

Finansal Doküman Yönetiminde Yeni Nesil Mimari

Ölçeklenebilirlik, Verimlilik ve Güvenlik için
Tasarlanmış Akıllı Sıkıştırma ve Depolama Stratejisi

[Internal Project Name/Code] | [Date]

Karşı Karşıya Olduğumuz Veri Yönetimi Zorluğu

Mevcut doküman altyapımız, hem büyümeye hızımıza hem de yasal uyumluluk gereksinimlerimize cevap vermekte zorlanıyordu. Bu durum, stratejik bir mimari değişikliğini zorunlu kıladı.



Veri Hacmi

Günde ortalama **10 GB** yeni doküman (PDF, JPG/JPEG formatında yasal sözleşmeler).



Miras Altyapı

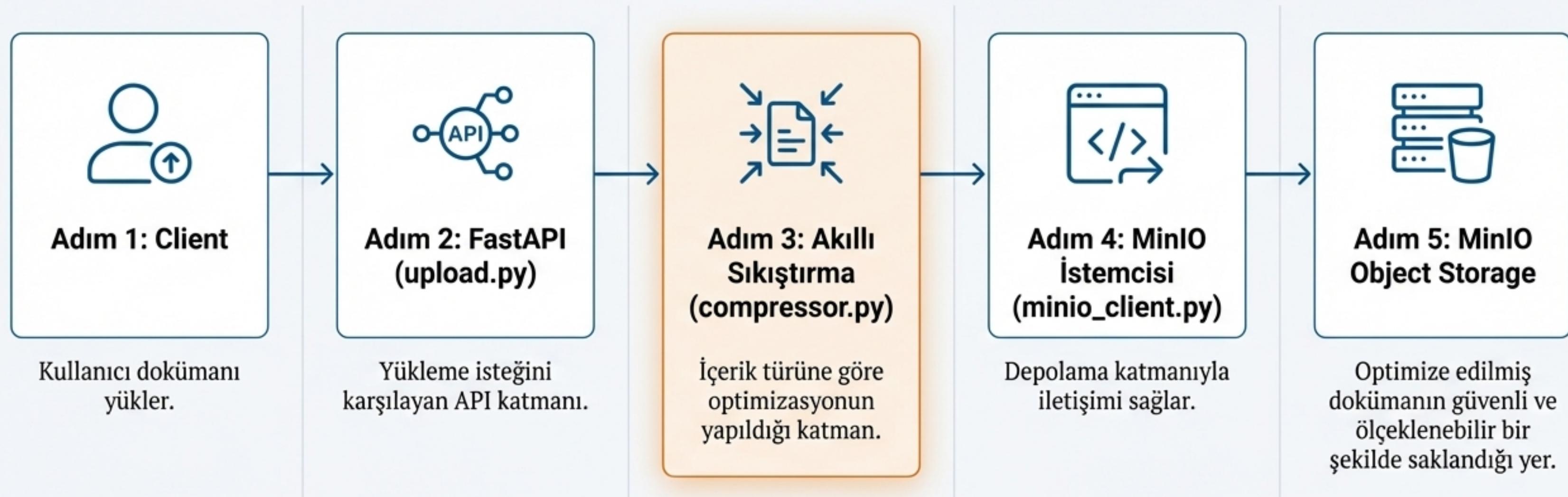
Çok sayıda bağlı diskten (**K, L, M sürücüler**) taşınması gereken büyük bir doküman arşivi.



İş Gereksinimleri

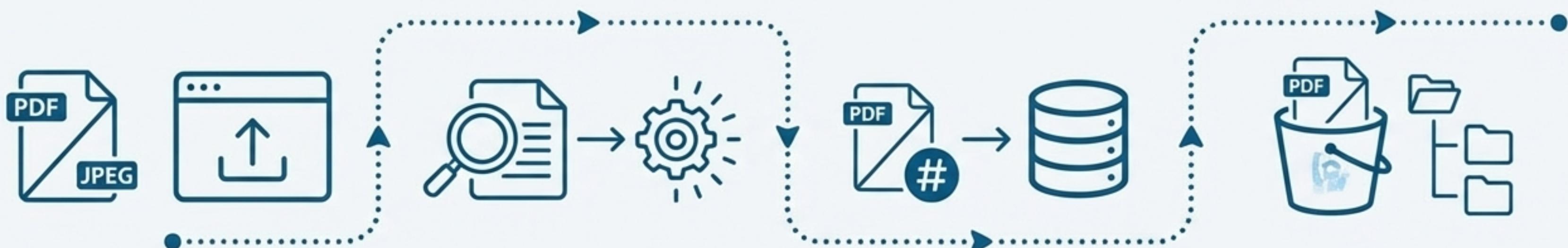
Finansal veriler için kurumsal düzeyde güvenlik, değişmezlik (*immutability*) ve denetlenebilirlik.

Geliştirdiğimiz Yüksek Verimli Doküman İşleme Mimarisi



Her bileşen, performansı ve güvenilirliği en üst düzeye çıkarmak için özenle seçilmiş ve entegre edilmiştir. Sistem, dokümanları şeffaf bir şekilde işler ve kullanıcı deneyimini korur.

Bir Dokümanın Yolculuğu: Yüklemeden Depolamaya



YÜKLEME

Kullanıcı, arayüz üzerinden PDF veya JPEG dosyasını yükler. Sistem, FastAPI endpoint'i ile dosyayı alır.

ANALİZ & SIKIŞTIRMA

`compressor.py` devreye girer. Dosya içeriği analiz edilir (JPEG mi, metin tabanlı PDF mi, taranmış PDF mi?). Türüne özel optimizasyon algoritması çalıştırılır.

DEPOLAMA

Optimize edilmiş dosya, `minio_client.py` aracılığıyla MinIO'ya gönderilir. Dosya bütünlüğü için hash bilgisi hesaplanır.

SAKLAMA

MinIO, dosyayı belirlenen depolama politikalarına göre saklar. Dosya yolu yapısı korunur:
`disk/user_uuid/timestamp/upload_pdf/file`

Kullanıcı için sıkıştırma süreci tamamen şeffaftır. Dosyalar ZIP veya Base64 gibi formatlara dönüştürülmez, orijinal kullanıcı deneyimi korunur.

Akıllı Sıkıştırmanın Arkasındaki Strateji

Compression is content-aware and non-destructive.

Sıkıştırma işlemi içeriğe duyarlıdır ve veri kaybına yol açmaz. Amacımız, dosya kalitesini kullanıcıların fark edemeyeceği bir seviyede tutarken depolama alanından maksimum tasarruf sağlamaktır.

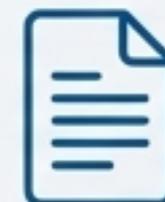


JPEG / JPG Dosyaları

Yaklaşım Agresif yeniden kodlama.

Yöntem Kalite tabanlı sıkıştırma (Quality 75) ve gereksiz metadata'ların temizlenmesi.

Sonuç Maksimum boyut kazancı.



PDF Dosyaları

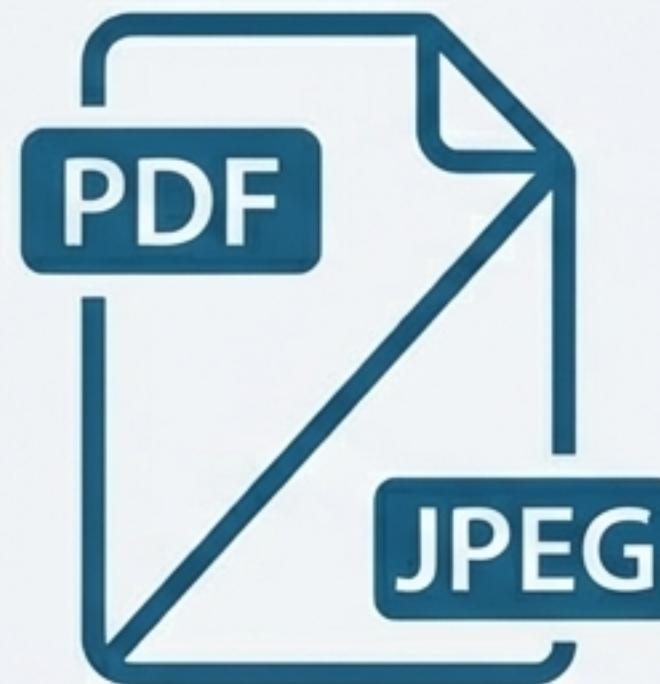
Yaklaşım Yapısal optimizasyon.

Yöntem Metin tabanlı PDF'ler için `pikepdf` ile yapısal temizlik. Görüntü içerenler için potansiyel Ghostscript entegrasyonu.

Sonuç Dosyanın içeriğine bağlı olarak değişken kazanç.

JPEG Optimizasyonu: Depolamada Büyük Kazanım

ÖNCE



158,850 bytes (~155 KB)

SONRA



17,831 bytes (~17 KB)

Sonuç Analizi

- Bu sonuç, JPEG işleme hattımızın (pipeline) mükemmel çalıştığını kanıtlıyor.
- Kalite 75 ayarı ve metadata temizliği, kullanıcıların görsel kalite farkı algılamadığı, ancak depolama maliyetlerini dramatik şekilde düşürdüğü ‘tatlı noktayı’ oluşturuyor.

PDF Optimizasyonu: Akıllı ve Ölçülü Yaklaşım

Neden PDF'ler JPEG Kadar Küçülmeli?

Metin Ağırlıklı PDF'ler

Bu dosyalar zaten sıkıştırılmış metin verileri içerir. `pikepdf` ile yapılan yapısal temizlik, genellikle **%0-5** arasında kazanç sağlar. Bu anormal bir durum değil, beklenen bir sonuctur.

*Örnek: 658 KB'lık bir dosyada ~%1.5 kazanç.



Taranmış (Görüntü İceren) PDF'ler

Asıl potansiyel buradadır. Mevcut altyapı bu dosyalara dokunmaz. Gelecekteki Ghostscript entegrasyonu ile bu tür dosyalarda **%30-60** arasında kazanç hedeflenmektedir.



Örnek Kazanç (Metin Ağırlıklı PDF'den daha iyi bir sonuç)

Orijinal Boyut: **658.288 byte**

Optimize Edilmiş Boyut: **257.675 byte**

Sıkıştırma Oranı: **~%60,87**

Not: PDF sıkıştırma oranları, içeriğe bağlı olarak %1 ile %75 arasında geniş bir yelpazede değişebilir.

Teknik Detay: PDF İşlemede Stabiliteyi Sağlayan Kod İyileştirmesi

İlk denemelerde, bazı PDF dosyaları işlenirken sistem `500 Internal Server Error` hatası veriyordu. Sorunun kaynağı, `pikepdf` kütüphanesinin belirli dosyalarda istisna fırlatan bir parametresiydi.

ÖNCE (Hatalı Kod)

```
- pdf.save(  
-     output_io,  
-     optimize_streams=True, # ❌ BU SATIR HATALARA NEDEN OLUŞTU  
-     compress_streams=True  
- )
```

SONRA (Kararlı Çözüm)

```
+ def optimize_pdf(data: bytes) -> bytes:  
+     # ... (try/except bloğu içinde)  
+     with pikepdf.open(input_io) as pdf:  
+         pdf.save(output_io, compress_streams=True)  
+     optimized = output_io.getvalue()  
+     # Sadece dosya boyutu küçüldüyse optimize edilmiş  
+     # halini döndür  
+     return optimized if len(optimized) < len(data) else  
+         data
```

Bu basit ama kritik değişiklik, PDF işleme sürecini %100 kararlı hale getirdi ve olası tüm hataları ortadan kaldırdı.

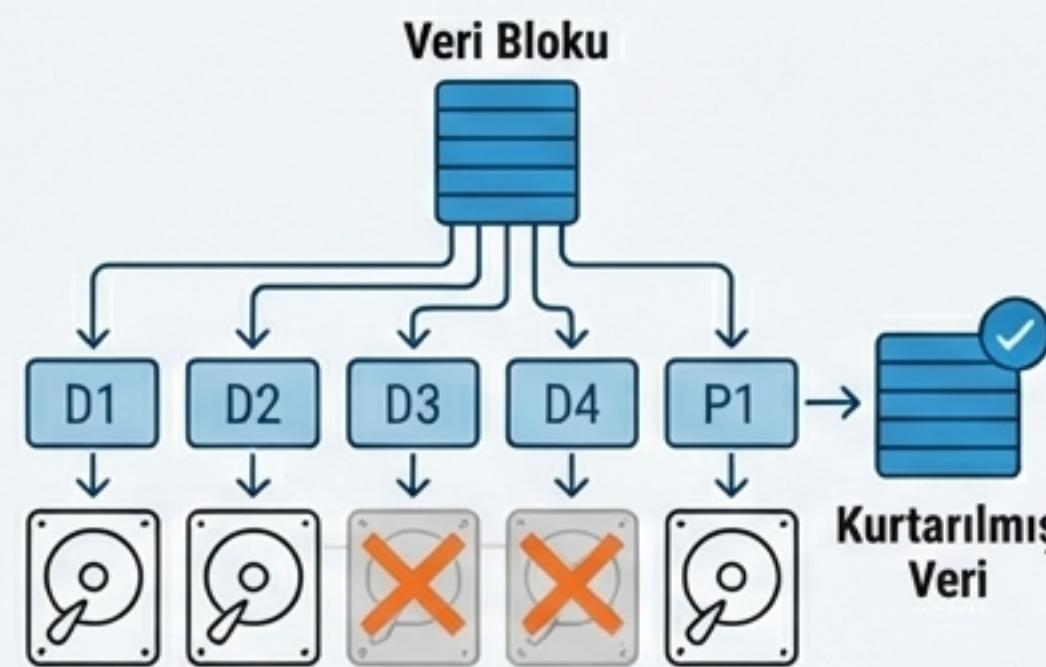
Depolama Altyapımızın Temeli: Neden MinIO?



- **S3 API Uyumluluğu:** Endüstri standartı olan S3 API'ını desteklemesi, mevcut araçlar ve SDK'lar ile sorunsuz entegrasyon sağlar. Gelecekte bulut geçiş senaryolarını basitleştirir.
- **Yüksek Performans ve Ölçeklenebilirlik:** Dağıtık yapısı sayesinde, yeni diskler veya sunucular eklenerek depolama kapasitesi ve performans yatay olarak kolayca ölçeklendirilebilir.
- **Kurumsal Düzeyde Özellikler:** Veri şifreleme, yaşam döngüsü yönetimi (lifecycle management), versiyonlama ve değişmezlik (object locking) gibi finansal sistemler için kritik olan özellikleri yerel olarak destekler.
- **Kendi Altyapımızda Kontrol (On-Premise):** Verilerimizin tamamen kendi kontrolümüz altındaki bir altyapıda barındırılmasına olanak tanır.

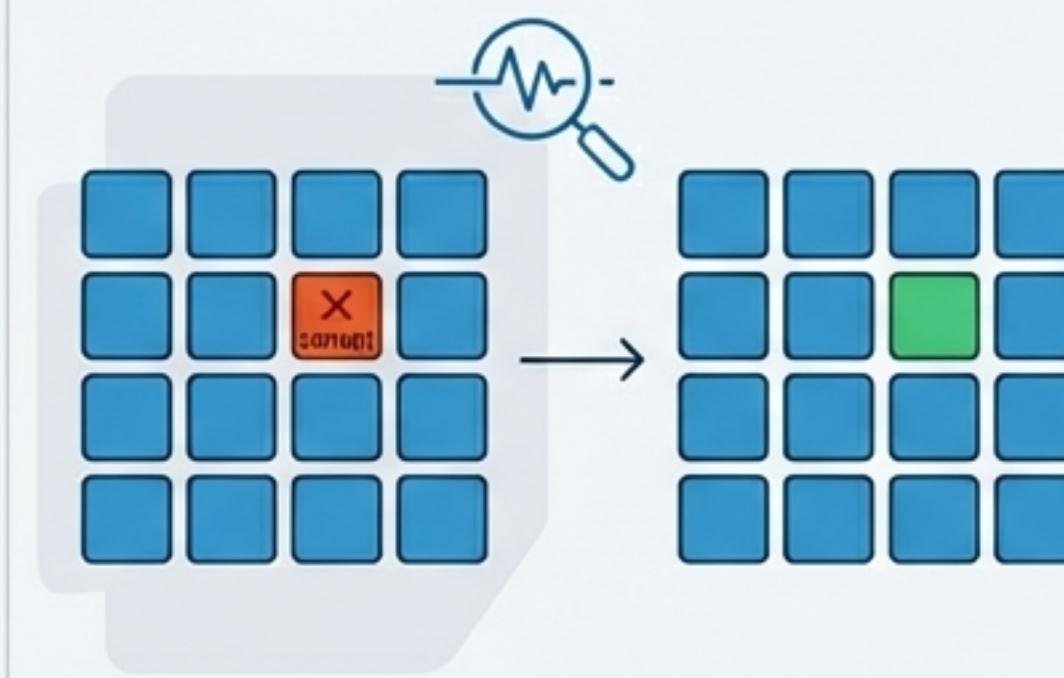
Dayanıklılık ve Ölçek için Tasarlanmış Çekirdek Kavramlar

Erasure Coding → Veri Dayanıklılığı



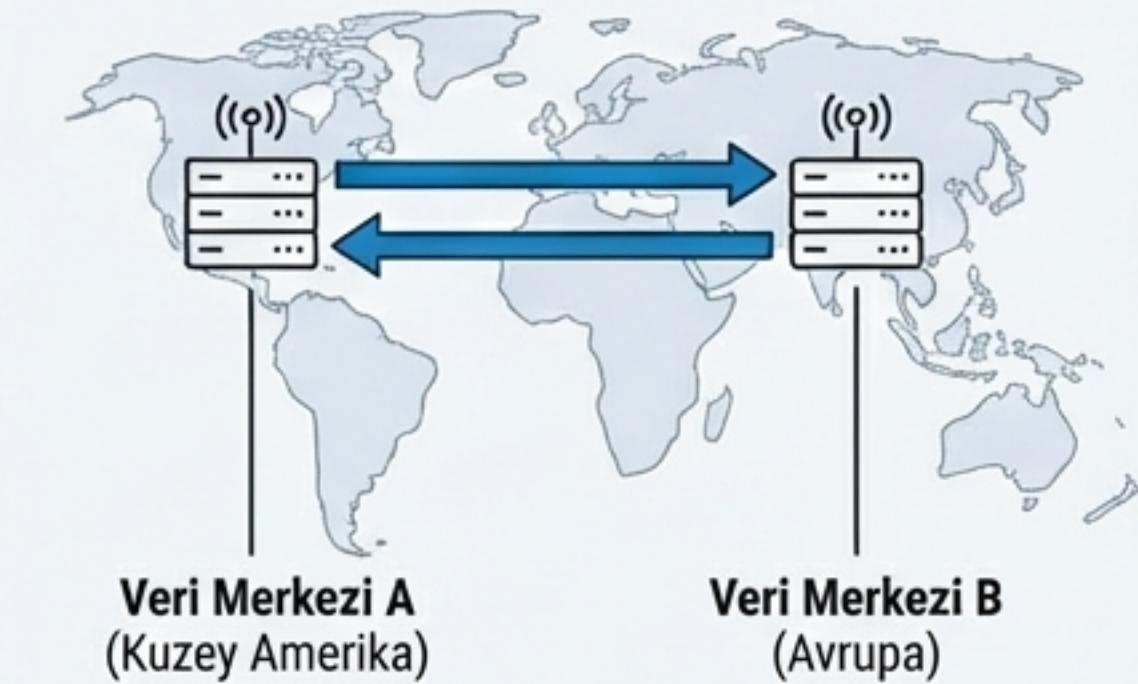
Verilerinizi donanım arızalarına karşı korumak için diskler arasında parçalara ayırarak dağıtır. Geleneksel RAID'den daha verimli ve güvenlidir. Birden fazla disk arızalansa bile veri kaybı yaşanmaz.

Healing → Otomatik İyileşme



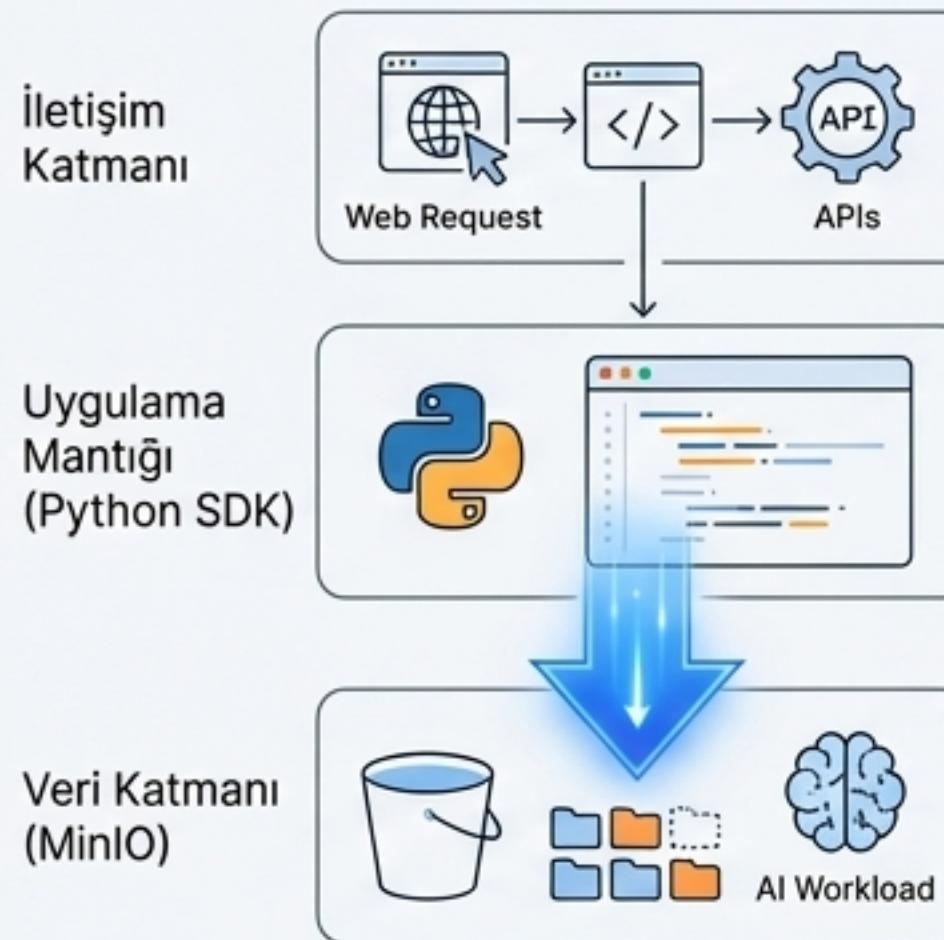
Sistem, kayıp veya bozuk veri parçalarını proaktif olarak tespit eder ve arka planda, kesinti olmaksızın otomatik olarak onarır. Bu, veri bütünlüğünü sürekli olarak garanti altına alır.

Replication → Coğrafi Yedeklilik



Felaket kurtarma senaryoları için verilerinizi farklı coğrafi bölgelerdeki siteler arasında aktif-aktif veya aktif-pasif modda kopyalar. Bu sayede bir veri merkezinin tamamen hizmet dışı kalması durumunda bile operasyonlar devam edebilir.

Uygulama ve Veri Katmanı Arasındaki Köprü: Python SDK



MinIO Python Client SDK, uygulamamızın (FastAPI) depolama katmanıyla (MinIO) sorunsuz ve verimli bir şekilde iletişim kurmasını sağlayan üst düzey API'ler sunar.

file_uploader.py

```
# 1. İstemci Oluşturma
client = Minio("play.min.io",
                access_key="...",
                secret_key="...")

# 2. Bucket Kontrolü ve Oluşturma
found = client.bucket_exists(bucket_name)
if not found:
    client.make_bucket(bucket_name)

# 3. Nesne (Dosya) Yükleme
client.fput_object(bucket_name,
                    destination_file,
                    source_file)
```

1. İstemci Oluşturma

Minio() metodu ile sunucu adresi ve kimlik bilgileri kullanılarak bağlantı nesnesi oluşturulur.

2. Bucket Yönetimi

bucket_exists() ile kovanın varlığı kontrol edilir, make_bucket() ile oluşturulur.

3. Nesne Yükleme

fput_object() ile yerel dosya, belirtilen kovaya ve yeni bir adla yüklenir.

(i) Gerekli Python Sürümü: 3.7+

Projenin Etkisi: Performans ve Verimlilik Karnesi

Depolama Tasarrufu (JPEG)

~%88

Görüntü dosyalarında elde edilen ortalama depolama alanı kazancı.

Depolama Tasarrufu (PDF)

%1 - %75

İçeriğe bağlı olarak PDF dosyalarında elde edilen değişken kazanç aralığı.

Sistem Güvenilirliği

Yüksek

MinIO'nun Erasure Coding ve Otomatik İyileşme (Healing) özellikleri sayesinde donanım arızalarına karşı tam koruma.

Ölçeklenebilirlik

Yatay (Horizontal)

Yeni donanım eklenecek kapasitenin kesintisiz olarak artırılabilmesi.

Kullanıcı Deneyimi

Korundu

Sıkıştırma işlemi son kullanıcı için tamamen şeffaf ve kesintisizdir.

Gelecek Vizyonu ve Yol Haritası

Mevcut mimari, gelecekteki iyileştirmeler için sağlam bir zemin sunmaktadır. Değer katacak bir sonraki adımlar için potansiyel odak alanlarımız şunlardır:



Ghostscript Destekli PDF İşleme Hattı

Hedef: Taranmış (scan) ve yoğun görüntü içeren PDF'lerde %30-60 ek depolama kazancı sağlamak. iLovePDF benzeri bir optimizasyon yeteneği kazandırmak.



B) Eşik Değer (Threshold) Kuralları

Hedef: Verimliliği artırmak için akıllı kurallar eklemek. Örneğin: 1 MB'tan küçük PDF'leri sıkıştırma adımını atlayarak (skip) işlem süresini kısaltmak.



C) Sıkıştırma Raporu Endpoint'i

Hedef: Sistemin ne kadar tasarruf sağladığını anlık olarak takip edebilmek için günlük/haftalık kazanç raporları sunan bir API endpoint'i oluşturmak.

Sonuç: Geleceğe Hazır, Akıllı Bir Doküman Platformu

Bu proje ile sadece güncel bir veri yönetimi sorununu çözmekle kalmadık; aynı zamanda şirketin gelecekteki ihtiyaçlarına cevap verebilecek, yüksek verimli, ölçeklenebilir ve güvenli bir doküman yönetim platformunun temellerini attık.



Maliyet Etkin

Depolama maliyetlerini, özellikle görüntü ağırlıklı dosyalarda, önemli ölçüde azaltır.



Güvenli ve Uyumlu

Kurumsal düzeyde veri koruma ve denetlenebilirlik özellikleri sunar.



Ölçeklenebilir

Gelecekteki veri büyümeyi sorunsuz bir şekilde karşılamaya hazırır.