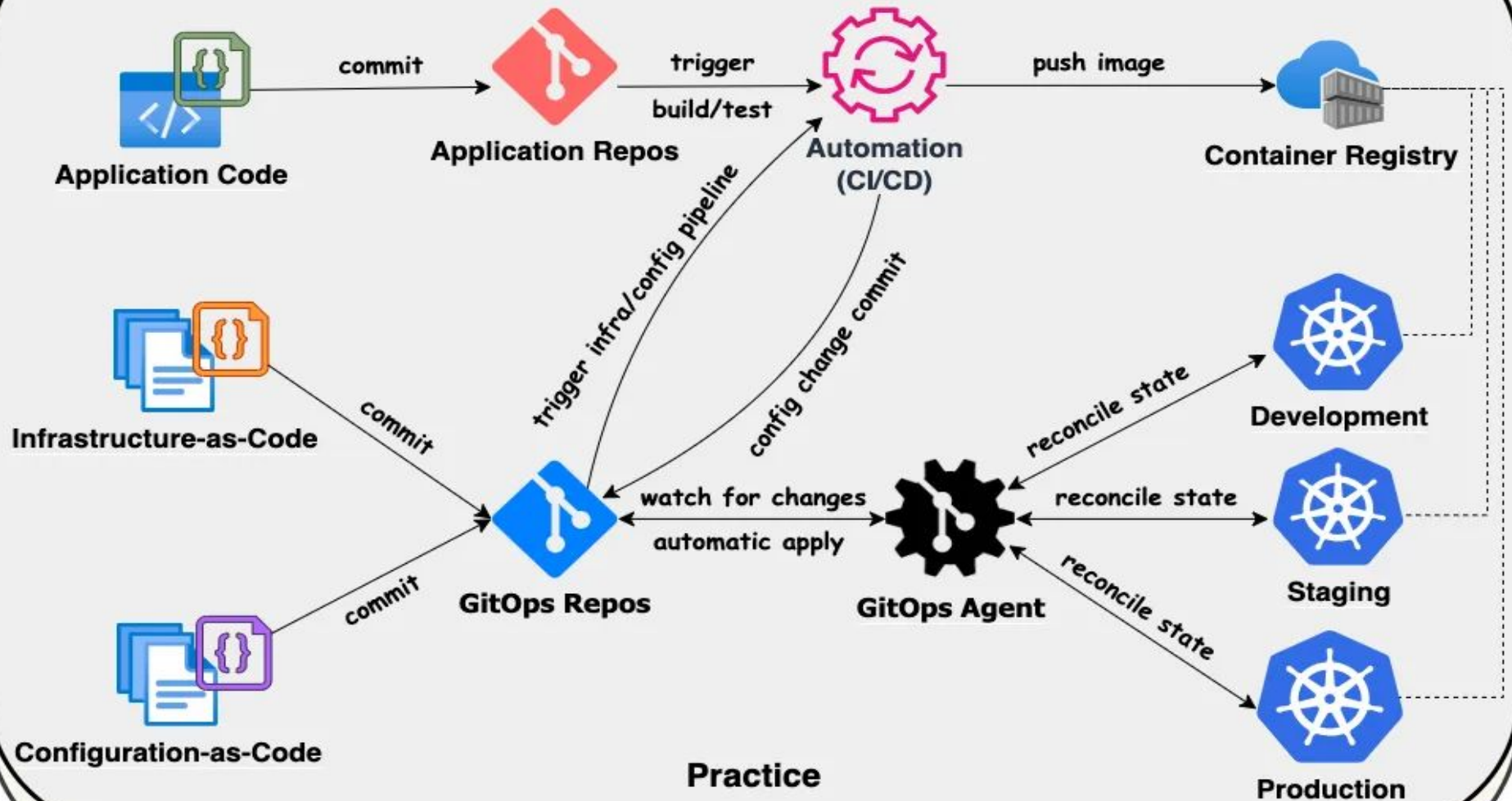
Principles

"Everything-as-Code"

Git:
Single Source of Truth

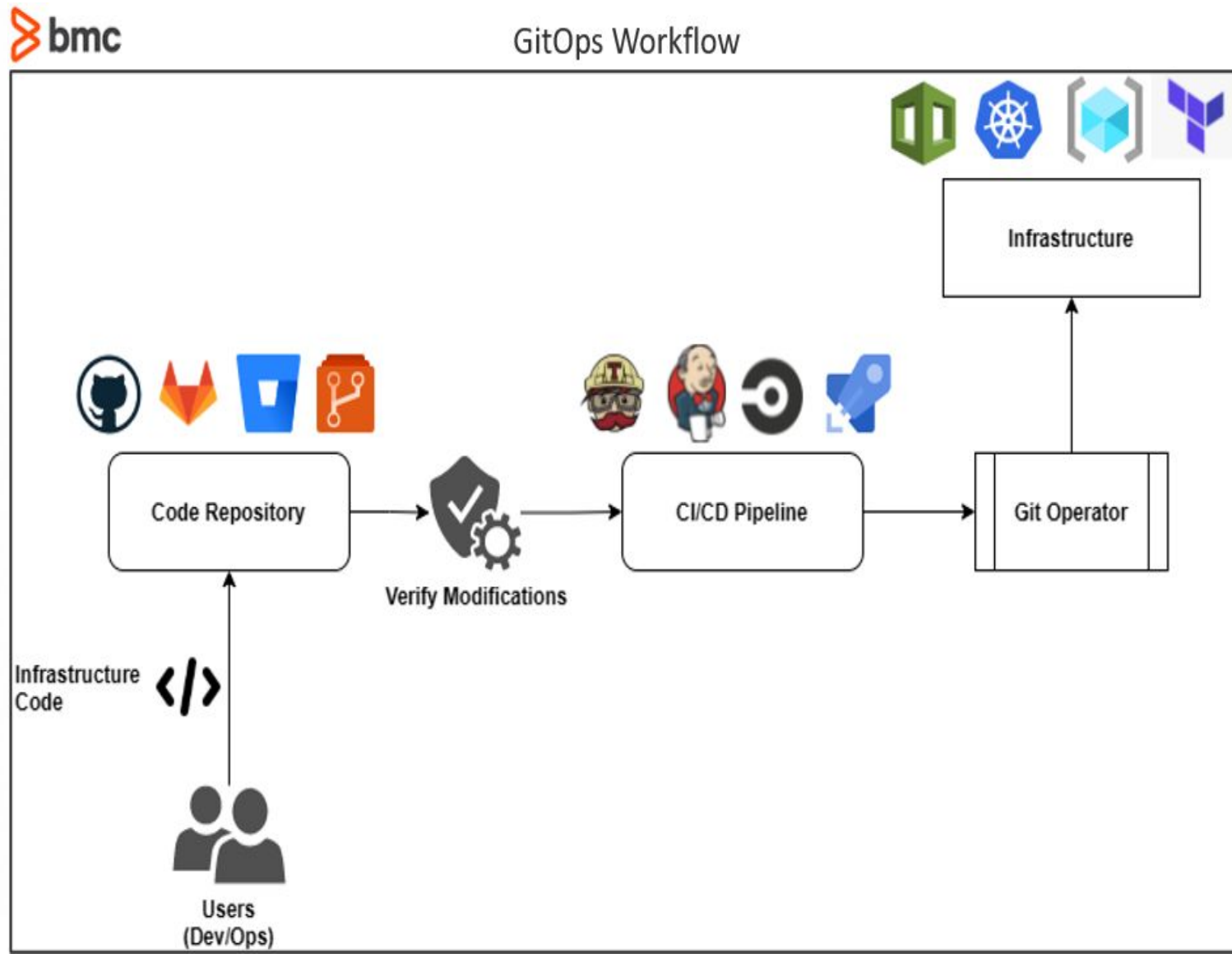
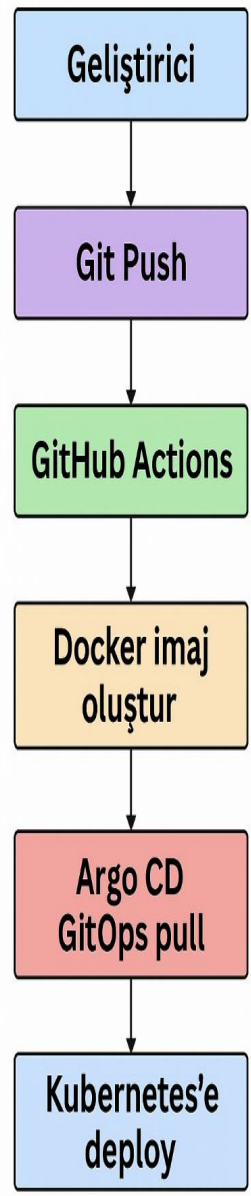
Desired State
Automation

Continuous
Reconciliation



GitOps Akış Diyagramı

GitOPS Nasıl Çalışır?



Popüler GitOps Araçları

Argo CD

UI/CLI ile zengin, Helm/Kustomize destekli güçlü GitOps aracı

Flux CD

CNCF destekli, hafif, CLI odaklı, GitOps operatörü temelli sistem

Jenkins X

GitOps + CI özellikleri sunar, kompleks projeler için uygundur

Fleet

Rancher tarafından geliştirilmiş, multi-cluster odaklı GitOps çözümü



FLEET



flux



JENKINSX

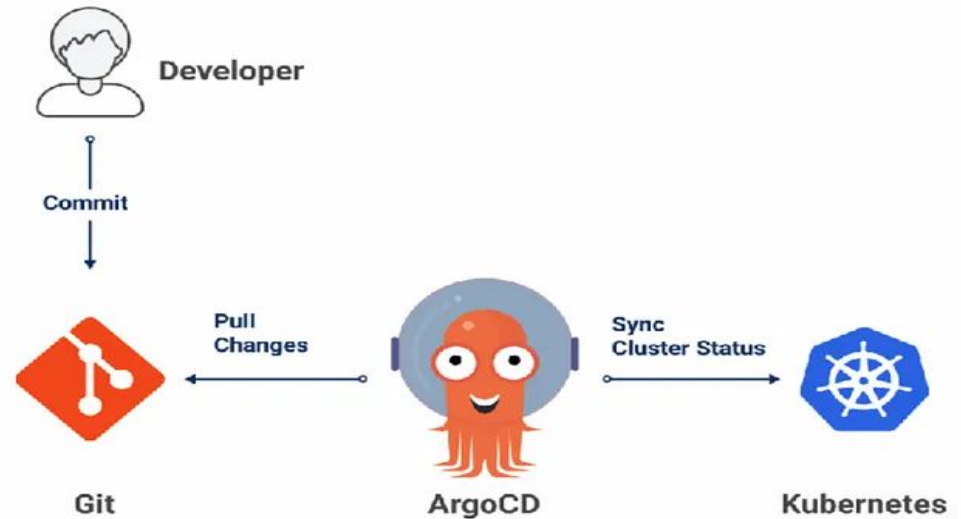


ArgoCD



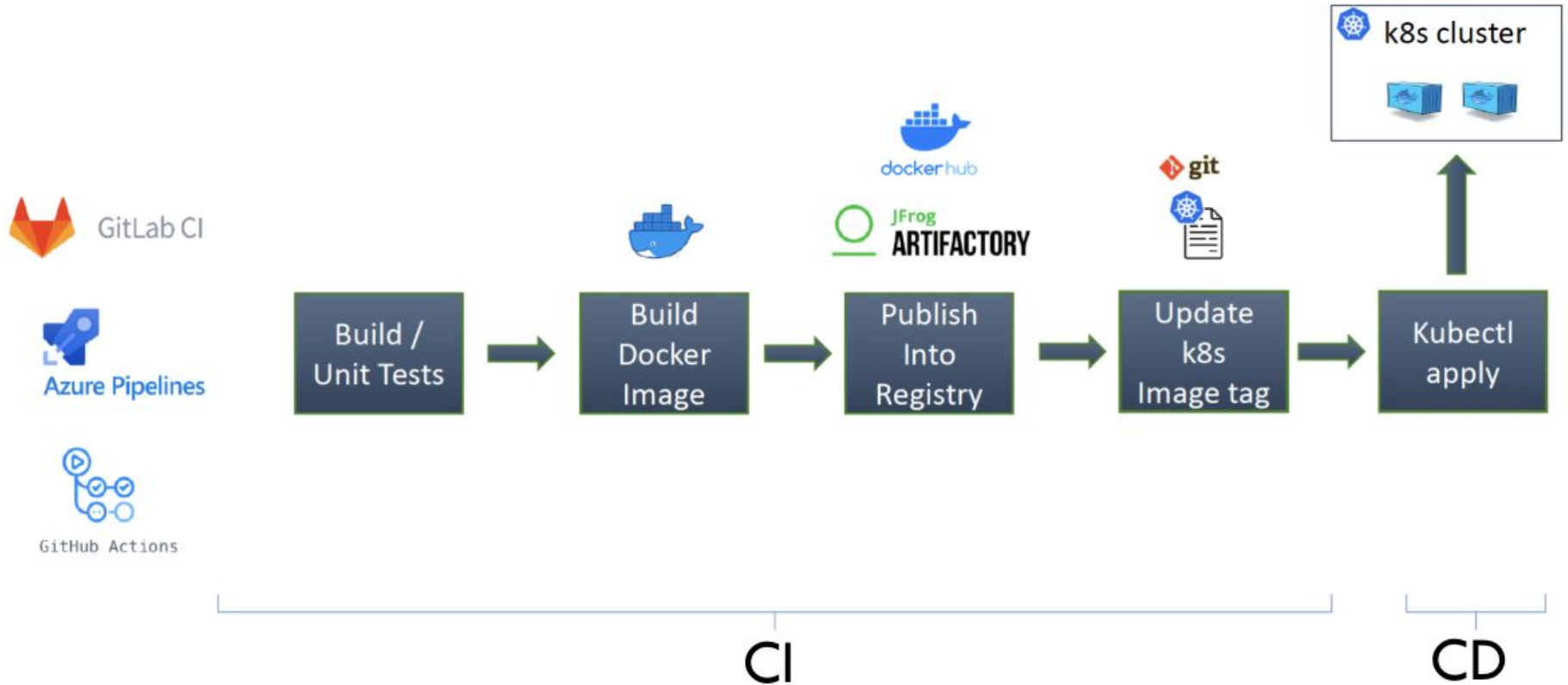
ArgoCD Nedir?

Argo CD, Kubernetes için deklaratif (bildirime dayalı) bir **GitOps sürekli teslim (CD)** aracıdır.

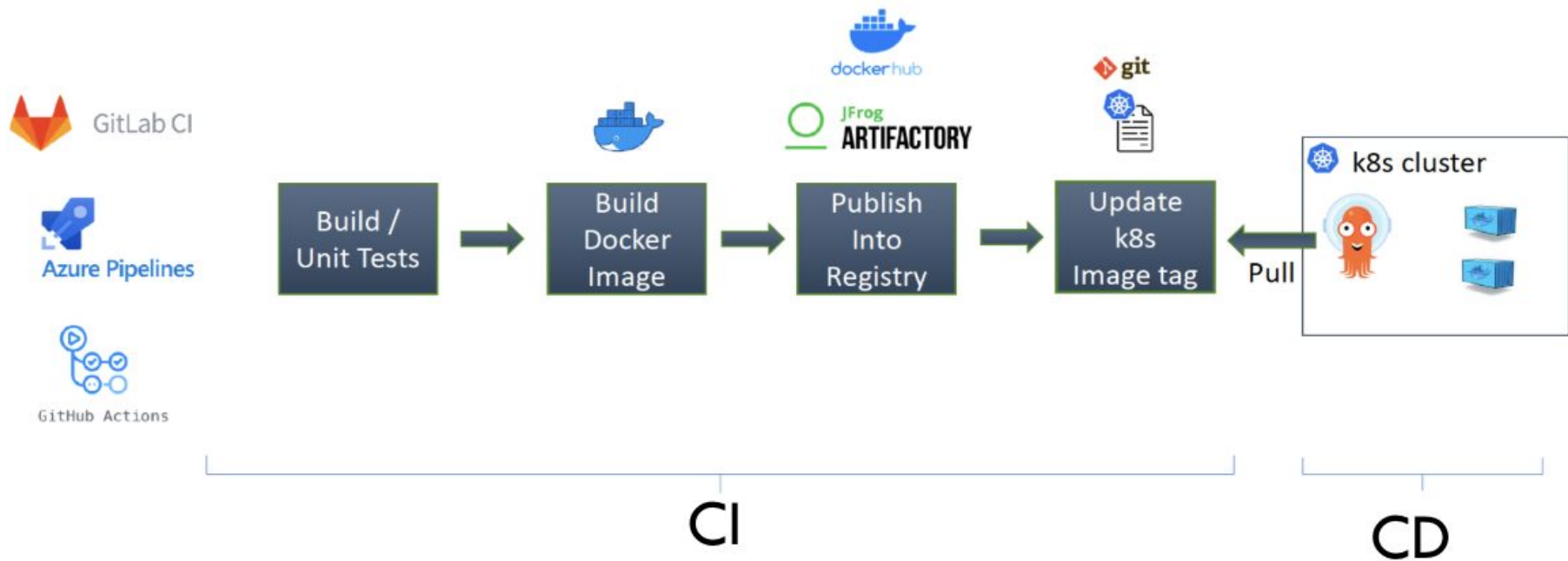


- Git deposunu referans alır.
- Kubernetes manifest dosyalarını uygular.
- Otomatik senkronizasyon ve self-healing desteği
- Helm, Kustomize, YAML, Jsonnet desteği
- Web UI + CLI + API erişimi

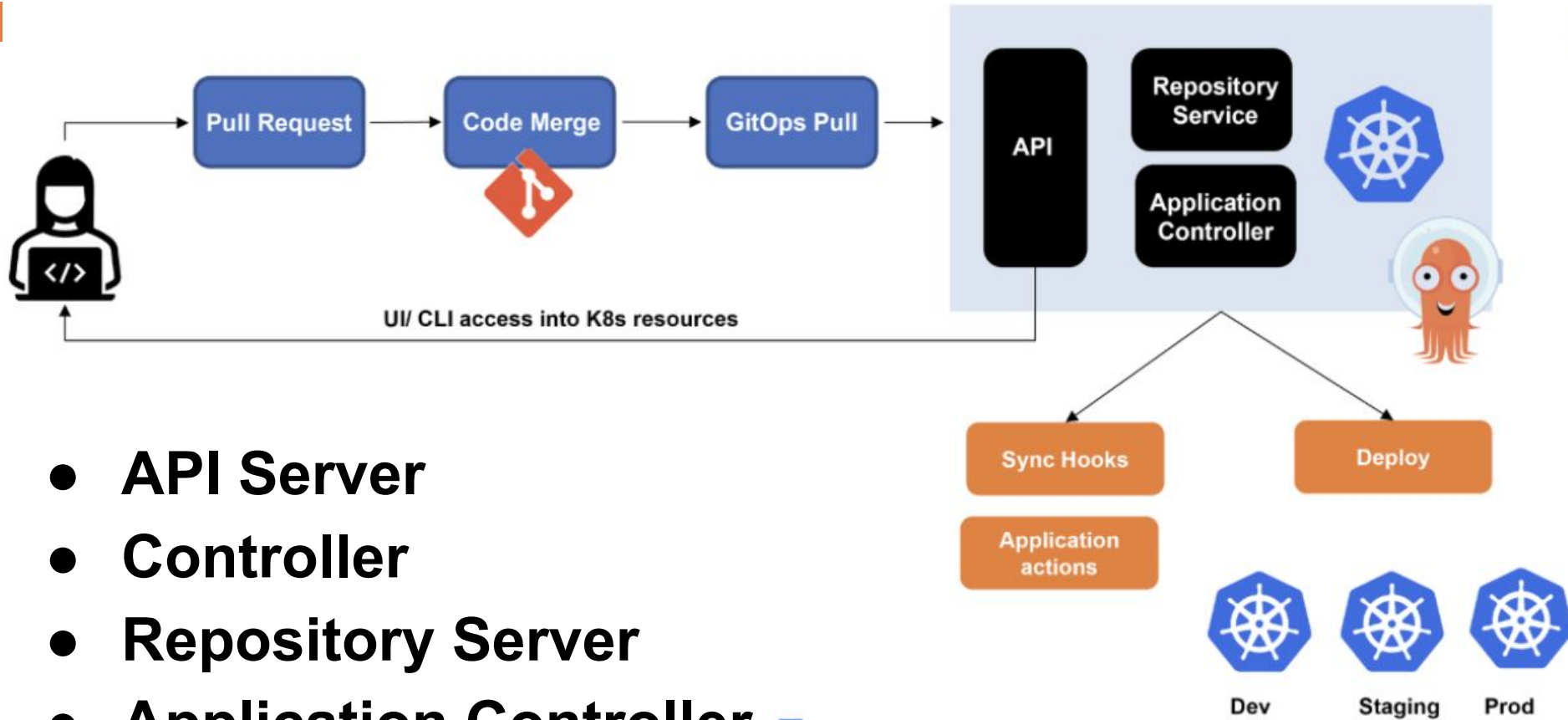
Traditional CD Tools - Push Model



ArgoCD: Pull-Based CD Model

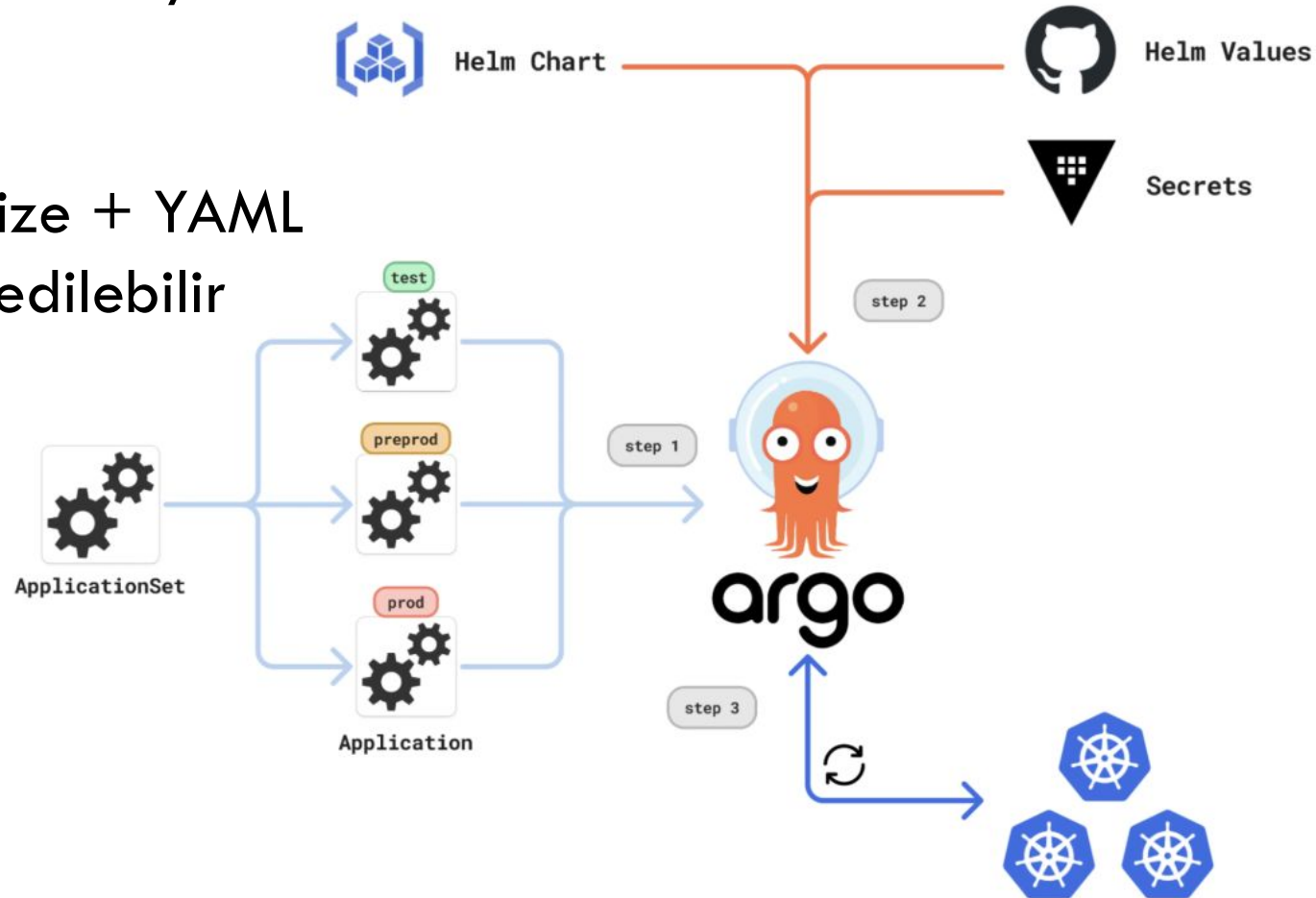


Architecture of Argo CD

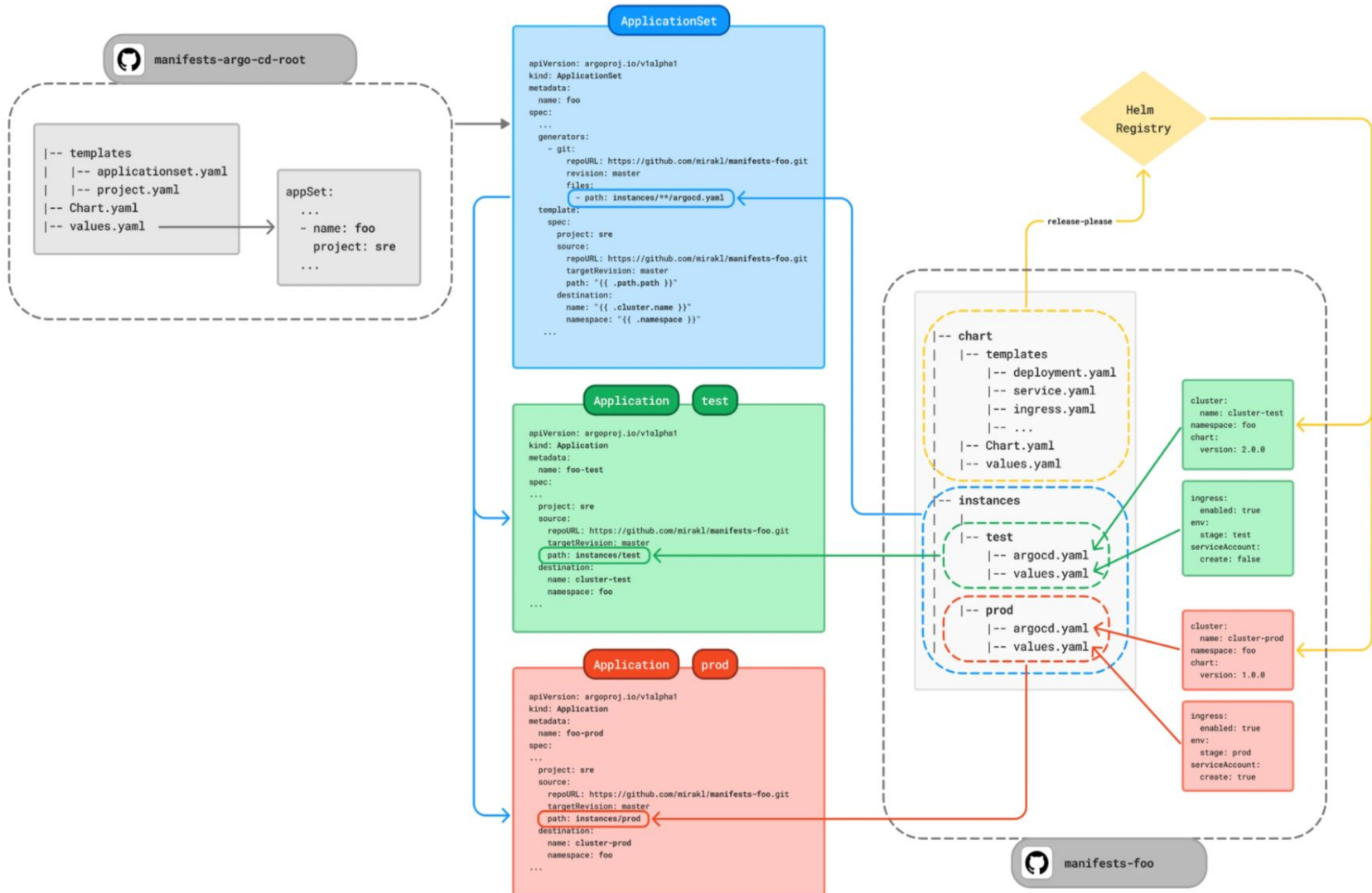


ArgoCD Avantajları

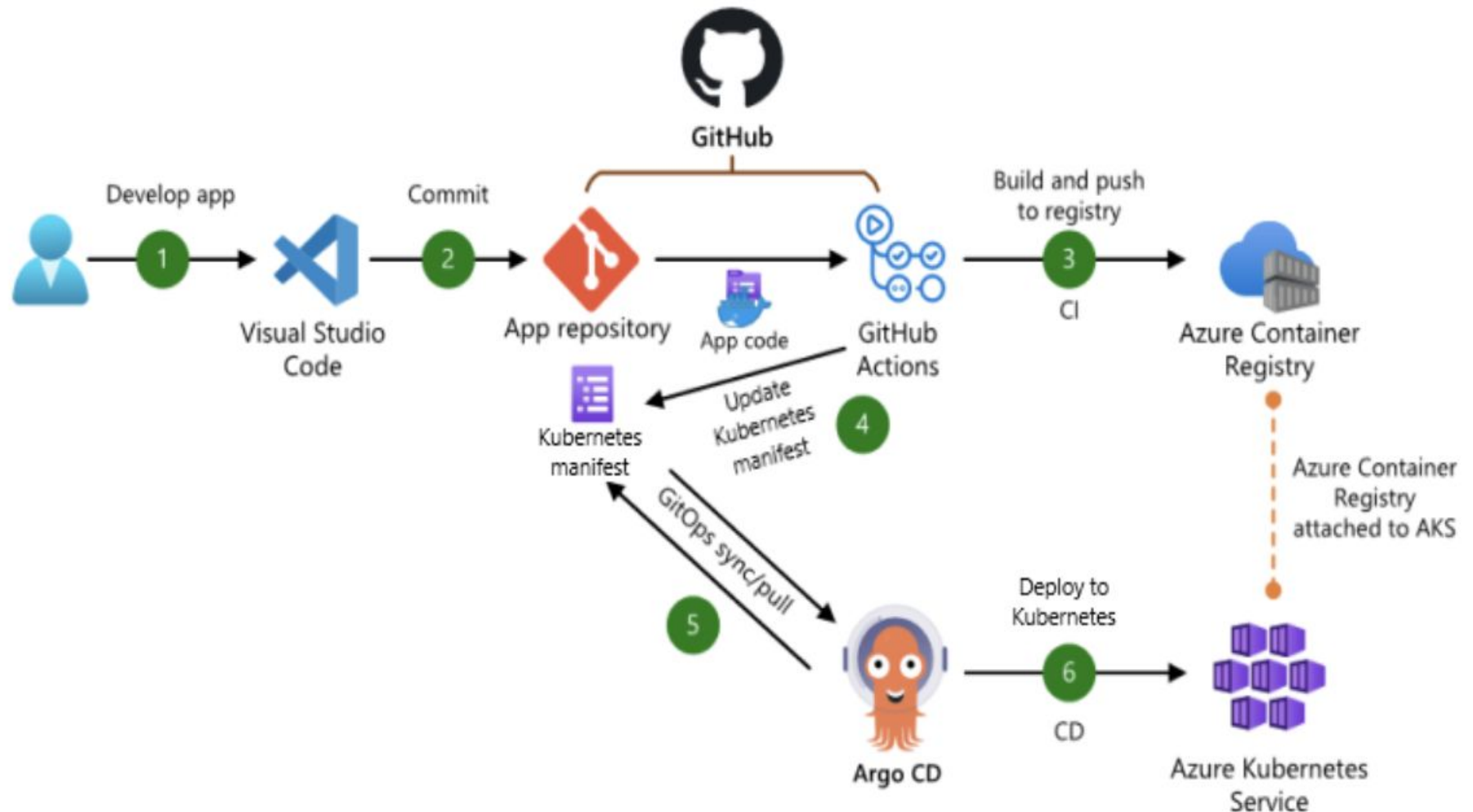
- Görsel UI
- Otomatik senkronizasyon ve self-healing
- Güvenli erişim
- Helm + Kustomize + YAML
- Kolay entegre edilebilir



ArgoCD Mimarisi



Argo CD ile CI/CD Nasıl Kurulur?



ArgoCD Arayüzü

Applications / **argocd-dev**

APP DETAILS APP DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH

APP HEALTH **Progressing** CURRENT SYNC STATUS **Synced** To 3.33.2 (3.33.2) LAST SYNC RESULT **Sync OK** To 3.33.2

Succeeded a few seconds ago (Thu Feb 03 2022 11:48:36 GMT-0500)

argocd-dev 90%

argocd-dex-server	argocd-dex-server-token-98nkw	
argocd-server	argocd-server-token-r7mf8	
argocd-dev-application-controller	argocd-dev-application-controll...	argocd-dev-application-controll...
argocd-dev-dex-server	argocd-dev-dex-server-5b47bf...	argocd-dev-dex-server-5b47bf...
argocd-dev-redis	argocd-dev-redis-64fdb9c8fb	argocd-dev-redis-64fdb9c8fb-c...
argocd-dev-repo-server	argocd-dev-repo-server-6985c...	argocd-dev-repo-server-6985c...
argocd-dev-server	argocd-dev-server-67b9d9649f	argocd-dev-server-67b9d9649f..

Kubernetes





Kubernetes Nedir?

- Kubernetes (k8s), konteynerleştirilmiş uygulamaları **otomatik olarak dağıtan, yöneten ve ölçeklendiren** açık kaynaklı bir platformdur.
- Google tarafından geliştirildi, bugün CNCF tarafından yönetilmektedir.
- Uygulamaları **containers** içinde çalıştırır ve sistem kaynaklarını verimli şekilde kullanır.



Neden Kubernetes?

 **Mikroservis mimarisi için idealdir**

 **Otomatik yeniden başlatma, dağıtım ve yük dengeleme sağlar**

 **Bulut uyumludur – AWS, Azure, GCP gibi sistemlerde çalışır**

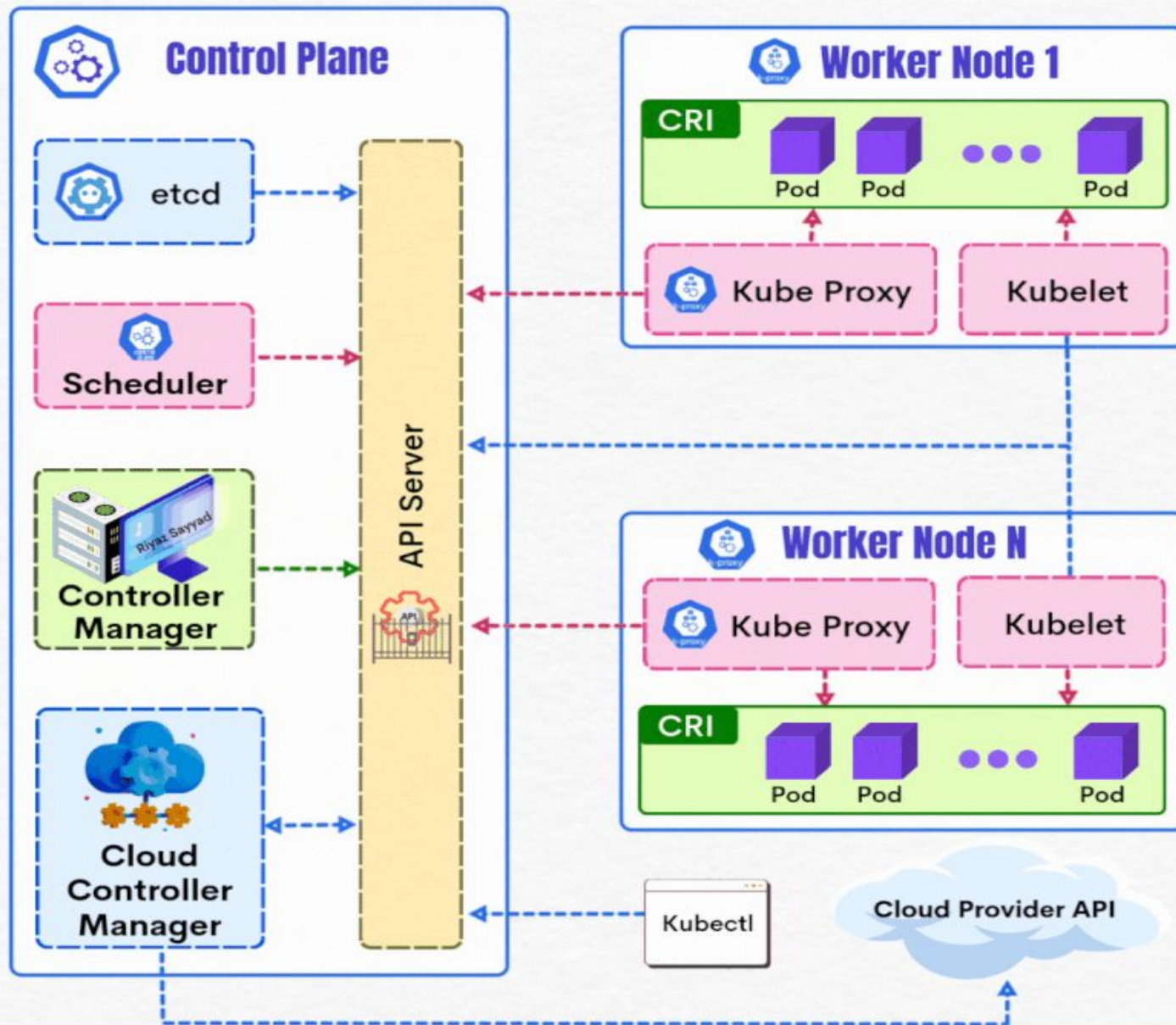
 **CI/CD sistemleriyle kolayca entegre olur**

 **Rolling Update ve Recreate gibi güncelleme stratejileriyle kesintisiz servis sunar**



Kubernetes Architecture

kubernetes Architecture





Control Plane or Master Node


Master Node(Control Plane): Kubernetes kümesinin kontrol merkezidir. Tüm kararlar burada alınır ve cluster yönetimi sağlanır.

Worker node'ların yönetimi, **API isteklerinin** işlenmesi, **pod'ların** dağıtılması ve **kaynak yönetimi** gibi işlevleri içerir.

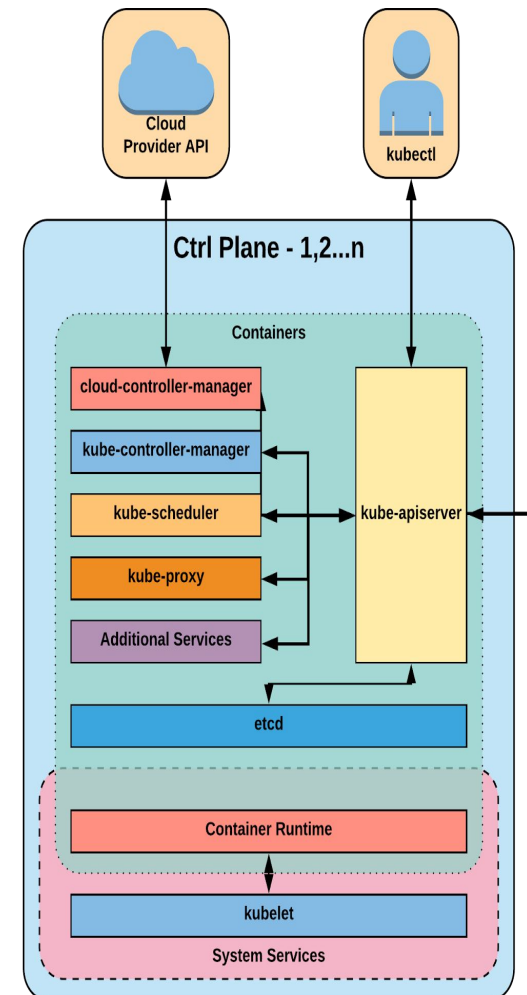
Bileşenler:

 **kube-apiserver:** REST API sağlar, tüm istekler buradan geçer

 **etcd:** Cluster'ın tüm konfigürasyon ve durum verisini saklar

 **kube-scheduler:** Pod'ların hangi node'da çalışacağına karar verir

 **kube-controller-manager:** Sistem durumunu izler ve düzenler





Worker Node

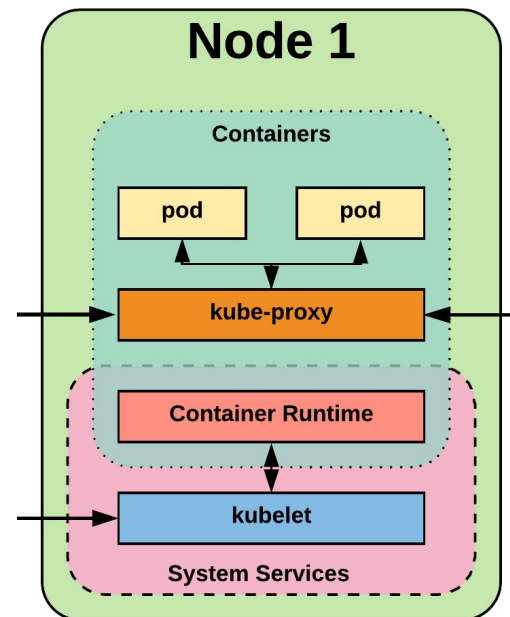
- **Worker Nodes**, Kubernetes kümesinde konteynerleştirilmiş uygulamaların çalıştığı sunuculardır.
- Her Worker Node, üzerinde bir veya daha fazla Pod çalıştırır ve konteynerlerin ağ, depolama gibi kaynaklara erişimini sağlar.

Bileşenler:

⚙️ **kubelet**: Her node'da çalışır, pod'ların durumunu yönetir

🌐 **kube-proxy**: Ağ trafiğini yönlendirir, servisler arası bağlantıyı sağlar

🐳 **Container Runtime**: Container'ları çalıştırır (örnek: containerd, CRI-O)



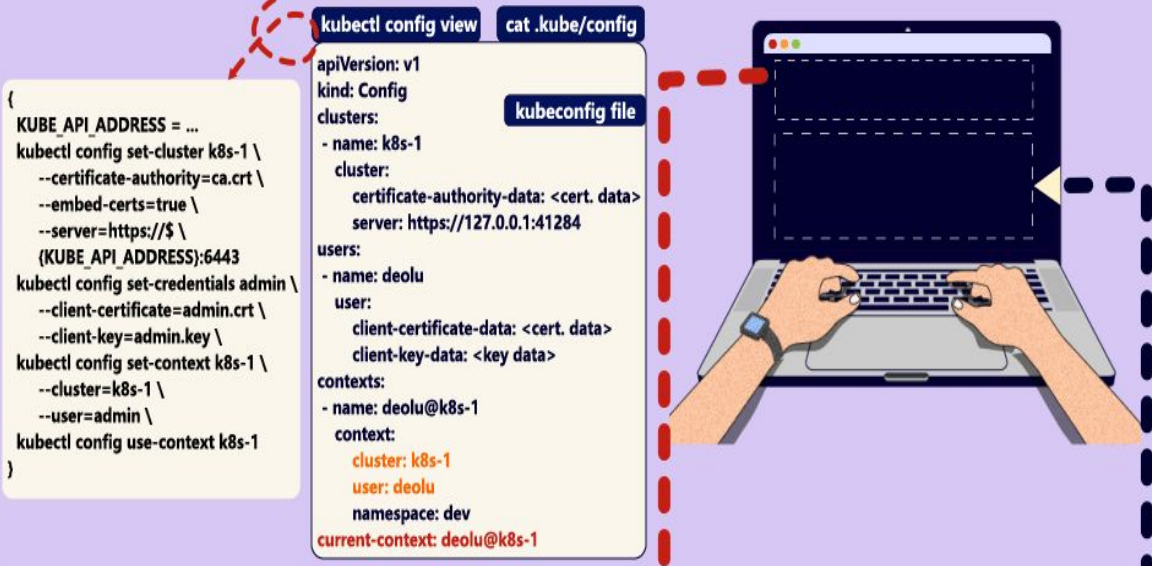
How to Connect to a Kubernetes Cluster

@hellodeolu



CLIENT

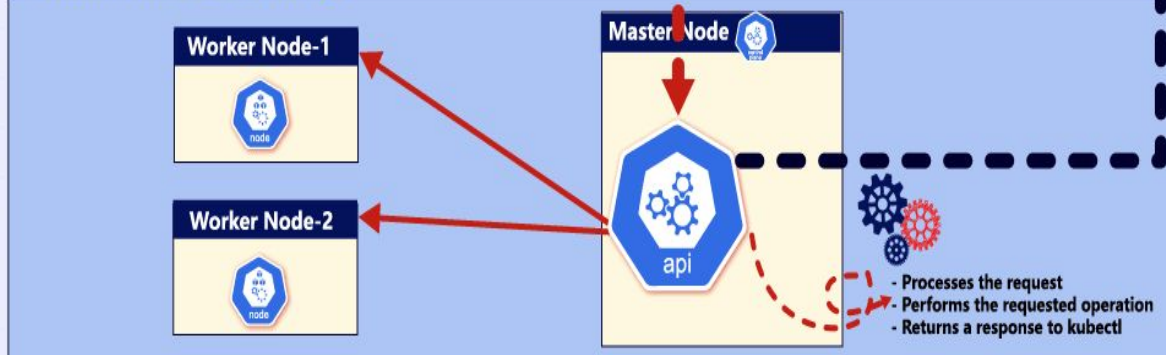
Connecting to a Kubernetes cluster remotely, you usually use **kubectl**. A command-line tool for interacting with Kubernetes resources and performing operations on the cluster. Kubectl uses the credentials and configuration info stored in the **.kube/config** file to authenticate and authorize requests to the API server.



HTTP request

response

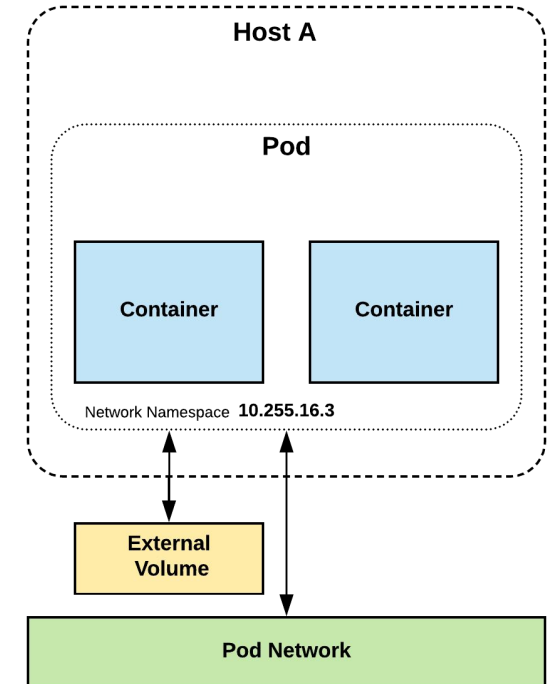
KUBERNETES CLUSTER





Pod

- **Pod**, Kubernetes'in atomik birimi ya da en küçük 'iş birimi'dir.
- Tüm iş yükleri pod'lar üzerinde çalışır
- Podlar, birden fazla konteyneri içerebilir ve bu konteynerler aynı disk alanlarını , ağ alanını paylaşırlar ve tek bir context içinde çalışır."





Pod Examples

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
```


```
apiVersion: v1
kind: Pod
metadata:
  name: multi-container-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: content
    image: alpine:latest
    command: ["/bin/sh", "-c"]
    args:
    - while true; do
      date >> /html/index.html;
      sleep 5;
    done
    volumeMounts:
    - name: html
      mountPath: /html
  volumes:
  - name: html
    emptyDir: {}
```



Deployment

Deployment, pod'ların **yönetimini**, **güncellenmesini ve yeniden oluşturulmasını** sağlayan bir üst yapıdır

 **Recreate:** Tüm pod'lar silinir, yenileri baştan oluşturulur

 **RollingUpdate:** Pod'lar yavaş yavaş sırayla güncellenir

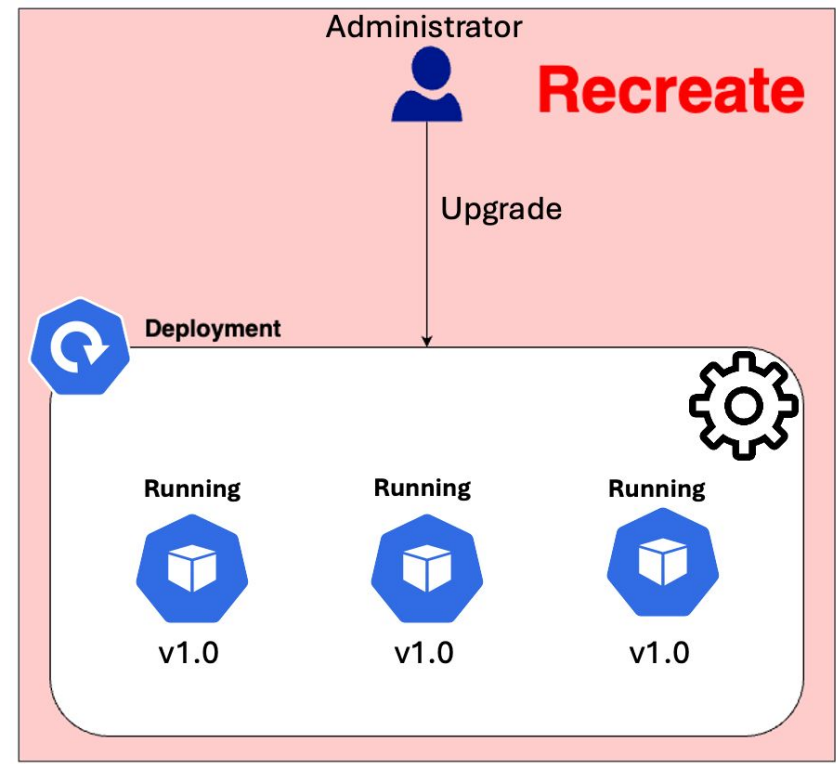
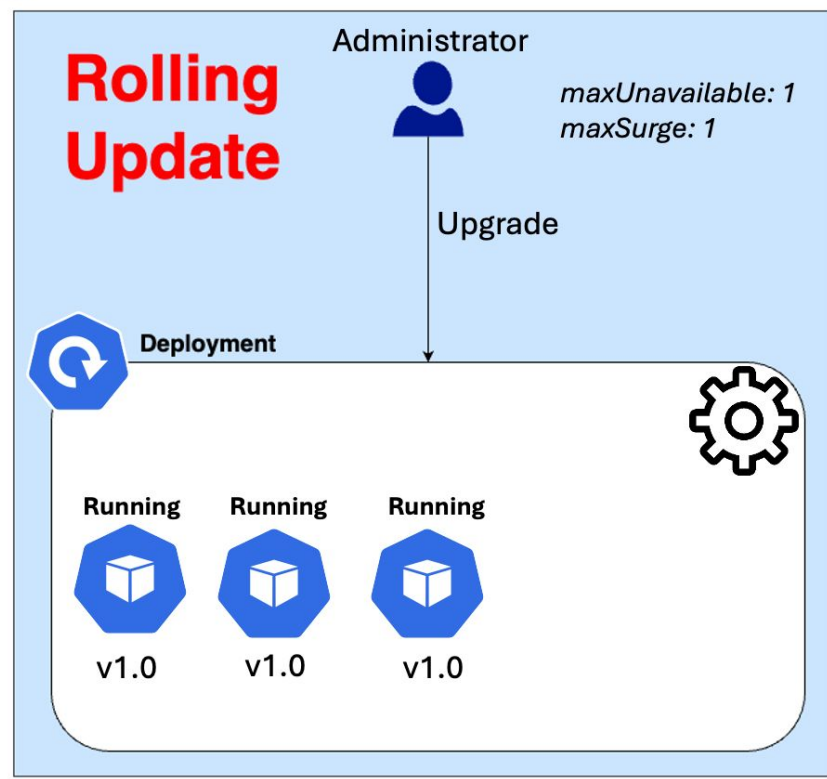
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```



Rolling update-Recreate



Anvesh Muppada



Deployment Strategies



Service Types

There are 4 major service types:

- **ClusterIP** (default)
- **NodePort**
- **LoadBalancer**
- **ExternalName**



ClusterIP Service

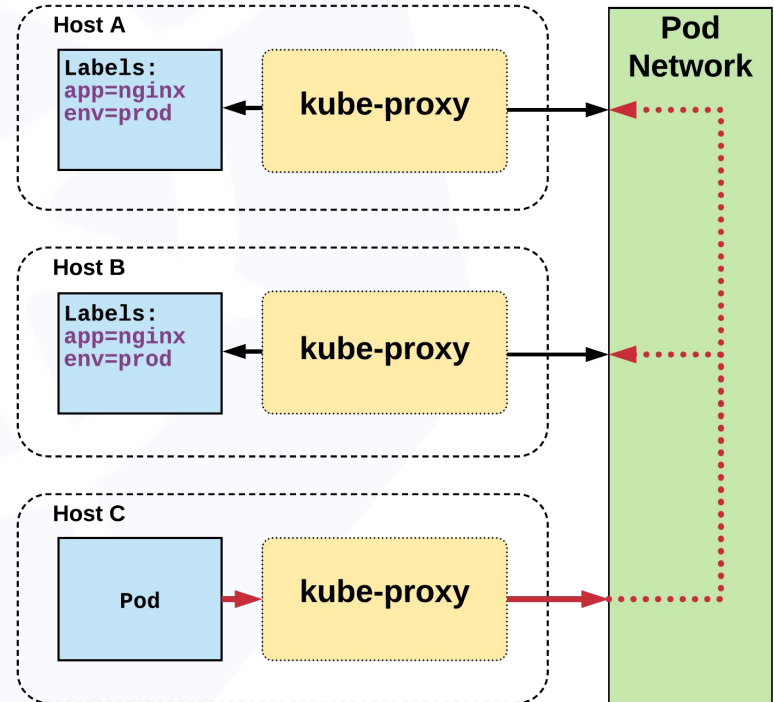
ClusterIP service yalnızca küme içindeki diğer bileşenler tarafından erişilmesine izin verir; dışarıdan erişim yoktur.

Her ClusterIP servisi, küme içinde benzersiz bir sanal IP adresine sahiptir, bu da hizmetin güvenilir bir şekilde tanımlanmasını sağlar.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  selector:
    app: nginx
    env: prod
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Cluster IP Service

```
Name:          example-prod
Selector:      app=nginx,env=prod
Type:         ClusterIP
IP:           10.96.28.176
Port:         <unset> 80/TCP
TargetPort:   80/TCP
Endpoints:    10.255.16.3:80,
              10.255.16.4:80
```



```
/ # nslookup example-prod.default.svc.cluster.local

Name:      example-prod.default.svc.cluster.local
Address 1: 10.96.28.176 example-prod.default.svc.cluster.local
```

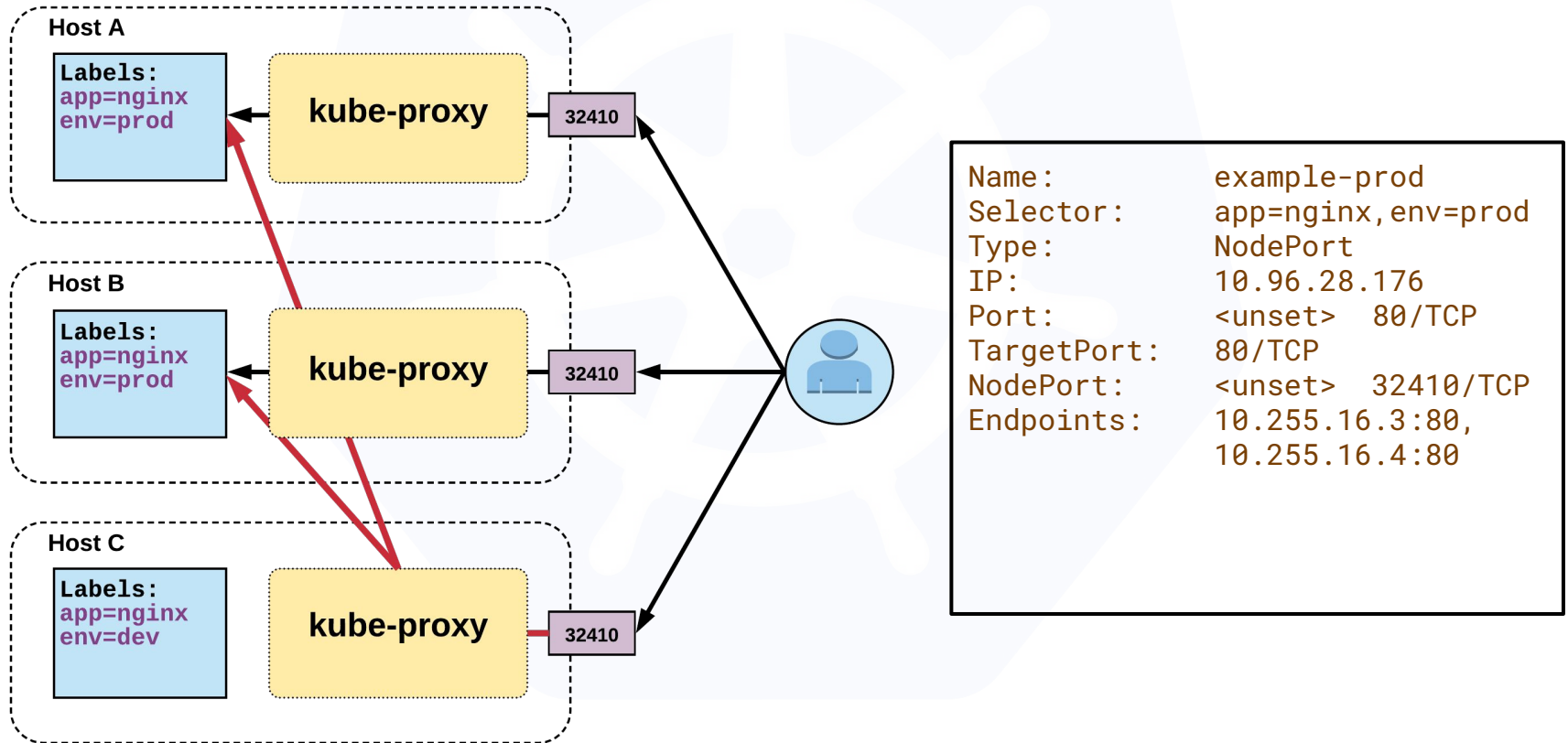


NodePort Service

- **NodePort servisi, ClusterIP servislerini genişleterek, daha fazla erişim imkanı sunar.**
- NodePort, her node'un IP adresinde belirli bir portu maruz bırakır. Bu sayede, kullanıcılar hizmete doğrudan node IP'si üzerinden erişebilir.
- Port, ya statik olarak tanımlanabilir ya da 30000 ile 32767 arasındaki bir aralıktan dinamik olarak alınabilir.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: NodePort
  selector:
    app: nginx
    env: prod
  ports:
    - nodePort: 32410
      protocol: TCP
      port: 80
      targetPort: 80
```

NodePort Service



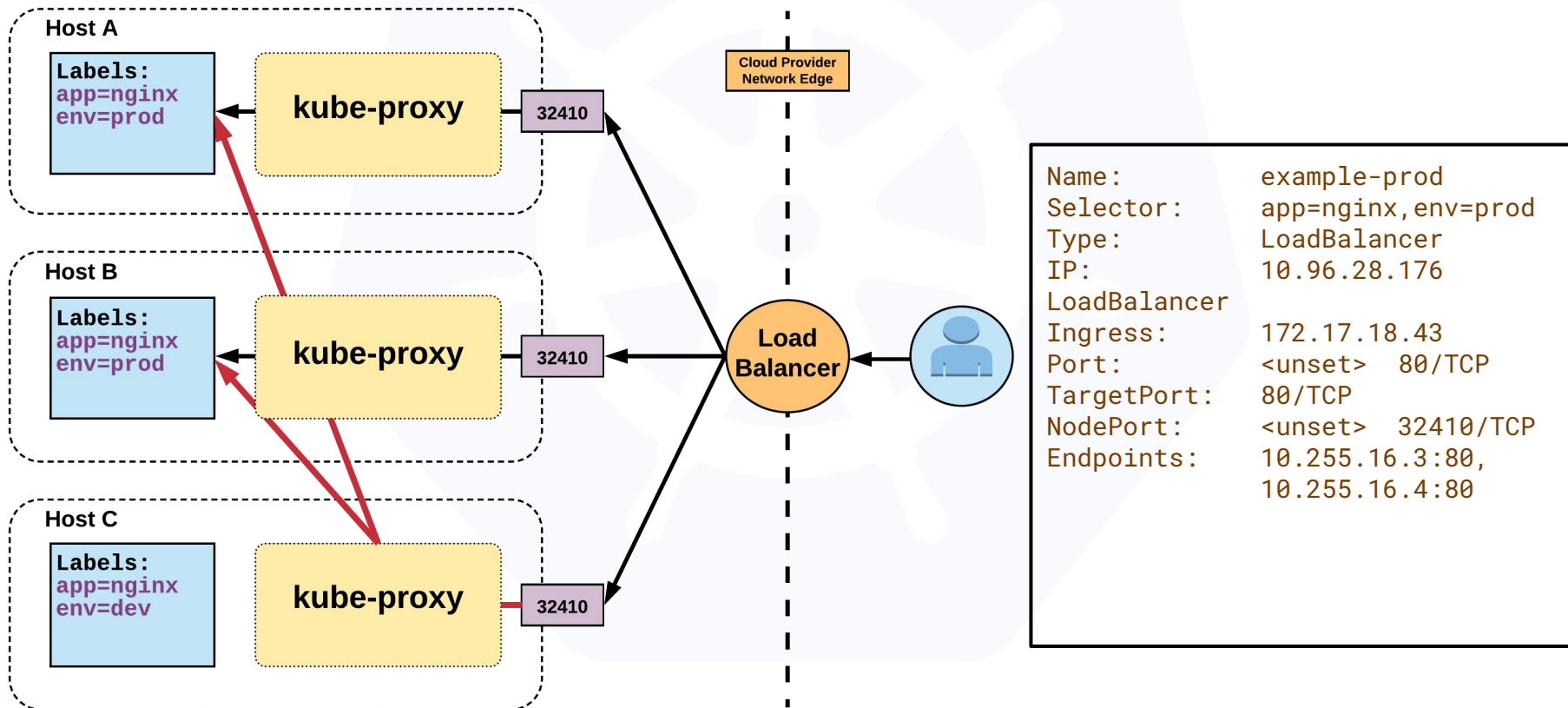


LoadBalancer Service

- **LoadBalancer** services **NodePort** hizmetlerini genişleterek daha gelişmiş bir erişim sağlar.
- gelen trafiği birden fazla node üzerinde dağıtarak yük dengeleme yapar, bu da uygulamanın daha iyi performans göstermesini sağlar.
- Loadbalancer cluster dışında uygulamalara erişim imkanı sağlar.
- LoadBalancer hizmetleri, genellikle bulut sağlayıcılarıyla entegre bir şekilde çalışır.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: LoadBalancer
  selector:
    app: nginx
    env: prod
  ports:
    protocol: TCP
    port: 80
    targetPort: 80
```

LoadBalancer Service



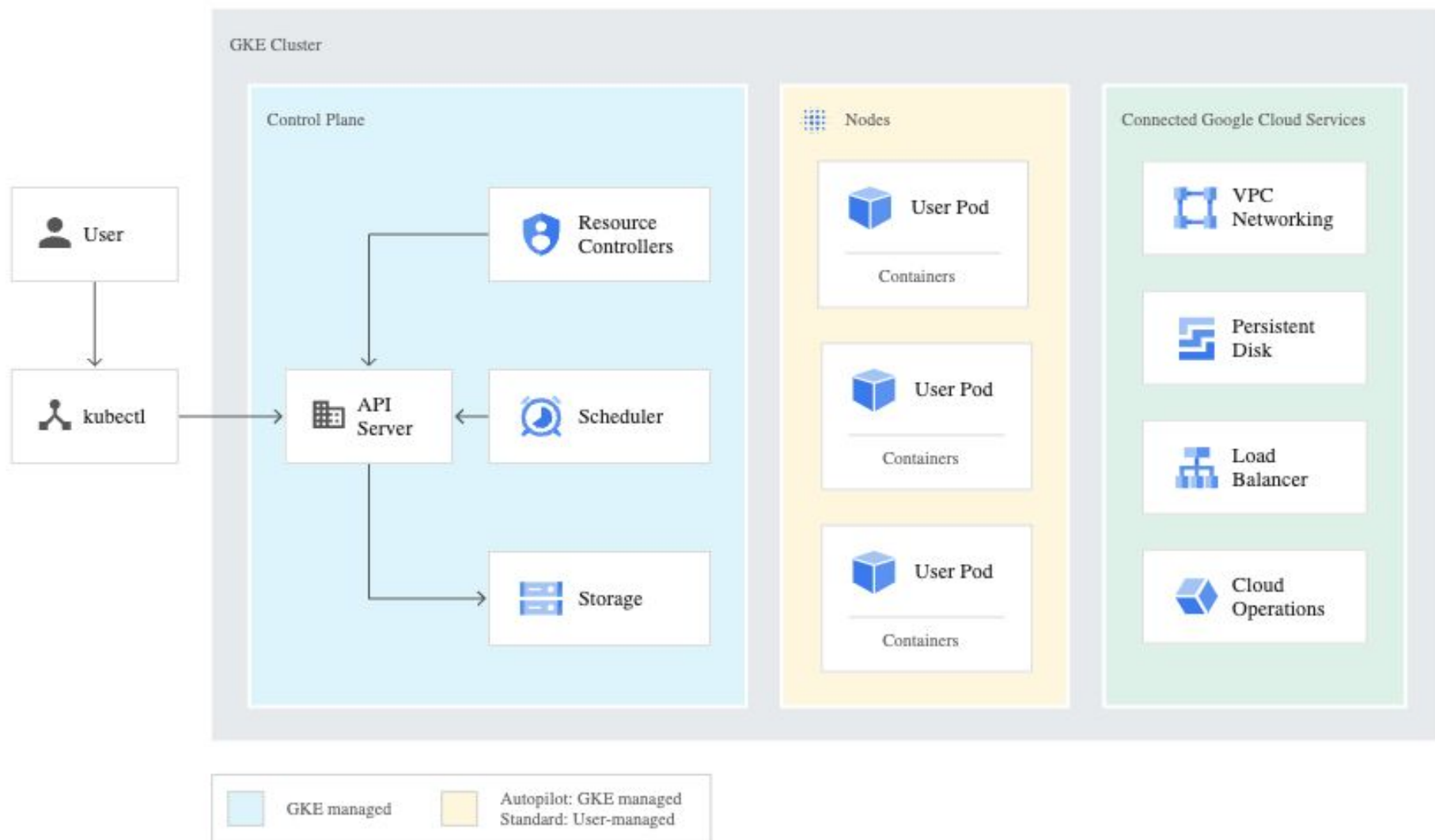


GKE (Google Kubernetes Engine) Nedir?

- GKE, Google Cloud tarafından sağlanan **yönetilen bir Kubernetes servisi**dir
- Kubernetes cluster'larının kurulumu, güncellenmesi, izlenmesi ve bakımı Google tarafından yönetilir
- GKE kullanarak Kubernetes ortamı dakikalar içinde hazır hale getirilebilir
- Otomatik node scaling, monitoring, IAM entegrasyonu gibi özellikler sunar



GKE Cluster Architecture





GKE has two types of cluster

	Autopilot mode	Standard mode
	Optimised Kubernetes cluster with a hands-off experience	Kubernetes cluster with node configuration flexibility
Scaling	Automatic based on workload	You configure scaling
Nodes	Google manages and configures your nodes	You manage and configure your nodes
Configuration	Streamlined configuration ready to use	You can configure all options
Workloads supported	Most workloads except these limitations	All Kubernetes workloads
Billing method	Pay per pod	Pay per node (VM)
SLA	Kubernetes API and node availability	Kubernetes API availability



GKE ile Çalışma Adımları



GKE Cluster oluştur



GCloud CLI ile kimlik doğrulaması yap



Uygulama manifest dosyaları hazırla
(deployment, service, vb.)



`kubectl apply -f` komutlarıyla dağıtımı
başlat



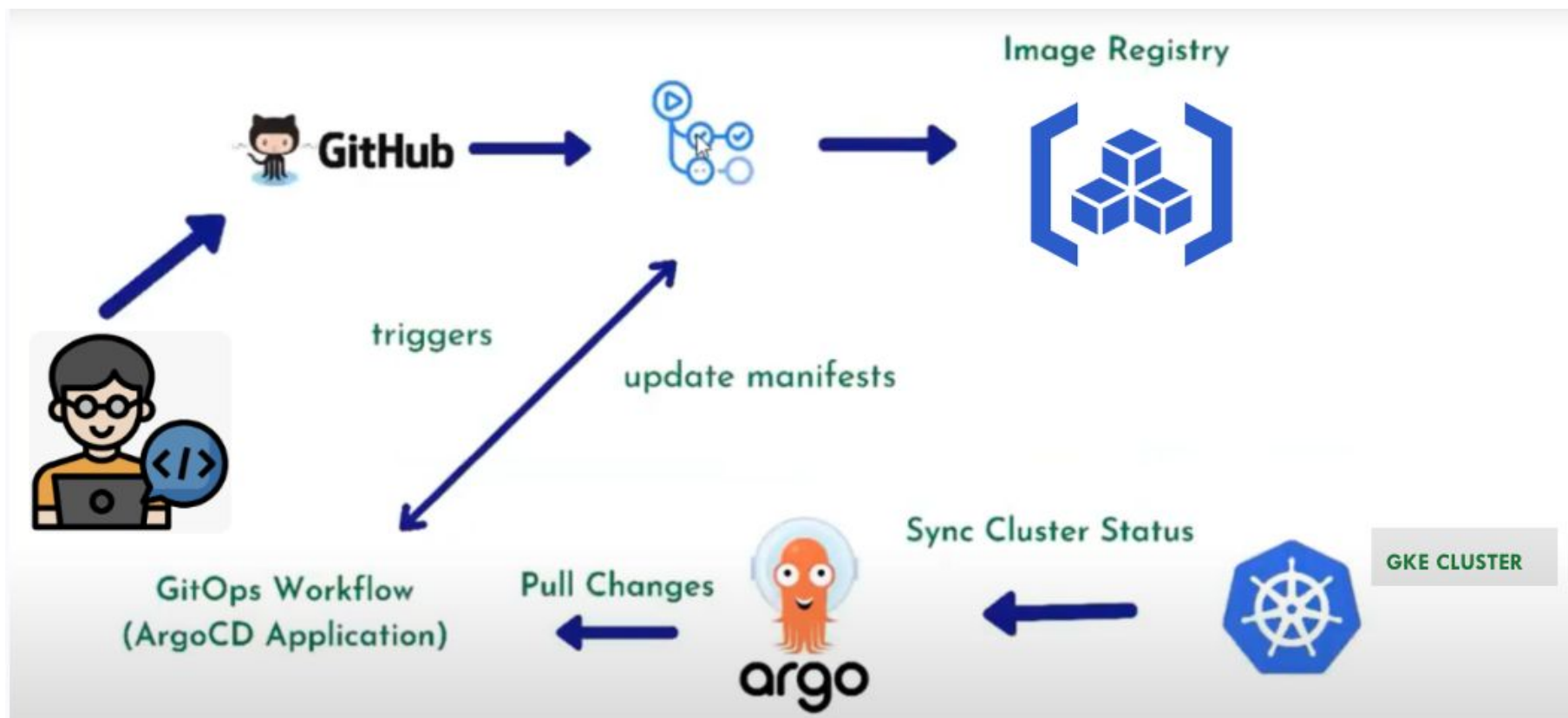
`kubectl get pods` ile pod durumunu kontrol
et



LoadBalancer veya Ingress ile dış dünyaya aç

GitHub Actions ve ArgoCD ile GitOps Workflow (Node.js - GKE)






ProJenin Adımları



- 1 Projeyi Klonla ve Lokal Ortamda Çalıştır**
- 2 GitHub Reposu ve Personal Access Token (PAT) Oluştur**
- 3 Google Cloud Projesi ve Service Account Oluştur**
- 4 Artifact Registry Oluştur**
- 5 GKE Cluster'ı Otomasyon Scripti ile Kur**
- 6 ArgoCD Kurulumu ve GitOps Yapılandırması**
- 7 GitHub Secrets ve Variables Tanımla**
- 8 GitHub Actions ile CI/CD Pipeline ve ArgoCD Entegrasyonu**
- 9 Canlı Ortamda Uygulamayı Test Et ve Doğrula**



Eğitime ait tüm sunum, döküman ve materyaller için:

<https://github.com/hakanbayraktar/ibb-tech>

Uygulamalı proje kodlarına ulaşmak için:

<https://github.com/hakanbayraktar/news-summary-gke> adresinden ulaşabilirsiniz