**🌩️ “Kendi Bulut Platformunu Yapmak” Ne Demek?**

**🎯 Hedef 1: Standart Kullanıcıya Hizmet**

Dosya saklasın, paylaşsın, yedek tutsun, basit arayüz olsun  
📦 Google Drive / Dropbox tarzı deneyim

**🎯 Hedef 2: Developer’a Hizmet**

API erişimi, webhook, SDK, belki CI/CD gibi şeyler  
☁️ Firebase / AWS S3 + Lambda gibi servisler

**🔧 Peki Neler Olmalı Bu Platformda?**

**🔐 Kullanıcı Giriş & Yetki (IAM mantığı)**

* Kullanıcı → Token → API erişimi
* Yetkilendirme: sadece kendi dosyasını görebilsin

**📁 Dosya Saklama (S3 mantığı)**

* Upload, Download, Expiring Link
* Bucket mantığı (kullanıcı bazlı klasör sistemi)
* MinIO veya doğrudan local+NGINX ile başlayabilirsin

**🧠 Arkaplan Görevleri (Lambda mantığı)**

* Kullanıcı dosya yükledi → “convert et”, “resize et”, “virüs tara” vs.
* Celery + Redis tam burada devreye girer

**🧪 Developer Friendly Özellikler**

* REST API veya GraphQL endpoint'leri
* Swagger veya ReDoc ile dokümantasyon
* Webhook sistemi (bir dosya yüklendiğinde senin sunucuna haber verilsin)
* Token bazlı SDK erişimi (örnek: Python, JS client)

**🧰 Admin Panel / Dashboard**

* Django Admin ile başlayabilirsin
* Daha sonra custom React tabanlı güzel bir arayüz

**📦 Planlama - Modüllerle Git**

| **Modül** | **Açıklama** |
| --- | --- |
| Auth | Kullanıcı login/register/token alma |
| Storage | Dosya yükleme, silme, listeleme |
| Task Engine | Celery ile arkaplan işlemler |
| API | Public REST/GraphQL endpoint’ler |
| UI | Standart kullanıcı arayüzü |
| Dev Portal | Swagger, Token yönetimi, Webhook ayarları |

**🧭 Yol Haritası (Gerçekçi Başlangıç)**

1. Django projesi kur
2. User & Token Auth sistemi yaz (DRF Token Auth)
3. Dosya upload/download fonksiyonu
4. Upload sonrası background işlem (örn: thumbnail)
5. Dosyalar için API + link üretimi
6. Swagger + API Key kontrolü
7. Web UI: dosya yöneticisi (dropzone.js, tailwind)
8. Geliştirici için token, webhook, analytics ekranı

## 🎯 MSC Platformunun Amaçları

| **Hedef** | **Açıklama** |
| --- | --- |
| 🧠 Öğrenme | Gerçek dünyadaki cloud servis mimarilerini öğrenmek |
| 👥 Kullanıcı | Basit dosya yönetimi, hızlı erişim, arayüz |
| 👨‍💻 Geliştirici | API + webhook + access token + event tetikleme gibi şeyler |
|  |  |

## ⚙️ Öğrenilecek Teknolojiler / Enstrümanlar

| **Alan** | **Teknoloji** | **Amaç** |
| --- | --- | --- |
| Backend | **Django + DRF** | CRUD, Auth, API |
| Task Queue | **Celery + Redis** | Arka plan işleri (image işlem, bildirim) |
| File Storage | **MinIO** veya local+Nginx | S3 deneyimi, link oluşturma |
| Database | **PostgreSQL** | DB tasarımı, sorgu performansı |
| DevOps | **Docker + Compose** | Her şey konteynerli, taşınabilir sistem |
| Logging/Monitor | **Grafana, Prometheus, Sentry** | Takip, debug, sağlık kontrolü |
| Messaging | **RabbitMQ** | Event mimarisi, microservice uyumu |
| Frontend (opsiyonel) | **React + Tailwind** | Dashboard, dosya yönetimi arayüzü |
| API Docs | **Swagger / ReDoc** | Geliştirici erişimi için belgeler |
| Auth | **JWT veya OAuth** | Geliştiriciye token ile erişim |

## 🧱 Temel Modüller (MVP Yapı)

| **Modül** | **Açıklama** |
| --- | --- |
| **auth** | Kullanıcı kaydı, login, token üretimi |
| **storage** | Dosya yükleme, erişim, link oluşturma |
| **tasks** | Arka plan işlemleri (thumbnail, hash hesaplama) |
| **developer** | Token, Webhook, API erişimi ekranları |
| **dashboard** | Dosya yönetim UI + analytics |
| **admin** | Kullanıcı, dosya, kuyruk yönetimi |