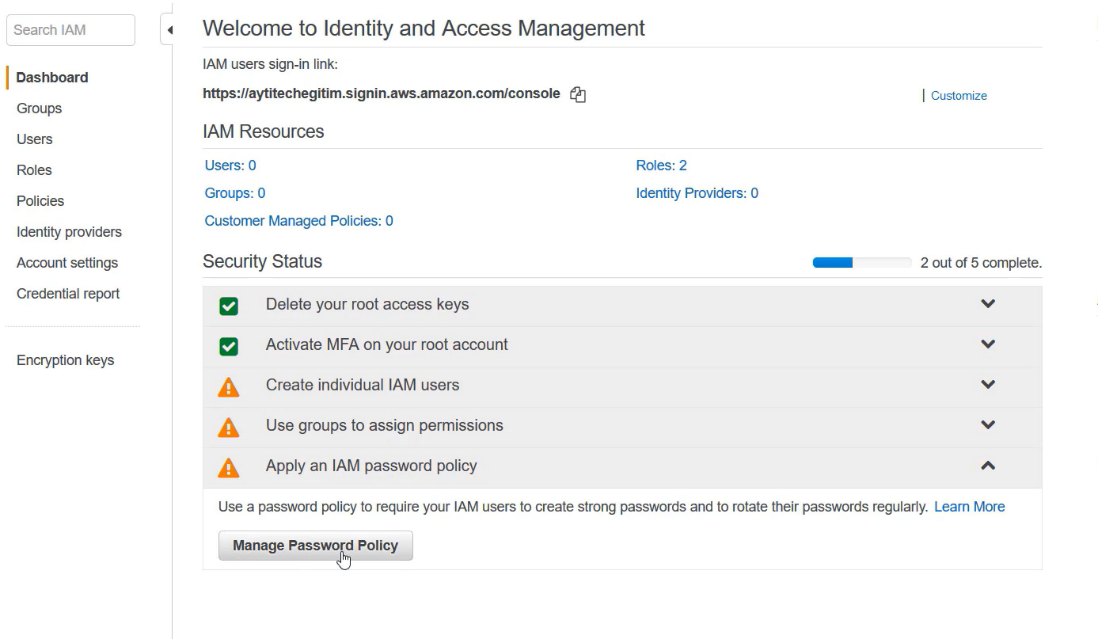
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



Create individual IAM users: gidin ve root kullanmadan user lar oluşturun.

Use groups to assign permission: gruplar oluşturup onlara yetkiler ata

Apply an IAM password policy: password policy oluştur(90 günde bir değiştirsin, son 5 şifreyi kullanamasın, 8 karakterli olsun -🡪 bunlara password policy diyoruz)

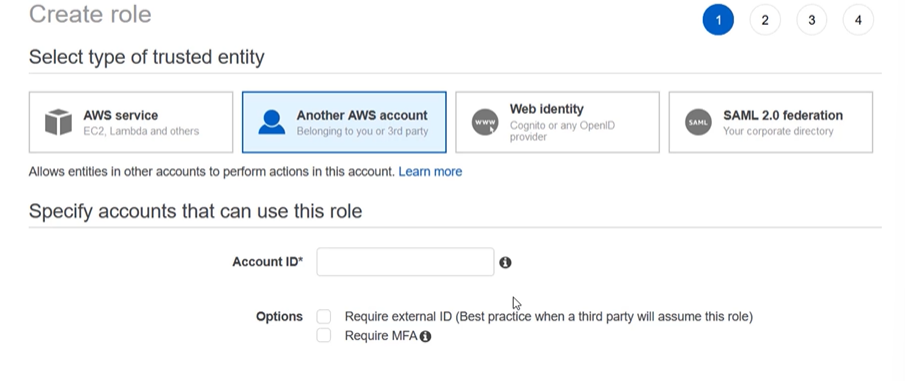
Policy = poliçeler

Roles = roller

S 3 adında bir policy oluştur, sonra yeni bir user oluştur ve bu yeni user a bu policy i ata.

Bu yeni kullanıcı S3 de verilen yetkiler dışında hiçbir şey yapamayacak, policy ler bu şekilde kısıtlı ortamlar ve kısıtlamalar oluşturmamızı sağlayan dosyalardır. Bunu rollere , group lara ve user lara atayabiliriz.

**Roller**; aws kaynaklarının bir başka aws kaynaklarına veya dışardan bir kullanıcının bizim aws kaynaklarımıza erişebilmeleri için oluşturduğumuz yetki seviyeleridir.



Aws hesabıyla ilgili bir rol atayacaksak 🡪 aws kısmından devam edeceğiz

Başka bir kaynağın erişimine yetki vereceksek 🡪 Another Aws account ı seçeceğiz.

Rol; farkı hesapların birbirleri üzerinde yetki sahibi olmalarına izin veren objelerdir. Yönetilmesi istenilen hesap üzerinde yeni bir kullanıcı oluşturmak yerine yönetilmesi istenilen hesap üzerinde yeni bir rol oluşturup bu rolde de izin verip; hesabın yönetilmesini sağlayabiliriz.

Örnek; EC 2 servisimizin başka AWS servislerine ulaşabileceğini seçelim.

EC2 yu seçtik. EC2 kaynaklarımıza rol atamak istediğimi belirttik ardından bu EC2 sanal makinelerin hangi servislere erişeceğini belirleyelim ve S3 Access i seçelim (İşte burada biz S3 Full Access policy seçerek yetkilerini belirlemiş olduk)

Burada rolümüze S3 Full Access poliçesini atamış olduk. Son kısımda ise review da yani role name ve role description ı dolduruyoruz, creat role diyerek oluşturmuş oluyoruz.



**S3 – SİMPLE STORAGE SERVİCE**

En eski servistir, AWS in dosya deposudur. Nesne tabanlıdır.

Basit Depolama servisidir, Amazon S3 verileri kovalar içinde nesneler olarak depolar.



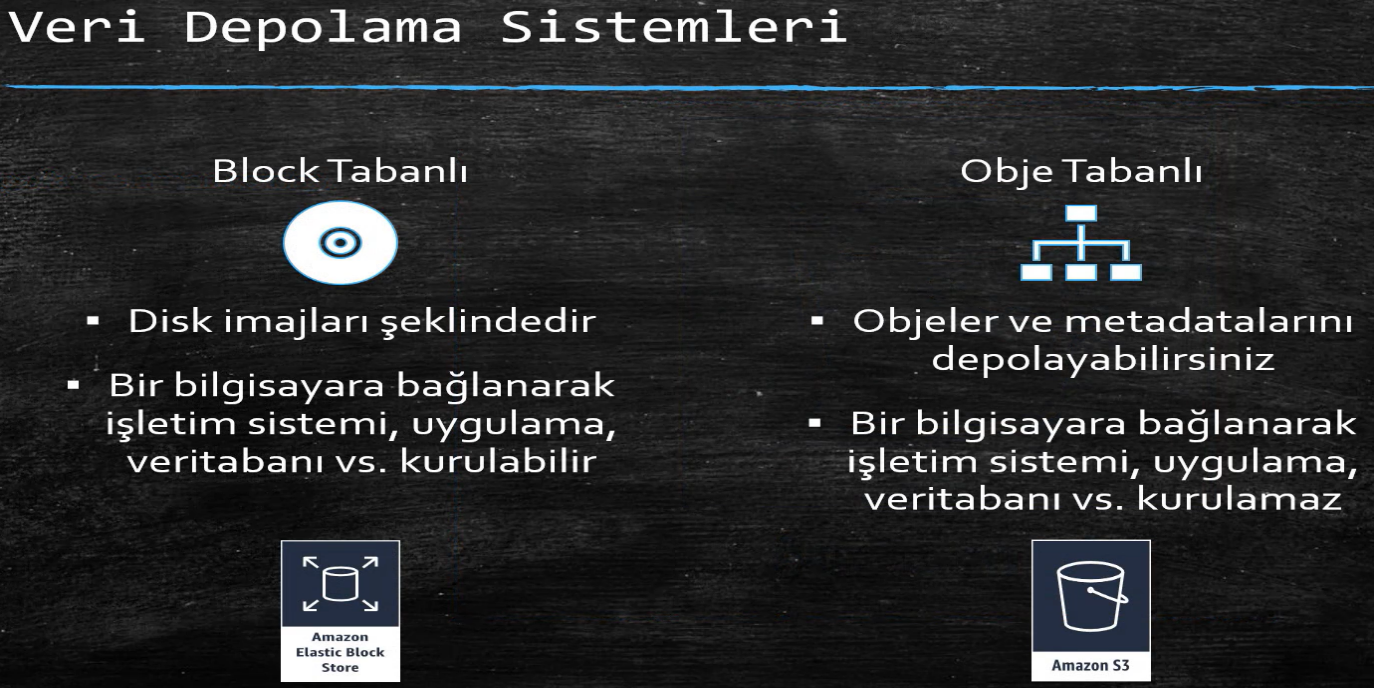
Veri depolama sistemleri 2 tanedir;

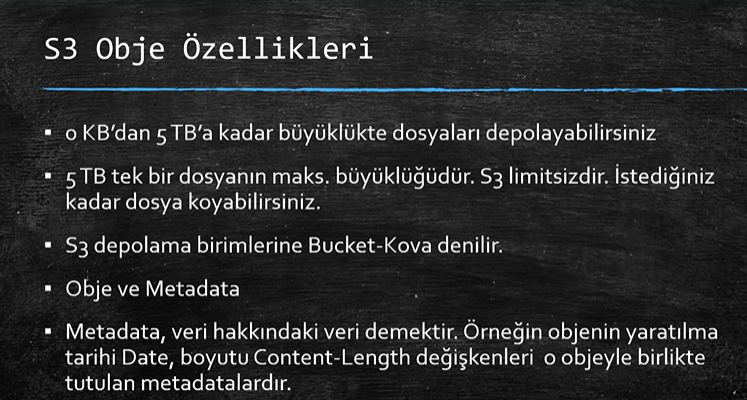
Block tabanlı;

Obje tabanlı(filebase tabanlı); klasör ve dosyalar yani nesneler depolayabildiğimizdir.

Örnek; S3 e bir işletim sistemi KURAMAYIZ.

S3; Onedrive, Google Drive ya da apple I Cloud u gibi dosya deposudur.





Bucket lar S3 oluşturulan ilk şeydir.

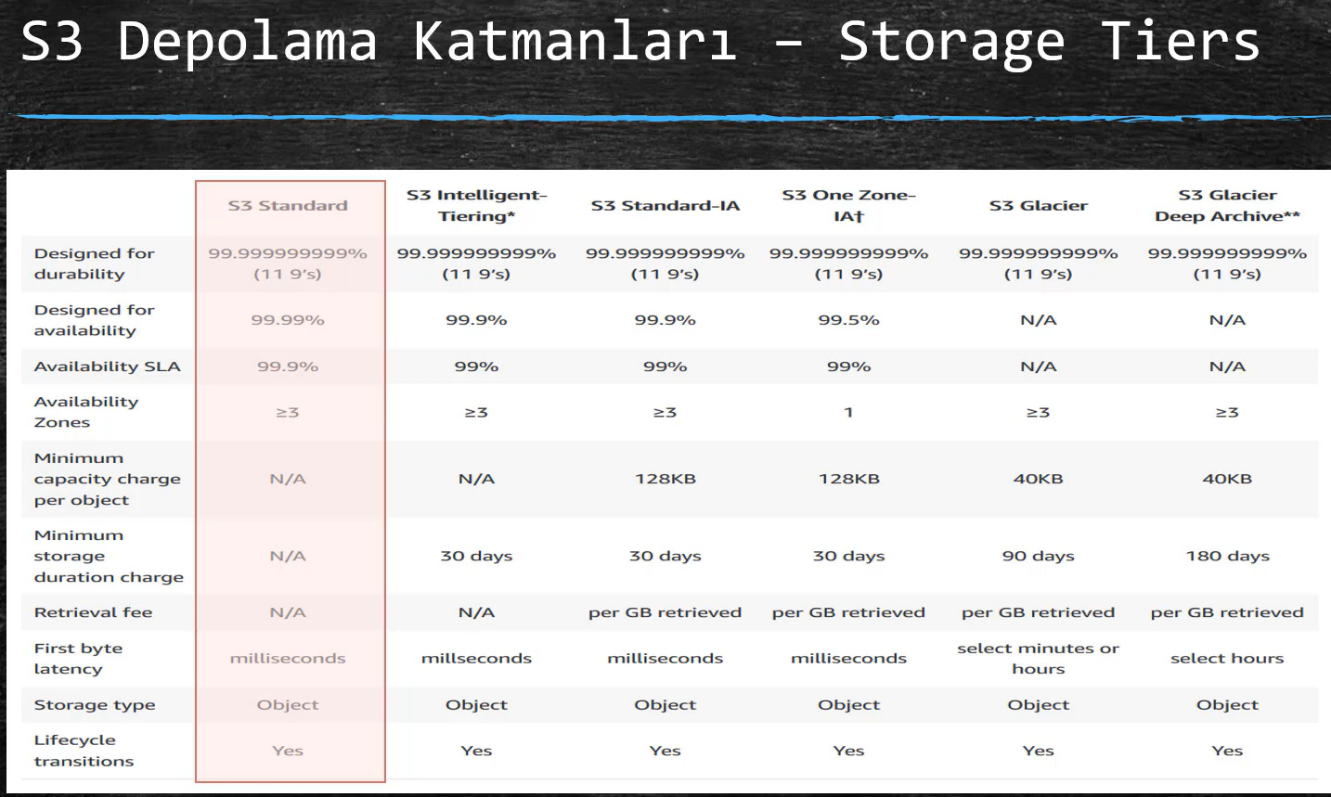
Bucket lar nesneler için container lardır.

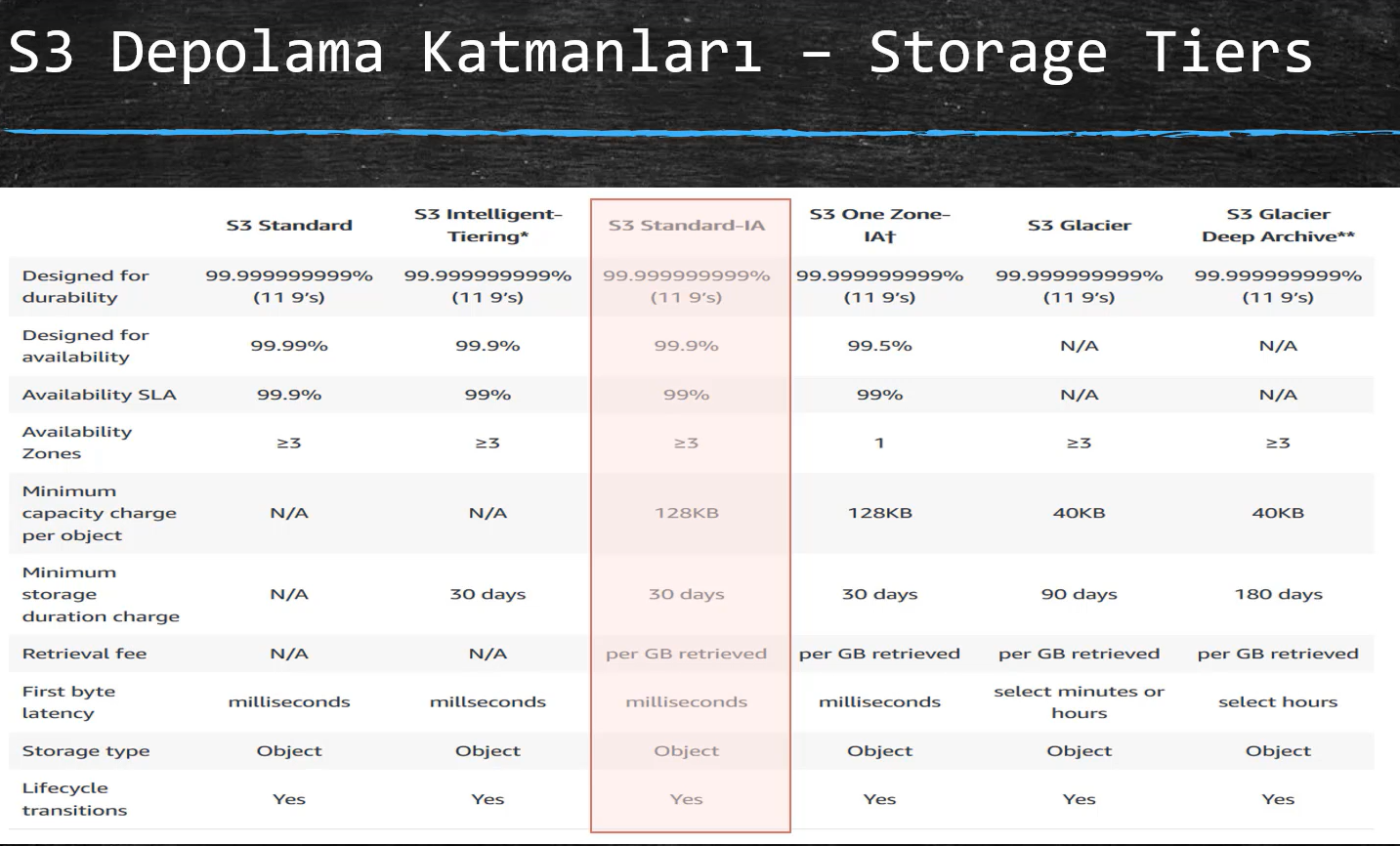
Bucket lar region bazlıdır.

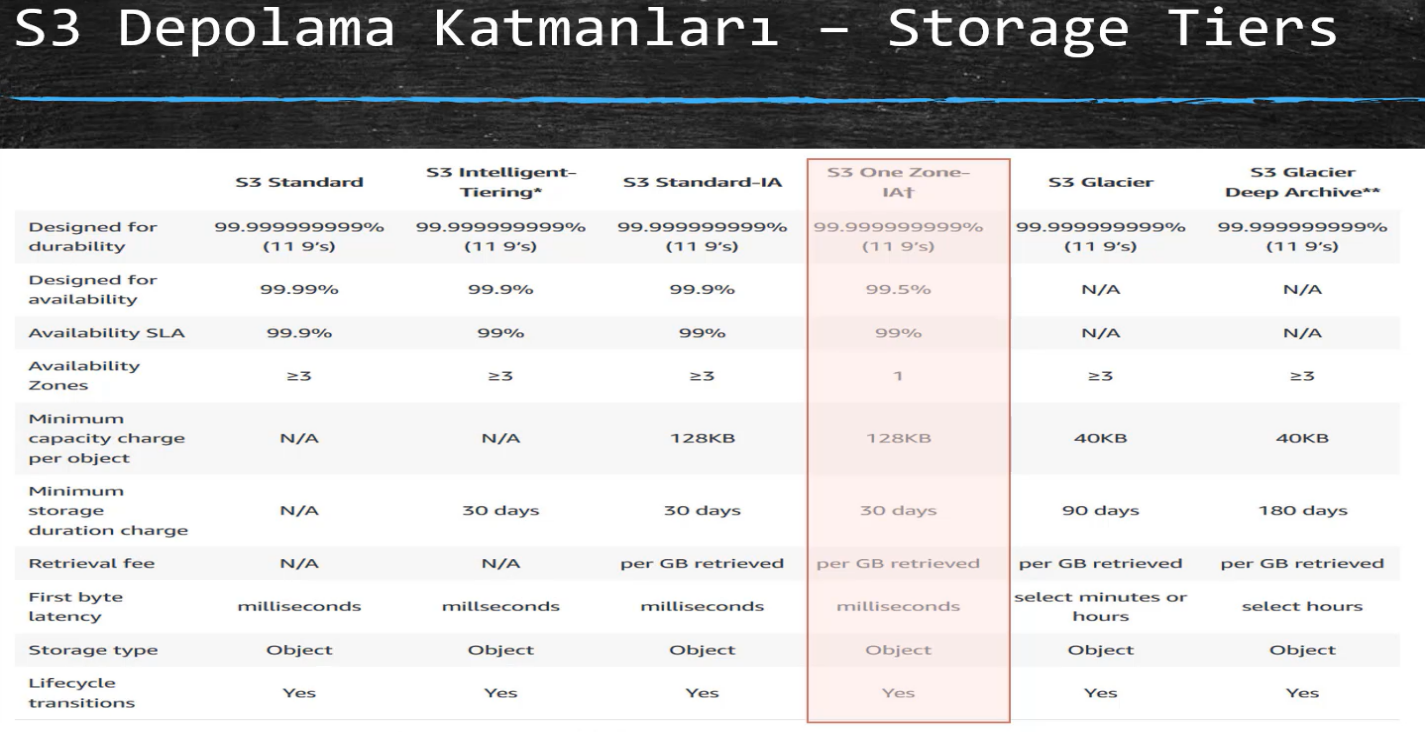
S3 nesne deposunun yanında bu nesne ile ilgili key dediğimiz ve aslında dosyanın ismi olan

Meta data dediğimiz; bu nesnenin bazı özelliklerini belirleyen bilgileri de tutar.

Bucket isimleri unic olmalıdır.

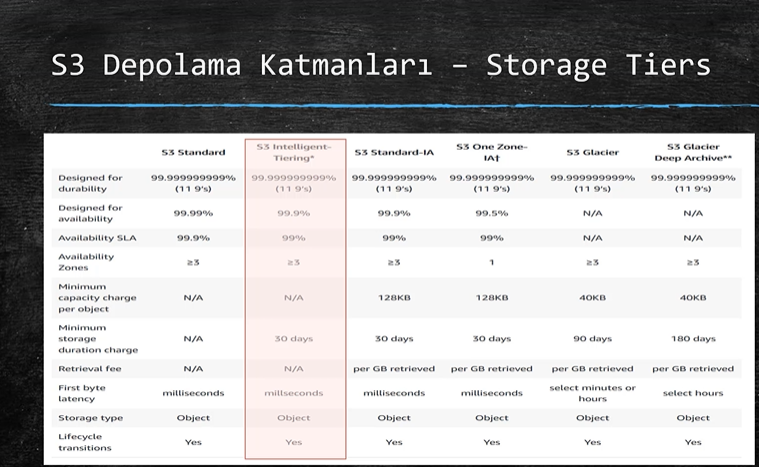






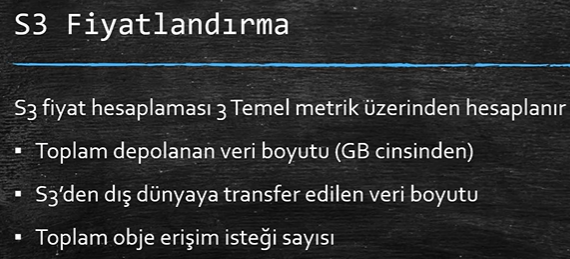
Aws artık bu katman geçişlerini kendisi yapıyor.

Buradaki S3 Glacier servisinde ise çok nadiren erişmek durumunda olduğumuz dosyalarımızı tutabiliriz.

Bu katman ise doğru çağırma katmanı değil ARŞİV olarak kullanılan bir katmandır.

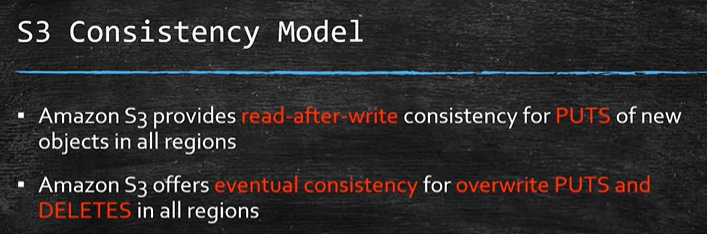
İntelliegent Tiering katmanı; Aws Glazcier katmanının bir kanca atılmış hali gibidir.

Bu katmanlar arası geçişi SDLC ile kuruyoruz ki upload anında katman geçişi yapabiliyoruz.



Katman fiyatları farklıdır; standart en pahalısıdır.

S3 de dosya depolamak maliyetli değil maliyetli olan kalem bu dosyalara erişmektir.



Biz bir dosyayı S3 e attığımız zaman upload tamamlanır, tamamlanmaz hemen erişim sağlanabilir.

Ama eğer daha önce daha önce upload edilen dosyada değişiklik yapar ya da silersek erişme anında biraz gecikme olabilir ki bunlar milisaniyelik gecikmelerdir.

Bu gecikmenin sebebine ise S3 ün bu dosyayı arkadaki birçok cihazda güncellemesinin zaman almasıdır. S3 bir dosya depolama servisi olarak görmemek lazım; S3 bize kendi API ve SDCase ile kendi yazacağımız yazılımların depolama ihtiyacını çözme imkanı sağlıyor. S3 esas gücü dosya depolaması değildir, bunu zaten yıllardır birçok servis ile sağlıyor aws. S3 bu depolama işlemini çok uygun fiyatlarla yazılımımız içine gömme imkanı sağlamasıyla diğerlerinden sıyrıldı. İşte güçlü yanı bu işte bizler Son kullanıcı olarak milisaniye bir farkın anlatıldığı read after write consistency, eventual consistency yani tutarlılık durumuna önem vermezseniz. Ama bir uygulama yazarken bunu göz önünde bulundurmak buna göre kurgu yapmak gerekmektedir.