Adattárolás a felhőben

Hogyan csatolunk tárhelyet a hálózaton — a hagyományos NFS/SAMBA/NAS világtól az AWS S3 tárolási osztályaiig.

Honnan indulunk — és hova jutunk el

1. blokk — Hálózati tárolás csatolása: NFS, SAMBA, iSCSI, ipari NAS/SAN

2. blokk — Ugyanez a felhőben: objektum, blokk és fájl tárolás (S3, EBS, EFS)

3–4. blokk — AWS tárolási osztályok (S3 Standard-tól a Glacier-ig) és a konzisztencia-modellek

5–6. blokk — A fájlrendszerek a háttérben, és egy kitekintő (LVM, RAID, OverlayFS)

A 3. órán megépítettük az adatközpont belső hálózatát (Clos/Spine–Leaf), a 4. órán megnéztük, hogyan aktiválunk rajta virtuális erőforrásokat. Most: hogyan jut el ezekhez a gépekhez a **tárhely** ugyanezen a hálózaton.

1. BLOKK

Hálózati tárolás csatolása

Mielőtt a felhőszolgáltatókhoz érnénk: hogyan oldja ezt meg egy hagyományos adatközpont, vállalati szerverterem?

Miért nem elég a helyi lemez?

Egy szerver saját, belső lemeze gyorsnak tűnik — de ha a gép meghibásodik, az adat vele vész el, és ha több szerver dolgozik ugyanazon az adaton, mindegyiknek külön másolatot kéne tartania.

A megoldás: a tárolást **leválasztjuk** a számítási kapacitástól, és a 3. órán megépített hálózaton keresztül tesszük elérhetővé — akár egyszerre több szerver számára is.

Két fő irány van: **fájl-szintű** megosztás (NFS, SAMBA — a kliens egy már kész fájlrendszert lát), és **blokk-szintű** megosztás (iSCSI — a kliens egy nyers lemezt lát, amire saját fájlrendszert tesz).

NFS (Network File System)

A Sun Microsystems 1984-es, azóta szabvánnyá vált protokollja — a UNIX/Linux világ natív hálózati fájlmegosztása.



NFSv3: egyszerű, állapotmentes. **NFSv4**: állapottartó, beépített zárolás és Kerberos-alapú hitelesítés — *ugyanerre a protokollra épül majd a később tárgyalt AWS EFS is.*

SAMBA / CIFS (SMB)

Az **SMB** (Server Message Block) a Microsoft Windows natív fájlmegosztási protokollja; a **Samba** ennek nyílt forráskódú, Linuxon futó szerver-implementációja.



Lehetővé teszi, hogy egy Linux szerver natívan szolgáljon ki Windows klienseket — jellemzően **Active Directory** tartományi hitelesítéssel integrálva vegyes (Windows + Linux) vállalati környezetekben.

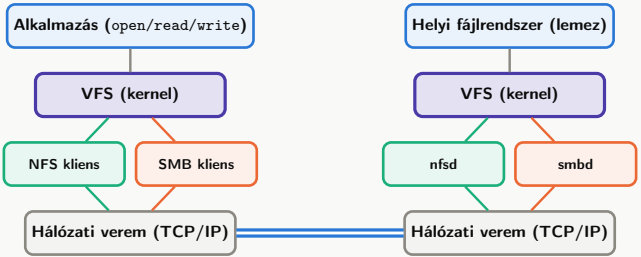
NFS vagy SAMBA — melyiket?

	NFS	SAMBA / SMB
Eredet	UNIX/Linux	Windows
Natív kliens	Linux, macOS	Windows
Hitelesítés	UID/GID, Kerberos (v4)	Felhasználónév/jelszó, AD
Tipikus használat	Linux szerverfarmok, EFS	Vegyes irodai/vállalati megosztás

A gyakorlatban gyakran **mindkettő fut egyszerre** ugyanazon a dobozon — ezt mutatja majd az ipari NAS-okról szóló dia, három dia múlva.

NFS és SAMBA a VFS modellben

Mindkét protokoll ugyanabba a kernel-rétegbe, a **VFS**-be (Virtual File System) csatlakozik be — az alkalmazás ugyanazokkal a rendszerhívásokkal dolgozik, akár helyi, akár hálózati a tárhely.



A kliens oldalon a mountolt protokoll dönti el, hogy a VFS az NFS vagy az SMB klienst hívja; a szerver oldalon az `nfsd`/`smbd` daemon maga is a **saját, helyi VFS-én** keresztül éri el a mögöttes fájlrendszert (ext4/XFS/...).

iSCSI — blokk-szintű hálózati tárolás

Az NFS/SAMBA **fájlokat** ad a kliensnek; az **iSCSI** ezzel szemben egy nyers **lemezt** (LUN-t) — a kliens ráteszi a saját fájlrendszerét, mintha helyi lemez lenne.

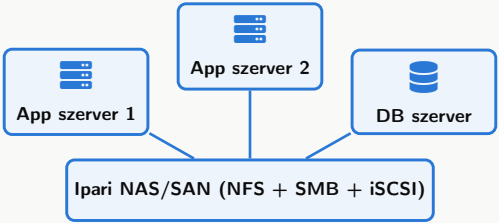


sima Ethernet / TCP-IP felett

Olcsóbb alternatíva a dedikált Fibre Channel SAN-hálózathoz képest, mert a már meglévő Ethernet-infrastruktúrán fut. **Ugyanezt a koncepciót** (hálózaton elérhető blokk-eszköz) valósítja meg felhő-natív, menedzselt formában — erről bővebben az AWS EBS-nél, néhány dia múlva.

Ipari NAS / SAN megoldások

Ezeket a protokollokat a gyakorlatban dedikált, sokszor **redundáns** tárolóeszközök szolgálják ki — egyetlen dobozból, egyszerre.



Kisebb léptékben: Synology, QNAP. Enterprise szinten: NetApp, Dell EMC, Pure Storage, HPE — jellemzőik: **redundáns vezérlők/tápegységek**, több protokoll egyszerre, beépített snapshot és replikáció, SSD/HDD közötti automatikus tiering.

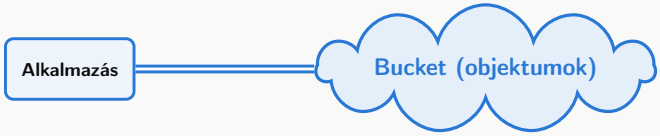
Ugyanez a felhőben — csak menedzselve

A felhőszolgáltatók pontosan ezeket a fogalmakat (fájl-, blokk- és egy teljesen új: **objektum**-szintű tárolás) kínálják **szolgáltatásként** — nem kell saját NAS-t vagy SAN-t üzemeltetni hozzá.

Ahogy a hypervisort (4. óra) a privát felhő platformok tették API-n elérhetővé, úgy teszi ugyanezt a tárolással az AWS S3, EBS és EFS.

Objektum tárolás (pl. AWS S3)

Fájlokat (**objektumokat**) tárol laposan, mappastruktúra nélkül — a „mappákat” csak a fájlnevek elé fűzött prefixek szimulálják.



Nincs klasszikus helyi megfelelője — nem csatolható fájlrendszerként. Nagy mennyiségű, ritkán módosuló adathoz (képek, videók, biztonsági mentések) ideális.

Blokk tárolás (pl. AWS EBS)

Egy nyers, particionálható, fájlrendszerrel ellátható **virtuális merevlemezt** ad, amit egyetlen szerverhez csatolunk.

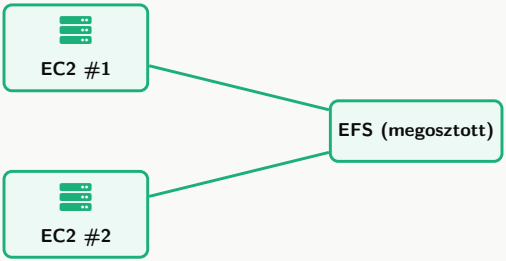


alacsony késleltetés, 1:1 kapcsolat

Az imént megnézett **iSCSI gondolat menedzselt, felhő-natív változata** — adatbázisok, operációs rendszer image-ek jellemző felhasználási területe, pillanatképekkel (snapshot).

Fájl tárolás (pl. AWS EFS)

NFSv4 alapokon — amit **egyszerre több szerver is** csatolhat és közösen írhat/olvashat.



Nem véletlen az NFS-dia: az **EFS egy teljesen menedzselt NFS szerver** — ugyanaz a protokoll, csak nem neked kell üzemeltetni a gépet mögötte.

A három modell összefoglalva

Szempont	Objektum (S3)	Blokk (EBS)	Fájl (EFS)
Elérés	HTTP(S) API	Blokk-szintű (iSCSI-szerű)	Hálózati FS (NFSv4)
Csatolható OS-be?	Nem	Igen	Igen
Egyszerre hány gép?	Korlátlan	Jellemzően 1	Sok
Jellemző adat	Képek, mentések	DB, OS-image	Megosztott konfig

3. BLOKK

AWS tárolási osztályok

Nem minden S3-ba kerülő adatot érünk el ugyanolyan gyakran — ennek megfelelően az ára sem egyforma.

„Meleg” és „hideg” adat

Egy frissen feltöltött profilkép percenként sok ezerszer kérhető le — egy 7 éve archivált számla talán soha többé.

⚠ Ha mindkettőt ugyanazon az áron, ugyanolyan gyors eléréssel tárolnánk, az archív adat feleslegesen drága lenne. Az AWS S3 ezért **több tárolási osztályt** kínál — ugyanaz az API, más ár/hozzáférési-sebesség arány.

A „meleg” oldal: S3 Standard család

S3 Standard

Azonnali hozzáférés, legmagasabb tárolási díj, nincs minimum tárolási idő — aktívan használt adatokhoz.

S3 Intelligent-Tiering

Automatikusan mozgatja az objektumokat a hozzáférési minta alapján — kis megfigyelési díjért cserébe nincs lekérési díj.

S3 Standard-IA

Olcsóbb tárolás, de lekéréskor felár; azonnali hozzáférés; min. 30 nap tárolás.

S3 One Zone-IA

Mint a Standard-IA, de csak **1 rendelkezésreállási zónában** — olcsóbb, de kevésbé ellenálló.

A „hideg” oldal: S3 Glacier család

Glacier Instant Retrieval

Archívum, de **milliszekundumos** hozzáférés — min. 90 nap tárolás.

Glacier Flexible Retrieval

Lekérés percek–órák alatt (sürgősség szerint árazva) — min. 90 nap tárolás.

Glacier Deep Archive

A legolcsóbb osztály; lekérés 12–48 óra — min. 180 nap tárolás.

Jellemző felhasználás: jogi/compliance megőrzés (pl. 7–10 éves adatmegőrzési kötelezettség), katasztrófa utáni helyreállítási másolatok.

A hét osztály egy táblázatban

Osztály	Hozzáférés	Min. tárolás	Jellemző felhasználás
Standard	azonnali	—	aktívan használt adat
Intelligent-Tiering	azonnali	—	ismeretlen/változó hozzáférési minta
Standard-IA	azonnali	30 nap	ritkán elért, de gyorsan kellhet
One Zone-IA	azonnali	30 nap	újra-előállítható adat
Glacier Instant Retrieval	azonnali	90 nap	archívum, azonnali eléréssel
Glacier Flexible Retrieval	percek–órák	90 nap	ritka archívum, nem sürgős
Glacier Deep Archive	12–48 óra	180 nap	hosszú távú, jogi megőrzés

⚠ A „min. tárolás” **nem technikai korlát, hanem díjszabási garancia**: ha egy objektumot ennél korábban törölsz vagy másik osztályba mozgatsz, az AWS akkor is a teljes minimum időre számláz — ez tartja vissza attól, hogy az olcsó, archív osztályokat aktívan, gyakran változó adatra használd.

Lifecycle policy — automatikus átmenet

Nem kell kézzel eldönteni, melyik objektum melyik osztályba való — egy **lifecycle szabály** automatikusan átmozgatja az adatot, ahogy öregszik.

Példa: egy naplófájl 0–30 napig **S3 Standard**-ban van, 30–90 nap között átkerül **Standard-IA**-ba, 90 nap után pedig **Glacier Flexible Retrieval**-be — mindezt egyetlen, egyszer beállított szabály intézi.

Erős konzisztencia (strong consistency)

Egy sikeres írás után **minden következő olvasás** garantáltan a legfrissebb adatot adja vissza — függetlenül attól, melyik replikát/zónát érte el a kérés.

Ez az, amire intuitívan számítunk — de elosztott, több adatközpontra replikált rendszerben ennek biztosítása valódi mérnöki erőfeszítés és néha teljesítmény-kompromisszum.

Végső konzisztencia (eventual consistency)

Egy írás után **rövid ideig** előfordulhat, hogy egy másik replikáról olvasva még a **régi** adatot kapjuk vissza — de egy idő után (jellemzően másodperceken belül) minden replika konzisztenssé válik.

⚠ Cserébe nagyobb rendelkezésreállást és jobb teljesítményt enged — ez a klasszikus **kompromisszum** elosztott rendszerekben (konzisztencia vs. rendelkezésreállás/sebesség).

Egy valós példa: az AWS S3 története

Az AWS S3 **sokáig csak végső konzisztenciát** biztosított egyes műveleteknél (pl. egy új objektum listázása rövid ideig elmaradhatott).

2020 vége óta viszont minden művelethez (GET, PUT, LIST, DELETE) **erős**, olvasás-írás utáni konzisztenciát garantál — külön felár nélkül.

Jó példa arra, hogy egy ilyen garancia nem statikus — érdemes mindig az aktuális dokumentációt ellenőrizni, ne a pár évvel ezelőtti tudásra hagyatkozni.

5. BLOKK

Linux fájlrendszerek a tárolás mögött

Egy blokk tároló (akár EBS, akár egy NAS LUN-ja) önmagában csak nyers tárterület — fájlrendszer kell rá.

ext4

A legtöbb Linux disztribúció (pl. Ubuntu) **alapértelmezett** fájlrendszere.

Naplózó (journaling) fájlrendszer: a metaadat-módosításokat előbb egy naplóba írja, csak utána a végleges helyükre — így egy váratlan áramkimaradás esetén is gyorsan, konzisztens állapotba állítható vissza.

Kiforrott, jól ismert, kompatibilis — jó alapértelmezett választás, ha nincs speciális igényünk.

XFS

Eredetileg az SGI IRIX rendszeréhez készült, ma a **Red Hat Enterprise Linux** (és AlmaLinux, Rocky Linux) alapértelmezettje.

Kifejezetten **nagy fájlokra és nagy párhuzamos I/O-terhelésre** optimalizált (adatbázisok, nagy médiafájlok), jól skálázódik több CPU maggal is.

Natívan **nem támogat pillanatképet** (ehhez alsóbb szintű megoldás, pl. LVM snapshot kell), és a mérete utólag csak **növelhető**, nem csökkenthető.

BTRFS

Modern, **copy-on-write (CoW)** elvű fájlrendszer (az openSUSE alapértelmezettje, sok NAS gyártó is használja).

A CoW elv miatt egy módosításkor sosem írja felül a régi adatot a helyén — ennek köszönhetően a **pillanatképek** (snapshot) készítése szinte azonnali, nulla extra hellyel. Beépített adatintegritás-ellenőrzés, tömörítés, több lemez összekapcsolása (natív RAID).

A BTRFS RAID5/RAID6 módja évek óta ismert, nem javított „write hole” hibával küzd — éles környezetben nem ajánlott.

ZFS

Eredetileg a Sun Solaris rendszeréhez készült, ma nyílt forráskódú folytatása az **OpenZFS**.

Egyszerre fájlrendszer **és** kötetkezelő: CoW elv (mint a BTRFS), beépített checksumming minden adaton, saját **RAIDZ** (nem szenved a write hole jelenségtől), tömörítés, és az ARC nevű RAM-alapú gyorsítótár.

Magasabb erőforrásigény (elsősorban RAM) — cserébe kiemelkedően megbízható, elsősorban dedikált tárolószervereken, NAS-okon terjedt el — gyakran **ugyanazokon az ipari NAS dobozokon, amiket az 1. blokkban láttunk**.

ext4 / XFS / BTRFS / ZFS összefoglalva

	CoW	Snapshot	Natív RAID	Jellemző
ext4	Nem	Nem	Nem	Általános célú alapértelmezett
XFS	Nem	Nem	Nem	Nagy fájlok, párhuzamos I/O
BTRFS	Igen	Igen	Igen*	Desktop/NAS, gyors snapshot
ZFS	Igen	Igen	Igen	Dedikált tárolószerverek, NAS

*BTRFS RAID5/6 — write hole hiba miatt éles környezetben kerülendő

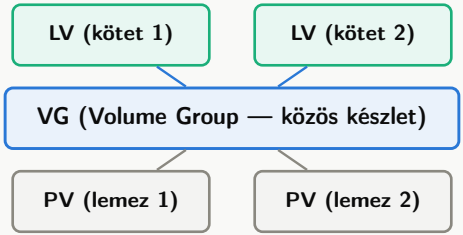
6. BLOKK

Kitekintő: érdemes még megnézni

Nem kifejezetten felhő-specifikus, de Linux-üzemeltetés során szinte biztosan találkozol velük.

LVM (Logical Volume Manager)

Egy réteg a **fizikai lemez(ek)** és a **fájlrendszer** között — innen a három alapfogalom.



A logikai kötet (LV) — amire a fájlrendszert tesszük — **online átméretezhető**, és **több fizikai lemezt is összefoghat**. Saját snapshot-mechanizmusa pótolja pl. az XFS hiányzó natív snapshotját.

Linux szoftveres RAID (mdadm)

A **hagyományos**, fájlrendszer-független módja, hogy több fizikai lemezből redundanciát vagy teljesítményt nyerjünk — a kernel blokk-eszköz szintjén (`md` eszközök), az `mdadm` paranccsal kezelve.

Támogatja a megszokott szinteket: RAID0 (csíkozás), RAID1 (tükrözés), RAID5/6 (paritásos redundancia), RAID10.

A korábban látott táblázat „natív RAID” oszlopa erre a rétegre utalt, ha a fájlrendszer (ext4, XFS) maga nem tud RAID-et — a BTRFS/ZFS viszont ezt **saját maga**, beépítve oldja meg, elkerülve az mdadm RAID5/6 write hole kockázatát is.

UnionFS — a koncepció

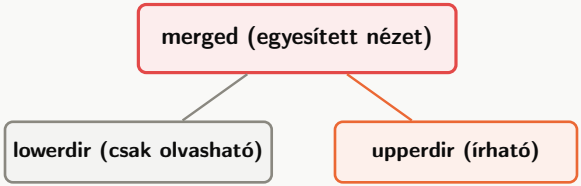
Az **union mount** ötlete: több, külön könyvtárat/fájlrendszert egyetlen, egyesített nézetként mutatunk meg.

Az **UnionFS** volt az egyik legkorábbi Linux-megvalósítása ennek az elvnek — azóta gyakran magát az *elvet* is „unionfs-szerű” néven emlegetik, függetlenül a konkrét implementációtól.

A gyakorlatban az írások egy kijelölt, felső rétegbe kerülnek, az alsó réteg(ek) változatlanok maradnak — ez az alapja a következő diának.

OverlayFS — és a Docker kapcsolat

A modern, **kernelbe épített** (Linux 3.18 óta) megvalósítása ugyanennek az elvnek — sokkal egyszerűbb/gyorsabb, mint a régebbi UnionFS/AUFS.



Ez a Docker image-rétegezésének alapja: minden Dockerfile-utasítás egy új, csak olvasható réteg (lowerdir), a futó konténer saját, írható rétege (upperdir) pedig csak a *változásokat* tárolja — erre a félév második felében, a Docker-blokkban még visszatérünk.

ÖSSZEFOGLALÁS

Mit vittünk haza?

- Hálózati tárolás alapjai: **NFS** (fájl), **SAMBA** (Windows-kompatibilis fájl), **iSCSI** (blokk) — ipari NAS/SAN-okon gyakran mindhárom egyszerre
- A felhő ugyanezt menedzselt szolgáltatásként adja: **S3** (objektum), **EBS** (blokk), **EFS** (menedzselt NFS)
- Az S3-nak **több tárolási osztálya** van (Standard — Glacier Deep Archive) — a hozzáférési gyakoriság dönti el, melyik éri meg
- A **konzisztencia-modell** (erős vs. végső) nem apróság — és nem is statikus (lásd az S3 2020-as váltását)
- A blokk tároló mögött mindig egy **fájlrendszer** (ext4/XFS/BTRFS/ZFS) dolgozik; üzemeltetésnél ott van még az **LVM**, a **RAID (mdadm)**, és az **OverlayFS** — ami a Docker alapja is egyben

KÖVETKEZŐ ÓRA: FELHŐBIZTONSÁG

Kérdések?

A hagyományos NFS/SAMBA/NAS világtól az AWS tárolási modelljein és osztályain át a Linux fájlrendszerekig — és egy pillantás az OverlayFS-re, ami a Docker alapja.