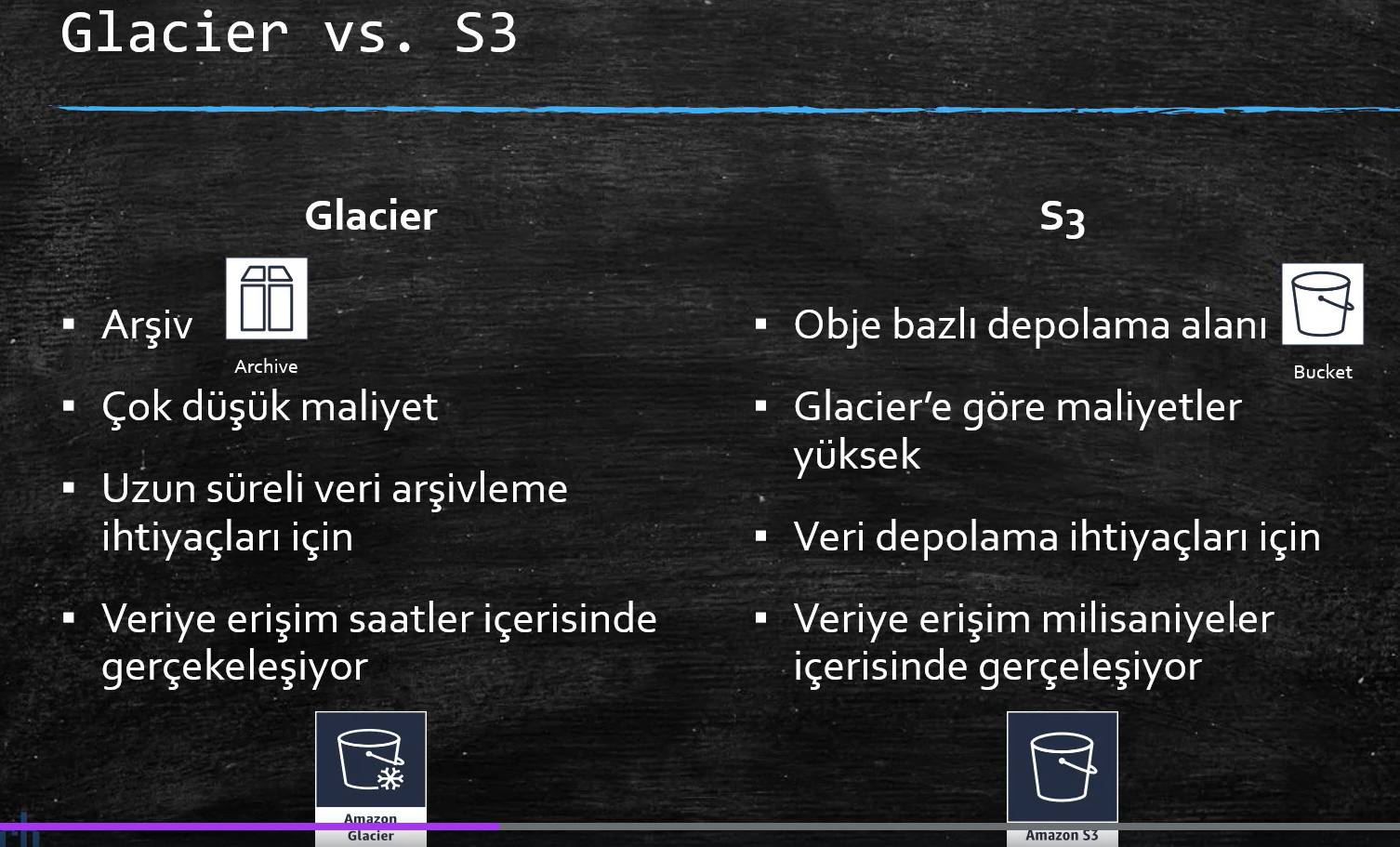
51. VİDEO

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Uzun bir dönem saklamamız gereken ve nadiren eriştiğimiz dosyalar için güvenli bir arşiv arıyorsak bu Glacier dir. Bir muhasebe şirketinin 10 yıl saklamak zorunda old. Evraklar gibi.

Glacier; veri arşivleme ve yedekleme işini yapar.



S3 gibi veri depolama amaçlı değil ARŞİV leme amaçlıdır. Arşiv deki yani Glacier deki bir dosyaya ya direkt erişemeyiz. Bunu isteyen bir görev tanımlanır dosyanın kopyası S3 servisine yüklenir.

Glacier 2 kompenent den oluşur; Mahzen, Arşiv

Glacier ya da S3 Glacier service in adıdır. Bu servisde önce mahzen(vaolt) oluşturulur. Birden fazla vault oluşturabiliriz. Bu Vault içinde de arşiv (archive) ler yapabiliriz. Arşivler S3 deki bucket ler gibidir.

Başka önemli 2 husus;

Glacier yönetim konsolundan sadece Mahzen açmaya izin verir.

Bunun dışında kalan arşiv oluşturma buraya dosya yükleme, çekme vb. işlemlerin tamamını komut satırından gerçekleştirir. Ya da aws SD KEY ler kullanarak kendi yazdığımız yazılıma entegre edebiliriz.

S3 de kullanıldığımız Glacier Storage Tier ın bu Glacier ekranından yönetilmediğini bilmemiz gerekiyor.

S3 bir dosyayı lifesycle management GLACİER katmanına aldığımızda o dosya GLACİER de depolanır ama biz bunu sadece S3 ekranından görür ve yönetiriz. Glacier servisi üzerinde bu dosyalara erişemeyiz.

52. VİDEO (GLACİER UYGULAMA)

Sadece bir tane vault oluşturduk. (create kısmından)

Glacier uygulama adımları;

https://docs.aws.amazon.com/cli/latest/userguide/cli-services-glacier.html

0: komut satırından egitimvault diye bir vault yarat

aws glacier create-vault --account-id - --vault-name egitimvault

1: 3 mb largefile isimli bir dosya yarat

dd if=/dev/urandom of=largefile bs=3145728 count=1

2: Dosyayı split ile böl

split --bytes=1048576 --verbose largefile chunk

3:Multipart upload için bir alan açıyoruz

aws glacier initiate-multipart-upload --account-id - --archive-description "multipart upload test" --part-size 1048576 --vault-name egitimvault

komutu sonucu

{

"location": "/097479760550/vaults/egitimvault/multipart-uploads/ssW0KxfMgdJ4LTRposuVTpPgrGsTAPze\_GQUwtFWrJwIKEkNOdX\_F0z-yNAA\_pQIptvpZueGWWUszGZ446HBIE9aWWVr",

"uploadId": "ssW0KxfMgdJ4LTRposuVTpPgrGsTAPze\_GQUwtFWrJwIKEkNOdX\_F0z-yNAA\_pQIptvpZueGWWUszGZ446HBIE9aWWVr"

}

4: Sonra bunu bir değişkene atıyoruz ve multipart upload'a başlıyoruz

$UPLOADID="ssW0KxfMgdJ4LTRposuVTpPgrGsTAPze\_GQUwtFWrJwIKEkNOdX\_F0z-yNAA\_pQIptvpZueGWWUszGZ446HBIE9aWWVr"

aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkaa --range 'bytes 0-1048575/\*' --account-id - --vault-name egitimvault

aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkab --range 'bytes 1048576-2097151/\*' --account-id - --vault-name egitimvault

aws glacier upload-multipart-part --upload-id $UPLOADID --body chunkac --range 'bytes 2097152-3145727/\*' --account-id - --vault-name egitimvault

5: Openssl ile dosyanın hashını cıkarıyoruz ki buradaki dosya ile giden dosya aynı mı kontrol edebilelim

$ openssl dgst -sha256 -binary chunkaa > hash1

$ openssl dgst -sha256 -binary chunkab > hash2

$ openssl dgst -sha256 -binary chunkac > hash3

6: ilk 2 hashi yeni bir dosya yapıyoruz sonra onun da da hashini alıyoruz

$ cat hash1 hash2 > hash12

$ openssl dgst -sha256 -binary hash12 > hash12hash

7: sonra bunu ucuncu dosyayla birlestiriyoruz ve en sonunda o hash degerini degiskene atiyoruz

$ cat hash12hash hash3 > hash123

$ openssl dgst -sha256 hash123

SHA256(hash123)= 9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67

$ TREEHASH=9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67

8: son olarak da uploadu tamamlıyoruz

aws glacier complete-multipart-upload --checksum $TREEHASH --archive-size 3145728 --upload-id $UPLOADID --account-id - --vault-name egitimvault