



YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ
SOFTWARE TECHNOLOGIES RESEARCH INSTITUTE

Rook ile Kubernetes Üzerinde Depolama Orkestrasyonu

BERKAN YILDIRIM

Devops Engineer at TÜBİTAK BİLGEM YTE



İçerik

- | | |
|----|---|
| 3 | <u>Kubernetes Üzerinde Depolamanın Gelişimi</u> |
| 4 | <u>PV & PVC Kavramları</u> |
| 5 | <u>Binding İşlemi / Dynamic Provisioning</u> |
| 9 | <u>Ceph Yapısı ve Bileşenleri</u> |
| 14 | <u>Rook Operatörü</u> |
| 20 | <u>Rook - CEPH External</u> |



2014

Google Kubernetes projesini başlattı. İlk sürümlerde temel iş yükü yönetimi sağlandı, ancak veri tutma konusunda henüz kapsamlı özellikler yoktu.



2015

Persistent Volumes (PV) ve **Persistent Volume Claims (PVC)** tanıtıldı. Bu, kalıcı veri depolama için temel bir yapı sağladı.



2016

Dynamic Provisioning özelliği tanıtıldı. Depolama sağlayıcıları dinamik olarak PV oluşturabilir hale geldi.



2017

StorageClass tanıtıldı. Bu özellik, farklı depolama sınıfları oluşturma ve yönetme imkanı sundu.



2018

Container Storage Interface (CSI) tanıtıldı. Bu, Kubernetes'in farklı depolama çözümleriyle daha uyumlu çalışmasını sağladı.
Volume Snapshots desteği tanıtıldı.



2020

CSI Volume Cloning desteği tanıtıldı. Mevcut PVC'den yeni bir PVC klonlama imkanı sunuldu.

Persistent Volumes (PV)

Kubernetes kümesinde depolama sağlayan bir kaynak türüdür. PV, bağımsız bir depolama birimi olarak tanımlanır ve Kubernetes kümesinde dinamik veya manuel oluşturulabilir.

```
● ● ●  
  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: my-pv  
spec:  
  capacity:  
    storage: 10Gi  
  accessModes:  
    - ReadWriteOnce  
  persistentVolumeReclaimPolicy: Retain  
  hostPath:  
    path: "/mnt/data"
```

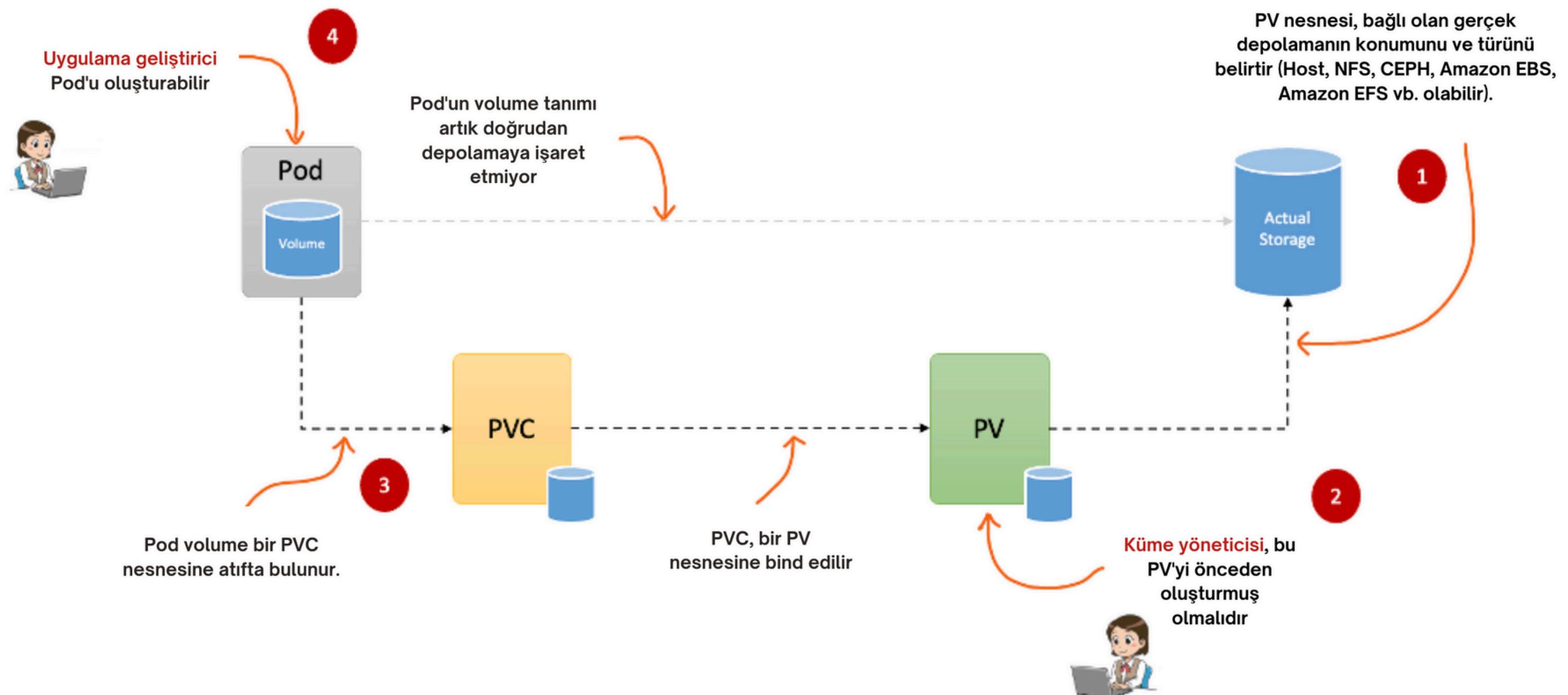
Persistent Volume Claims (PVC)

Kullanıcıların belirli bir PV kaynağını talep etmelerini sağlayan bir istektir. PVC, uygulamaların ihtiyaç duyduğu depolama alanını belirtir ve uygun PV ile eşleşir.

```
● ● ●  
  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: my-pvc  
spec:  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 10Gi
```

Binding İşlemi

PVC oluşturulduğunda, Kubernetes uygun bir PV arar ve PVC'nin ihtiyaçlarını karşılayan bir PV bulunduğuunda bağlar (bind eder). Eşleşme kriterleri arasında kapasite, erişim modları ve StorageClass bulunur.



StorageClass

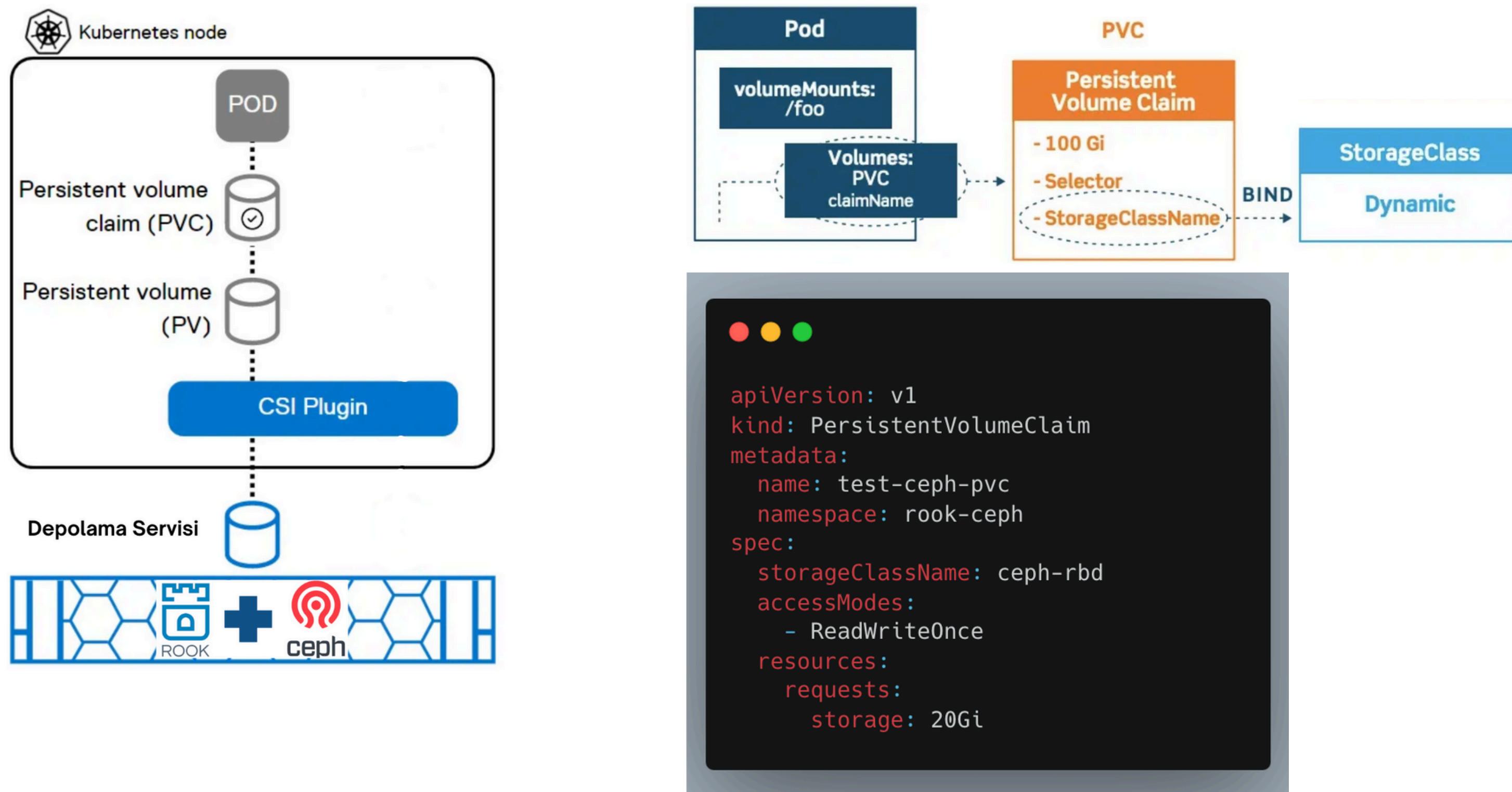
Farklı depolama sağlayıcıları ve konfigürasyonları için bir soyutlama katmanıdır. PVC oluşturulduğunda, belirli bir StorageClass kullanılarak dinamik olarak PV oluşturulabilir.

```
● ● ●

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: ceph-rbd
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
  provisioner: rook-ceph-external.rbd.csi.ceph.com
  parameters:
    csi.storage.k8s.io/fstype: ext4
    csi.storage.k8s.io/provisioner-secret-namespace: rook-ceph-external
    csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
    csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
    csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
    imageFormat: '2'
  clusterID: rook-ceph-external
  imageFeatures: layering
  csi.storage.k8s.io/controller-expand-secret-namespace: rook-ceph-external
  pool: didak1-pool
  csi.storage.k8s.io/node-stage-secret-namespace: rook-ceph-external
  reclaimPolicy: Delete
  allowVolumeExpansion: true
  volumeBindingMode: Immediate
```

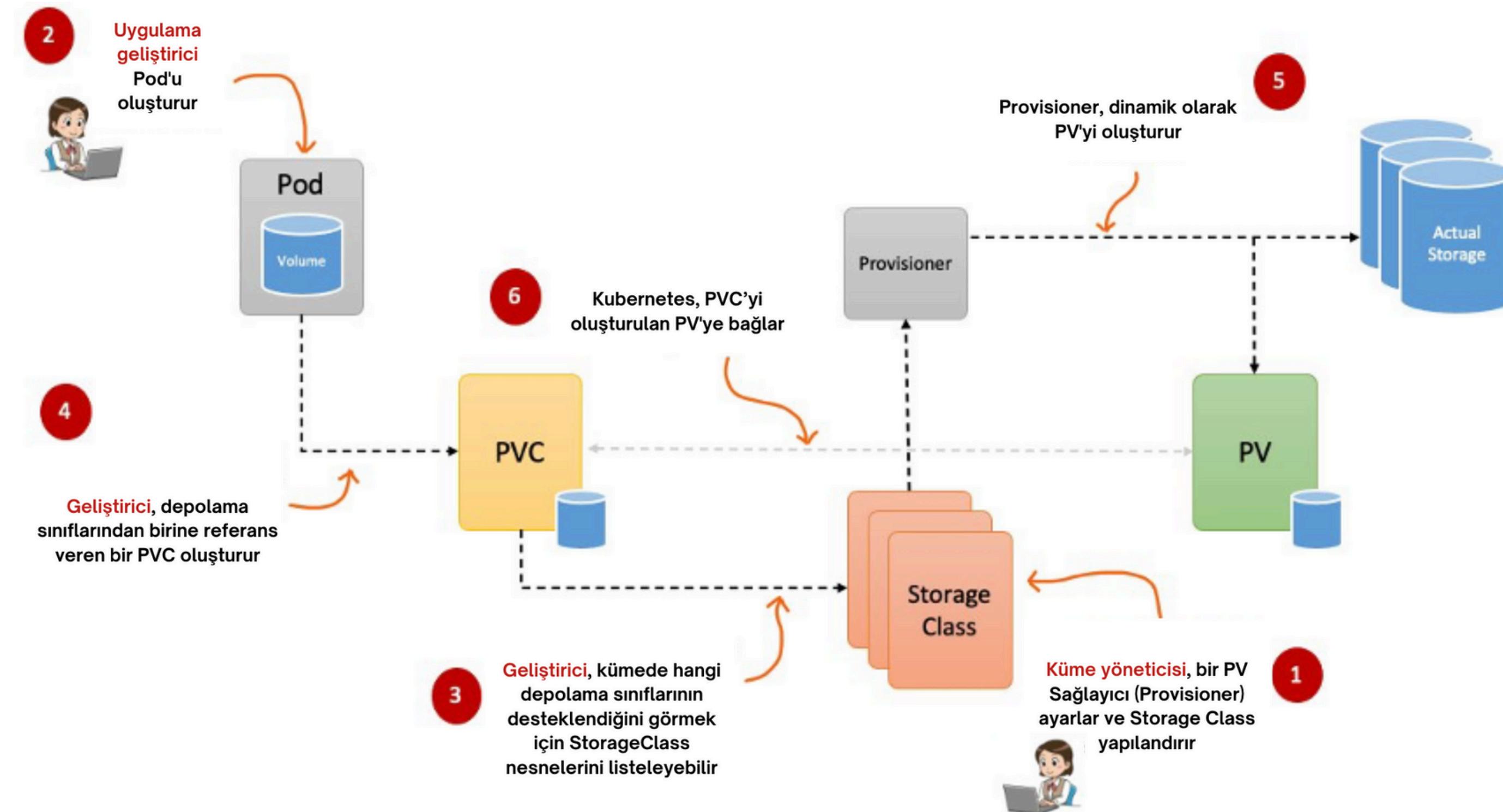
Container Storage Interface (CSI)

- Kubernetes ile depolama sağlayıcıları arasında bir standart arayüz sağlar, entegrasyon sürecini kolaylaştırır.
- Konteyner tabanlı uygulamaların depolama altyapılarını kullanmasını ve yönetmesini kolaylaştırır.
- Farklı depolama sağlayıcıları arasında kolay geçiş yapmayı sağlar.



Dynamic Storage Provisioning

Kubernetes'te depolama kaynaklarını dinamik olarak tahsis etme sürecini ifade eder. Kullanıcılar, belirli depolama sınıflarını kullanarak Persistent Volume Claim (PVC) oluşturduklarında, Kubernetes bu talepleri otomatik olarak karşılayacak Persistent Volume (PV) kaynaklarını dinamik olarak oluşturur. Bu, manuel PV oluşturma gereksinimini ortadan kaldırır ve depolama yönetimini büyük ölçüde basitleştirir.





ceph

"Software defined storage"

"Unified storage system"

"Scalable distributed storage"

"The future of storage"

"The Linux of storage"

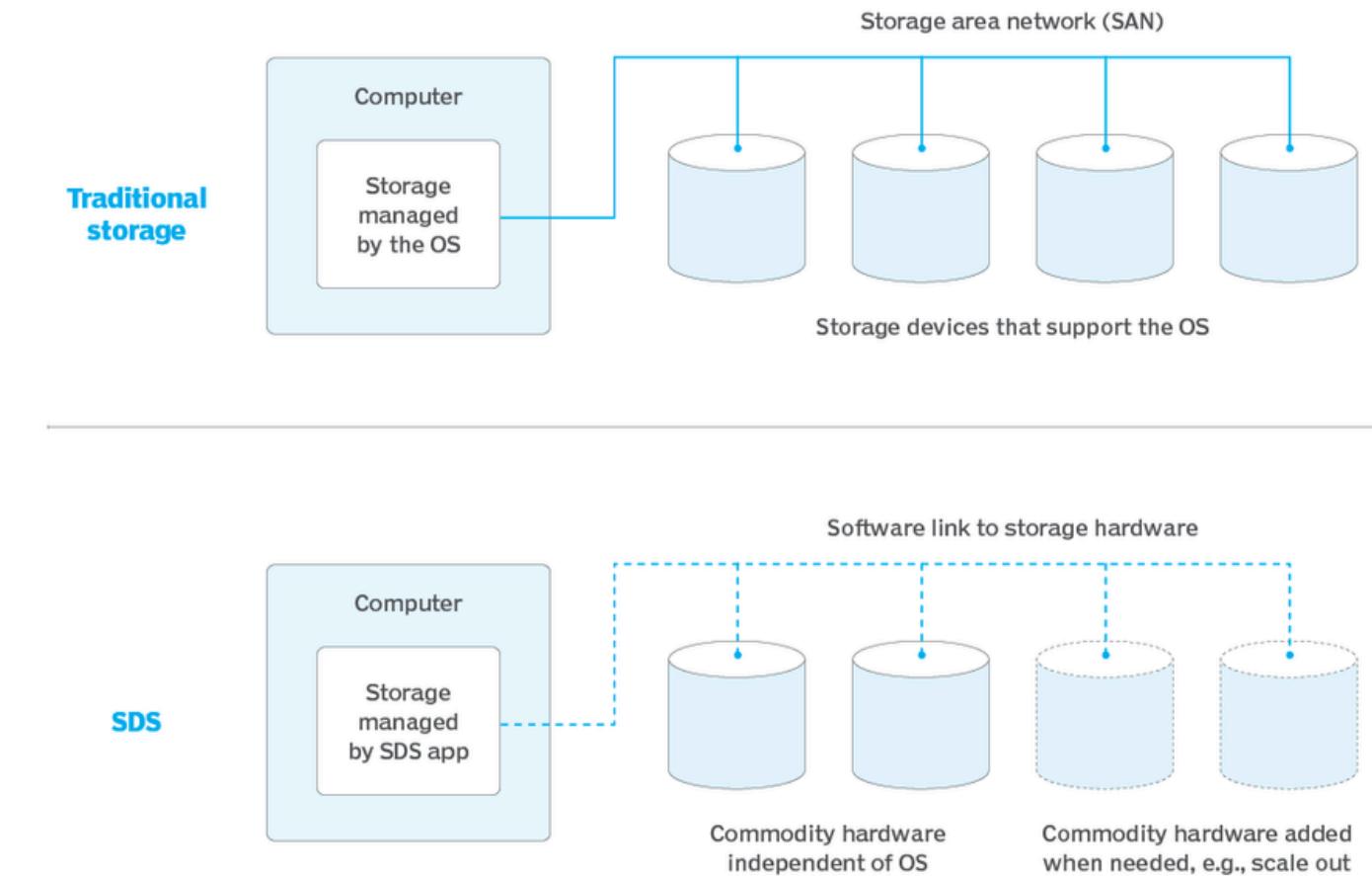


Yazılım Tanımlı Depolama Nedir?

Software Defined Storage (SDS)

- Bir soyutlama katmanı sağlayarak temel depolamanın karmaşıklıklarından kurtarır.
- Depolama tahsis etme mekanizması sağlar. (storage provisioning)
- Operasyonel yönetimi kolaylaştırır
- Farklı depolama cihazlarını tek bir mantıksal havuzda birleştirmeyi ve bu kaynakları daha esnek bir şekilde kullanmayı sağlar
- Bakım ve yönetim süreçlerini sadeleştirir ve kolaylaştırır
- CEPH , GlusterFS, OpenEBS, Longhorn

Traditional vs. SDS





ceph



Açık Kaynak



Esneklik ve Çok
Yönlülük



Kendi Kendini İyileştirme

Dağıtık Mimari



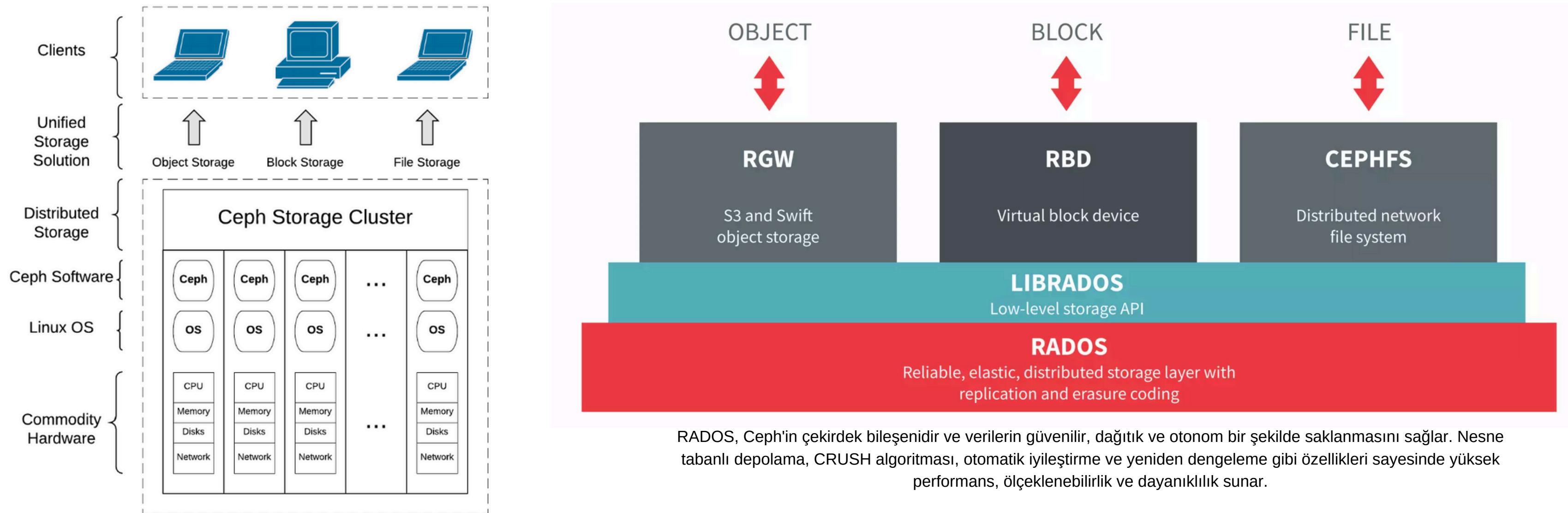
Yüksek Erişilebilirlik



Yüksek Ölçeklenebilirlik

Neden Ceph

Ceph Mimarisi



RADOS Bileşenleri



MON Monitors

CEPH'in merkezi yönetim ve izleme bileşenidir. CEPH kümesinin genel sağlığını izler, kritik yapılandırma bilgilerini saklar ve veri tutarlığını sağlarlar. Ceph'in yüksek erişilebilirlik ve hata toleransı sunan bir depolama sistemi olmasındaki temel bileşendir.



OSD Object Storage Daemons

Verilerin saklandığı ve yönetildiği bileşendir. Her OSD daemon, bir disk veya disk bölümü üzerinde çalışır ve verilerin replikasyonunu, iyileştirilmesini ve veri iyileştirme süreçlerini yürütür. Ceph, veri dayanıklılığı ve yüksek erişilebilirlik sağlamak için birden fazla kopya (replica) kullanılır.



MDS Metadata Servers

Ceph File System (CephFS) için dosya meta verilerini yönetir. Dağıtık meta veri yönetimi, yük dengeleme, failover ve yüksek erişilebilirlik özellikleriyle, MDS sunucuları CephFS'nin performansını ve güvenilirliğini artırır.



MGR Manager Daemons

Ceph kümesinin yönetim, izleme ve performans analizinden sorumludur bileşenleridir. Cluster'ın durumunu ve performansını izler, sağlık kontrolleri yapar ve çeşitli yönetim API'leri sağlarlar. Modüler yapıları ve eklentileri sayesinde esnek ve özelleştirilebilir bir yönetim çözümü sunarlar. Uyarı yönetimi özelliği ile yöneticilerin CEPH'i etkin bir şekilde yönetmesine olanak tanır ve olası sorunları hızlıca tespit edip çözmelerine yardımcı olur.

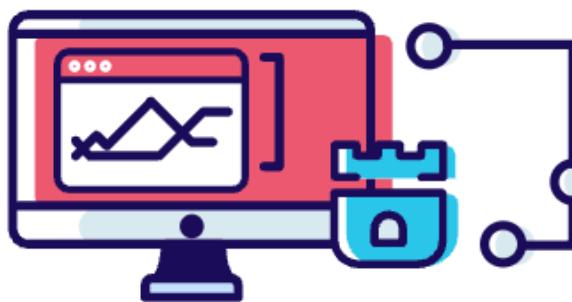


RGW RADOS Gateway

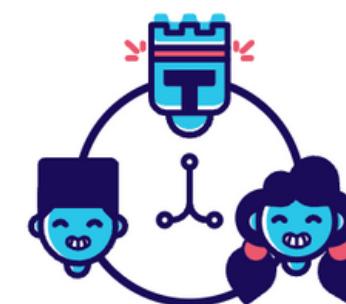
Ceph'in nesne depolama hizmeti sunan bileşenidir. S3 ve Swift protokollerini destekler. Kimlik doğrulama, yetkilendirme, veri replikasyonu ve hata toleransı gibi özelliklerle yüksek performanslı ve güvenilir nesne depolama hizmeti sunar.

Rook Operatörü

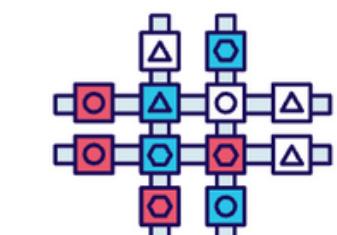
Rook Ceph operatörü, Kubernetes üzerinde Ceph kümelerini otomatik olarak dağıtmak, yönetmek ve ölçeklendirmek için kullanılan bir operatördür



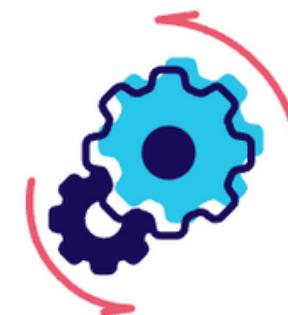
Açık kaynaklı Ceph depolama alanını yönetir



Apache 2.0 lisansı altında yayımlanan açık kaynaklı yazılım



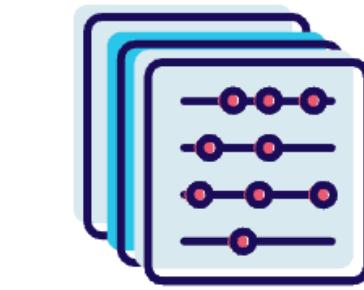
file, block, ve object Provision



Basit ve güvenilir otomatik depolama yönetimi



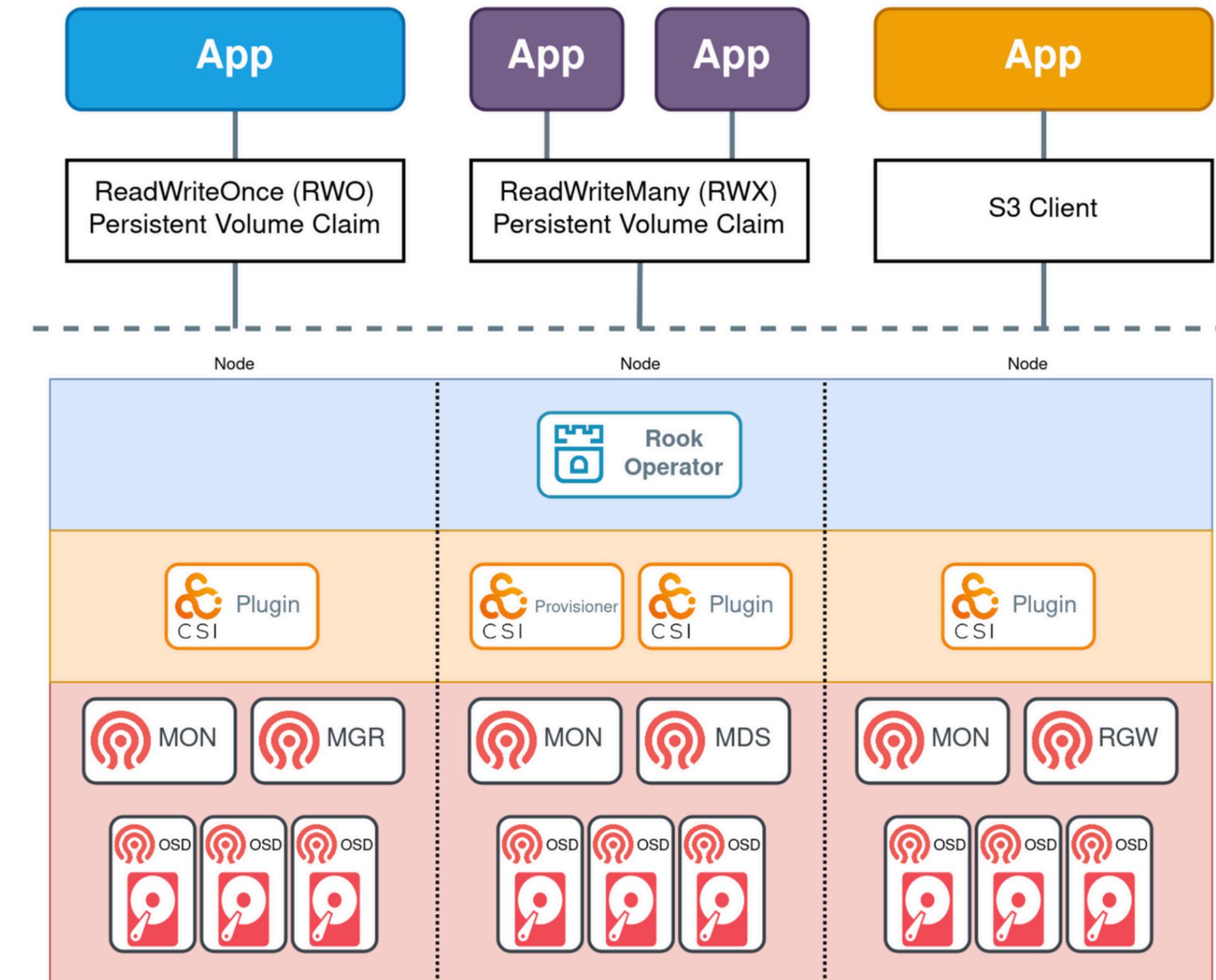
Veri kaybını önlemek için verileri çoklar ve küme içerisinde dengeli şekilde dağıtır



Depolama kümelerinde kolay ölçeklendirilme ve yönetim sağlar



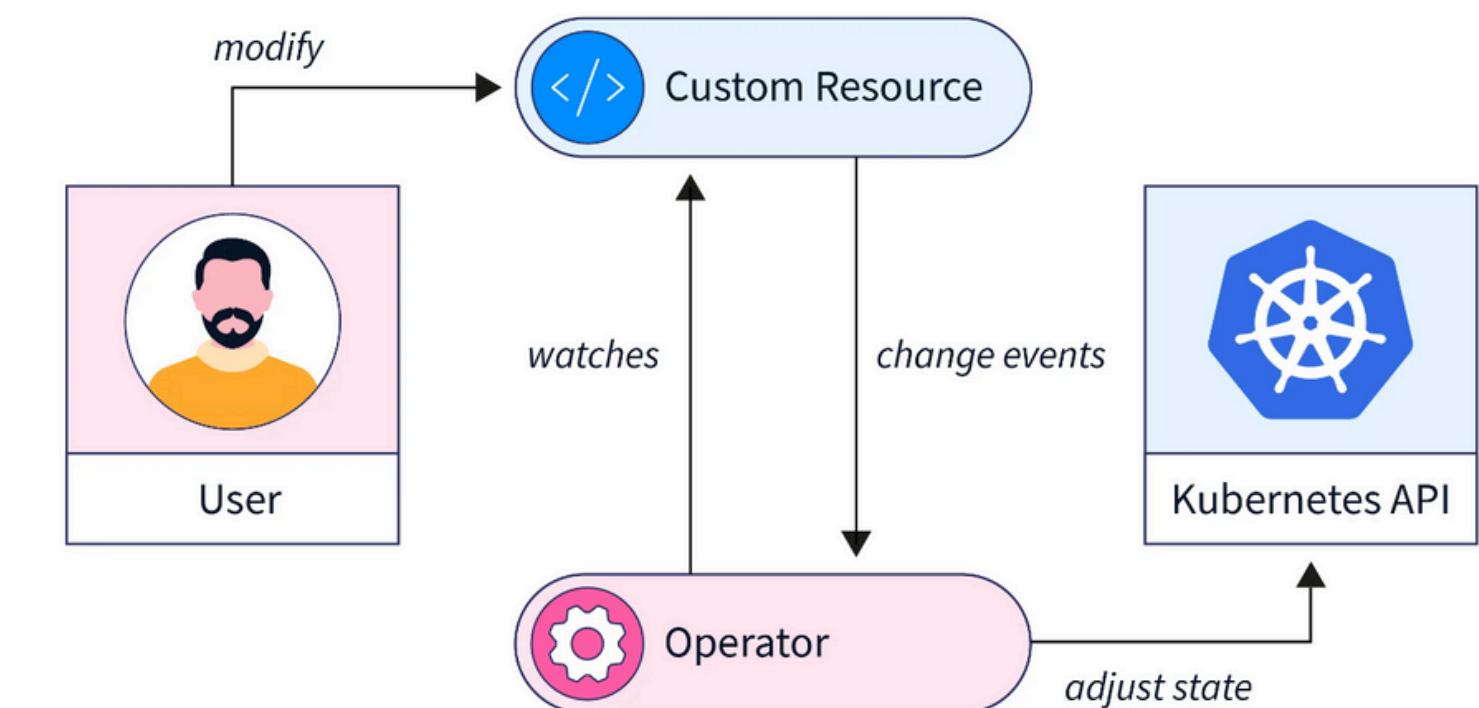
Rook Mimarisi



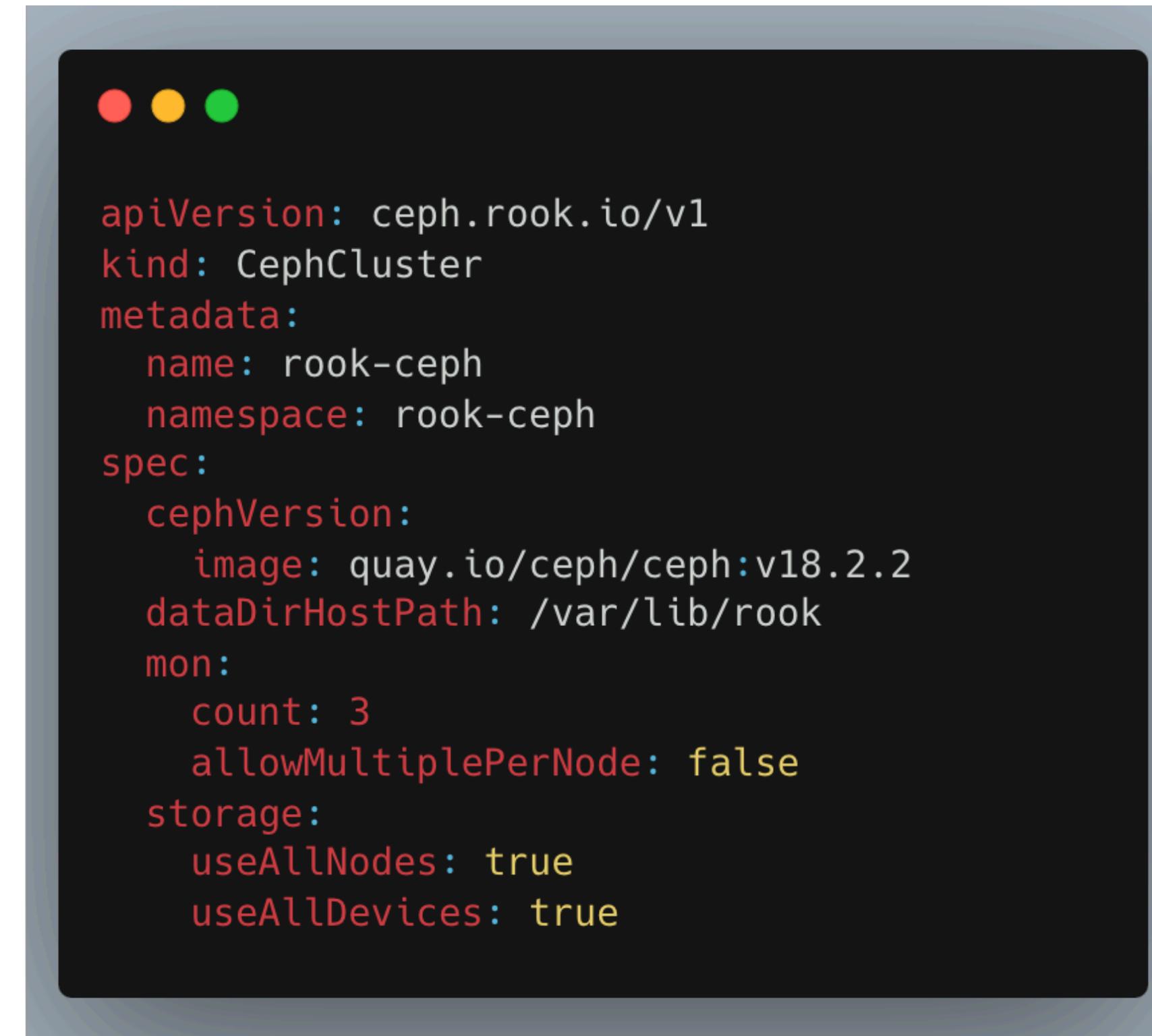
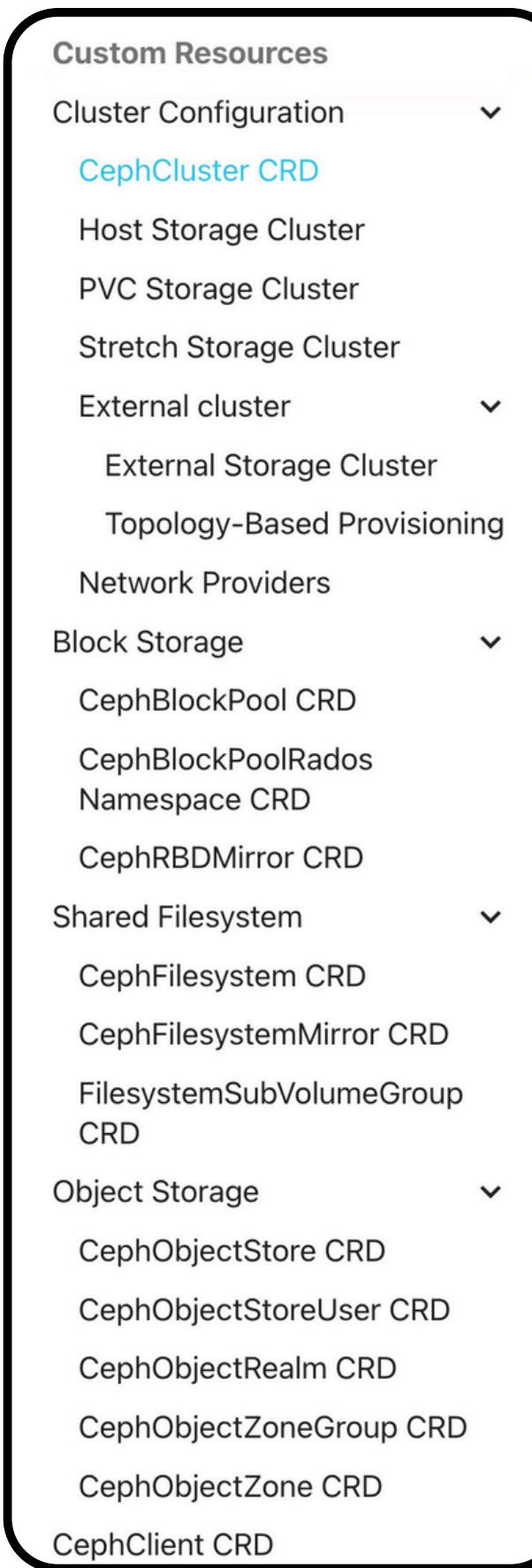
Kubernetes Operator Pattern

Kubernetes Operator Pattern, Kubernetes üzerinde özel kaynakların (Custom Resource Definitions - CRD) ve onların yaşam döngülerinin otomatik olarak yönetilmesini sağlayan bir yazılım tasarım desenidir.

- **Custom Resource Definitions (CRD):** Kubernetes API'sine yeni kaynak türleri eklemenizi sağlar. Pod, Service, Deployment gibi temel kaynak türlerinin ötesinde uygulamanıza özgü özel kaynak türleri oluşturmayı sağlar.
- **Controller:** Kubernetes'in kontrol döngülerini kullanarak kaynakların durumunu izler ve gerekiğinde bu kaynakların durumunu istenen hale getirir.
- **Operator:** Bir operatör, belirli bir uygulamanın veya servislerin Kubernetes üzerinde otomatik olarak kurulmasını, yapılandırılmasını, yönetilmesini ve ölçeklendirilmesini sağlayan özel bir Kubernetes controller'ıdır. Operatörler, CRD'ler ve controller'lar kullanılarak oluşturulur.



Rook Operator CRDs



The image displays a JSON configuration for a CephCluster resource. It includes fields for apiVersion, kind, metadata (name and namespace), spec (cephVersion, image, dataDirHostPath, mon count, allowMultiplePerNode), and storage (useAllNodes, useAllDevices). The configuration is set up for a three-node monitor cluster using the quay.io/ceph/ceph:v18.2.2 image and storing data in /var/lib/rook.

```
apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph
  namespace: rook-ceph
spec:
  cephVersion:
    image: quay.io/ceph/ceph:v18.2.2
  dataDirHostPath: /var/lib/rook
  mon:
    count: 3
    allowMultiplePerNode: false
  storage:
    useAllNodes: true
    useAllDevices: true
```

<https://rook.io/docs/rook/latest-release/CRDs/Cluster/ceph-cluster-crd/>

- **Sağlık Kontrolleri ve Otomatik Failover:** MON'ların sağlık kontrollerini yapar ve otomatik olarak failover sağlar. Bir MON podunun arızalanması durumunda diğer podun devreye girerek kesintisiz hizmeti sağlar.
- **Kubernetes ile Basitleştirilmiş Yönetim:** Ceph Cluster, Pools , Filesystem ve RGW (RADOS Gateway) yönetimini Kubernetes objeleri aracılığıyla basitleştirir. Bu sayede, bu bileşenlerin yönetimi ve yapılandırması daha kullanıcı dostu ve verimli hale gelir.
- **Otomatik Ölçeklendirme:** Rook, depolama cluster'larının ihtiyaçlara göre otomatik olarak ölçeklenmesini destekler. Bu, kaynakların dinamik olarak eklenip çıkarılmasını ve böylece depolama kapasitesinin ve performansının optimize edilmesini sağlar.
- **Otomatik Güncellemeler ve Bakım:** Rook, Ceph cluster'larının güncellenmesi ve bakımı süreçlerini otomatikleştirir. Bu, güncellemelerin kesintisiz ve güvenli bir şekilde yapılmasını sağlar.

Rook Helm ile Kurulum

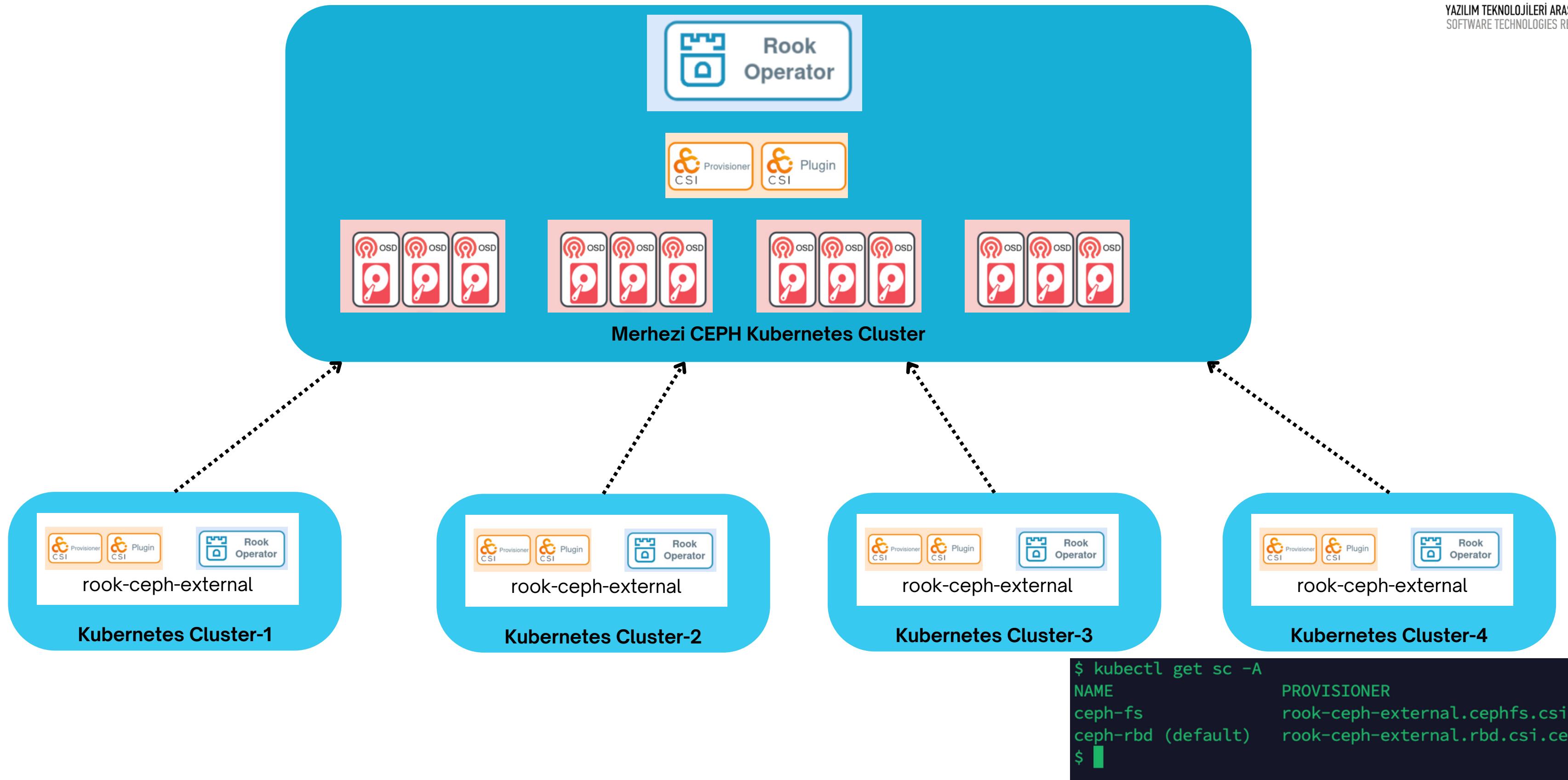
- **Rook Ceph Operator:** Ceph CR'leri izleyecek olan Ceph Operatörünü başlatır.

```
helm repo add rook-release https://charts.rook.io/release  
helm install --create-namespace --namespace rook-ceph \  
rook-ceph rook-release/rook-ceph -f values.yaml
```

- **Rook Ceph Cluster:** Operatörün kümeyi yapılandırmak için kullanacağı Ceph CR'leri oluşturur

```
helm install --create-namespace --namespace rook-ceph rook-ceph-cluster \  
--set operatorNamespace=rook-ceph rook-release/rook-ceph-cluster -f values.yaml
```

Rook-CEPH External



Performans Testi

```
<<K9s-Shell>> Pod: rook-ceph/rook-ceph-tools-8cb469857-zsr5n | Container: rook-ceph-tools
```

```
sbash-4.4$ ceph osd pool create testbench 100 100
pool 'testbench' created
bash-4.4$ rados bench -p testbench 10 write --no-cleanup
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 for up to 10 seconds or 0 objects
Object prefix: benchmark_data_node3_1541180
  sec Cur ops  started finished avg MB/s cur MB/s last lat(s) avg lat(s)
    0     0      0       0      0        0      -          0
    1    16     96     80  320.006   320  0.0465806  0.192949
    2    16    154    138 275.966   232  0.105978  0.225487
    3    16    202    186 247.965   192  0.178942  0.25092
    4    16    250    234 233.952   192  0.205582  0.264098
    5    16    300    284 227.152   200  0.160203  0.275355
    6    16    351    335 223.288   204  0.490098  0.276027
    7    16    391    375 214.244   160  0.179042  0.276287
    8    16    444    428 213.959   212  0.186373  0.28371
    9    16    510    494 219.514   264  0.0737926  0.280571
   10   16    561    545 217.96   204  0.227777  0.281364
Total time run: 10.179
Total writes made: 561
Write size: 4194304
Object size: 4194304
Bandwidth (MB/sec): 220.453
Stddev Bandwidth: 44.9494
Max bandwidth (MB/sec): 320
Min bandwidth (MB/sec): 160
Average IOPS: 55
Stddev IOPS: 11.2373
Max IOPS: 80
Min IOPS: 40
Average Latency(s): 0.289991
Stddev Latency(s): 0.164236
Max latency(s): 0.839659
Min latency(s): 0.0465806
bash-4.4$ rados bench -p testbench 10 seq
hints = 1
  sec Cur ops  started finished avg MB/s cur MB/s last lat(s) avg lat(s)
    0     0      0       0      0        0      -          0
    1    16     100    84  335.836   336  0.412773  0.129464
    2    16    216    200 399.866   464  0.104178  0.155205
    3    16    390    374 498.296   696  0.0470498  0.125594
    4    16    465    449 448.731   300  0.0814951  0.137398
Total time run: 4.80949
Total reads made: 561
Read size: 4194304
Object size: 4194304
Bandwidth (MB/sec): 466.578
Average IOPS: 116
Stddev IOPS: 44.7689
Max IOPS: 174
Min IOPS: 75
Average Latency(s): 0.134667
Max latency(s): 0.760522
Min latency(s): 0.0396309
15360+0 records in
15360+0 records out
16106127360 bytes (15.0GB) copied, 83.723972 seconds, 183.5MB/s
```

```
bash-4.4$ rados bench -p testbench 10 rand
hints = 1
```

```
sec Cur ops  started finished avg MB/s cur MB/s last lat(s) avg lat(s)
  0     0      0       0      0        0      -          0
  1    16     16     234    218 871.698   872  0.0953416  0.0652045
  2    16     16     437    421 839.673   812  0.0829836  0.0665378
  3    16     16     620    604 803.811   732  0.0470591  0.0766891
  4    16     16     827    811 809.764   828  0.0584332  0.0756201
  5    16     15     981    966 771.564   620  0.0427701  0.0739163
  6    16     16    1122   1106 736.332   560  0.0307369  0.0716529
  7    15     15    1261   1246 711.113   560  0.0398889  0.0877857
  8    15     15    1439   1424 710.971   712  0.0471763  0.087699
  9    15     15    1637   1622 719.835   792  0.0158774  0.0860241
 10   12     12    1809   1797 717.585   700  0.0919466  0.0847757
Total time run: 10.8416
Total reads made: 1809
Read size: 4194304
Object size: 4194304
Bandwidth (MB/sec): 667.43
Average IOPS: 166
Stddev IOPS: 27.6649
Max IOPS: 218
Min IOPS: 140
Average Latency(s): 0.0885391
Max latency(s): 3.08793
Min latency(s): 0.00924889
bash-4.4$ rados bench -p testbench 10 write -t 4 --run-name client1
hints = 1
Maintaining 4 concurrent writes of 4194304 bytes to objects of size 4194304 for up to 10 seconds or 0 objects
Object prefix: benchmark_data_node3_1541268
  sec Cur ops  started finished avg MB/s cur MB/s last lat(s) avg lat(s)
    0     0      0       0      0        0      -          0
    1     4      4      60      56 223.982   224  0.0716391  0.0630988
    2     4      4     110     106 211.977   200  0.0680417  0.0711791
    3     4      4     160     156 207.973   200  0.0501653  0.0747629
    4     4      4     210     206 205.973   200  0.0539513  0.0772217
    5     4      4     247     243 194.374   148  0.0443614  0.0792209
    6     4      4     293     289 192.641   184  0.0337323  0.0768245
    7     4      4     329     325 185.69    144  0.0491063  0.0854382
    8     4      4     382     378 188.975   212  0.0682203  0.0842618
    9     4      4     432     428 190.186   200  0.0417132  0.0839482
   10    4      4     480     476 190.364   192  0.0736124  0.0836337
Total time run: 10.0143
Total writes made: 480
Write size: 4194304
Object size: 4194304
Bandwidth (MB/sec): 191.726
Stddev Bandwidth: 25.7302
Max bandwidth (MB/sec): 224
Min bandwidth (MB/sec): 144
Average IOPS: 47
Stddev IOPS: 6.43256
Max IOPS: 56
Min IOPS: 36
Average Latency(s): 0.0834
Stddev Latency(s): 0.127823
Max latency(s): 2.56787
Min latency(s): 0.0290035
Cleaning up (deleting benchmark objects)
Removed 480 objects
Clean up completed and total clean up time :0.959901
bash-4.4$ 
```

 **rook-ceph-external-helm-stack** Public

 Unpin  Unwatch 1  Fork 0  Star 0

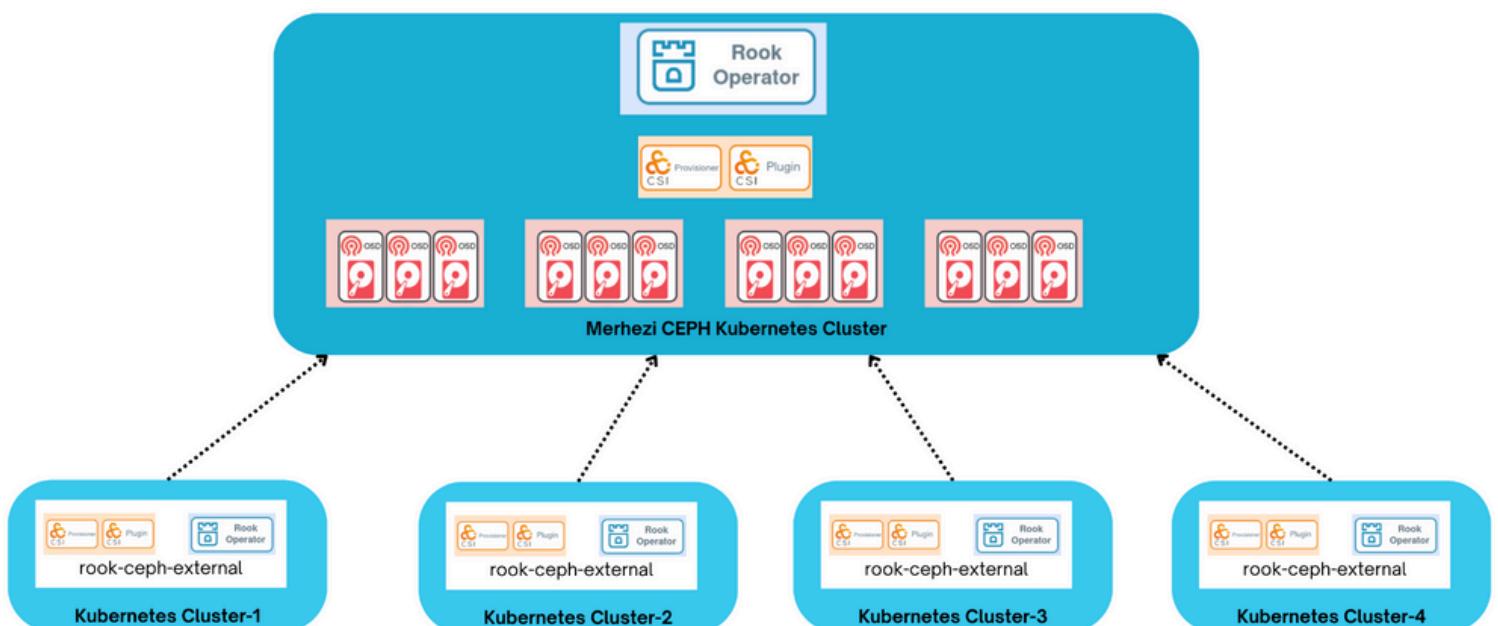
 main  1 Branch  0 Tags  Go to file  Add file 

Berkan YILDIRIM Initial commit d09cbfe · 4 minutes ago 2 Commits

 deploy	Initial commit	4 minutes ago
 img	Initial commit	4 minutes ago
 LICENSE	Initial commit	7 minutes ago
 README.md	Initial commit	4 minutes ago
 install-external.sh	Initial commit	4 minutes ago
 uninstall-external.sh	Initial commit	4 minutes ago

 README  Apache-2.0 license

Rook - CEPH Helm-stack

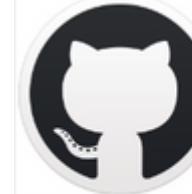
About

rook-ceph-external-helm-stack

-  [Readme](#)
-  [Apache-2.0 license](#)
-  [Activity](#)
-  [0 stars](#)
-  [1 watching](#)
-  [0 forks](#)

Releases

No releases published [Create a new release](#)

 [berkanyildirim/rook-ceph-external-helm-stack: rook-ceph-external-helm-stack](#)
rook-ceph-external-helm-stack. Contribute to berkanyildirim/rook-ceph-external-helm-stack...
GitHub

Packages

No packages published [Publish your first package](#)

Languages



● Python 81.9% ● Shell 18.1%



YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ
SOFTWARE TECHNOLOGIES RESEARCH INSTITUTE

TEŞEKKÜRLER

