NJE GAMF · FELHŐALAPÚ SZOLGÁLTATÁSOK · 2. ÓRA

# Felhő ökoszisztéma

Az AWS globális infrastruktúrája: régiók, rendelkezésre állási zónák, és hogy mit kapunk automatikusan az architektúrától — anélkül, hogy kérnénk.

ISMÉTLÉS & MAI TERV

# Honnan indultunk — és hova tartunk ma

### 📌 Múlt óra röviden

- Bérelt tárhelytől a VPS-en át a felhőig
- AWS regisztráció, free tier, biztonságos bankkártya-választás
- A felhő definíciója: megosztott, elasztikus erőforrások

### 🎯 Mai óra

- Szolgáltatási modellek (IaaS/PaaS/SaaS) és felelősségmegosztás
- Az AWS globális infrastruktúrája: régiók és zónák
- **Mi replikálódik automatikusan — és mi nem**
- Hogyan tervezzünk meghibásodásra ennek ismeretében

**Miért fontos ez?**

A projektfeladatban ASG-t, LB-t, RDS-t, VPC-t fogtok használni. Ezek mind erre az architektúrára épülnek — ha értitek a régió/zóna felépítést, hirtelen értelmet nyer, miért kell legalább két zónát megadni egy Load Balancernek, és miért nem elég egyetlen EC2 példány egy éles rendszerhez.

ALAPFOGALMAK

# Szolgáltatási modellek: ki csinálja meg helyettünk?

A kérdés mindig ugyanaz: minél feljebb megyünk a listán, annál kevesebbet kell magunknak üzemeltetnünk — cserébe annál kevesebb kontrollunk marad.

IaaS

### Infrastructure as a Service

Nyers erőforrás: szerver, hálózat, tárhely. Az OS-től felfelé minden a miénk. **Példa: EC2.**

PaaS

### Platform as a Service

Az OS-t, futtatókörnyezetet a szolgáltató adja. Mi csak a kódra és a konfigurációra fókuszálunk. **Példa: Elastic Beanstalk.**

SaaS

### Software as a Service

Kész alkalmazást használunk, a mögötte lévő infrastruktúráról semmit nem tudunk és nem is kell tudnunk. **Példa: Gmail.**

### Hol vagyunk mi ezen a félévben?

A gyakorlatokon **IaaS** szinten dolgozunk (EC2-t magunk konfiguráljuk), de közben olyan **menedzselt szolgáltatásokat** is bevetünk (RDS, S3), amik PaaS-jellegű kényelmet adnak egy-egy komponensre — ez a valóságban is tipikus keverék.

ALAPFOGALMAK

# Megosztott felelősség — ki véd meg mit?

A leggyakoribb felhős biztonsági incidens nem az AWS hibája — hanem a miénk, mert nem tudtuk, hogy egy adott réteg a mi felelősségünk volt.

| RÉTEG | SAAS | PAAS | IAAS |
|---|---|---|---|
| Felhasználói hozzáférés, adatok | mindig az ügyfél felelőssége | | |
| Alkalmazás | **AWS** | ügyfél | ügyfél |
| Operációs rendszer | **AWS** | **AWS** | ügyfél |
| Hálózati konfiguráció | **AWS** | **AWS** | ügyfél |
| Hypervisor, fizikai infrastruktúra | **AWS** | **AWS** | **AWS** |

**A mai óra pontosan erről a legalsó rétegről szól**

Mielőtt eldöntjük, mi a mi felelősségünk (pl. egy alkalmazás több zónában futtatása), értenünk kell, mit épít fel helyettünk az AWS a fizikai infrastruktúra szintjén.

ALAPFOGALMAK

# Privát, publikus, hibrid felhő

### 🔒 Privát felhő

Saját (vagy bérelt, de elkülönített) infrastruktúra. Teljes kontroll, de a teljes üzemeltetési terhet is mi visszük.

### 🌐 Publikus felhő

Megosztott infrastruktúra egy szolgáltatónál (AWS, Azure, GCP). Nincs üzemeltetési többletköltség, rugalmas árazás. **Ezzel dolgozunk a félévben.**

### 🔗 Hibrid felhő

A kettő kombinációja: érzékeny adat marad helyben, a többi kimehet a publikus felhőbe. Gyakori nagyvállalati megoldás.

A mai óra hátralévő részében kizárólag a **publikus felhővel** foglalkozunk — pontosabban azzal, hogy egy ekkora publikus felhő, mint az AWS, fizikailag hogyan épül fel.

MÉLYFÚRÁS

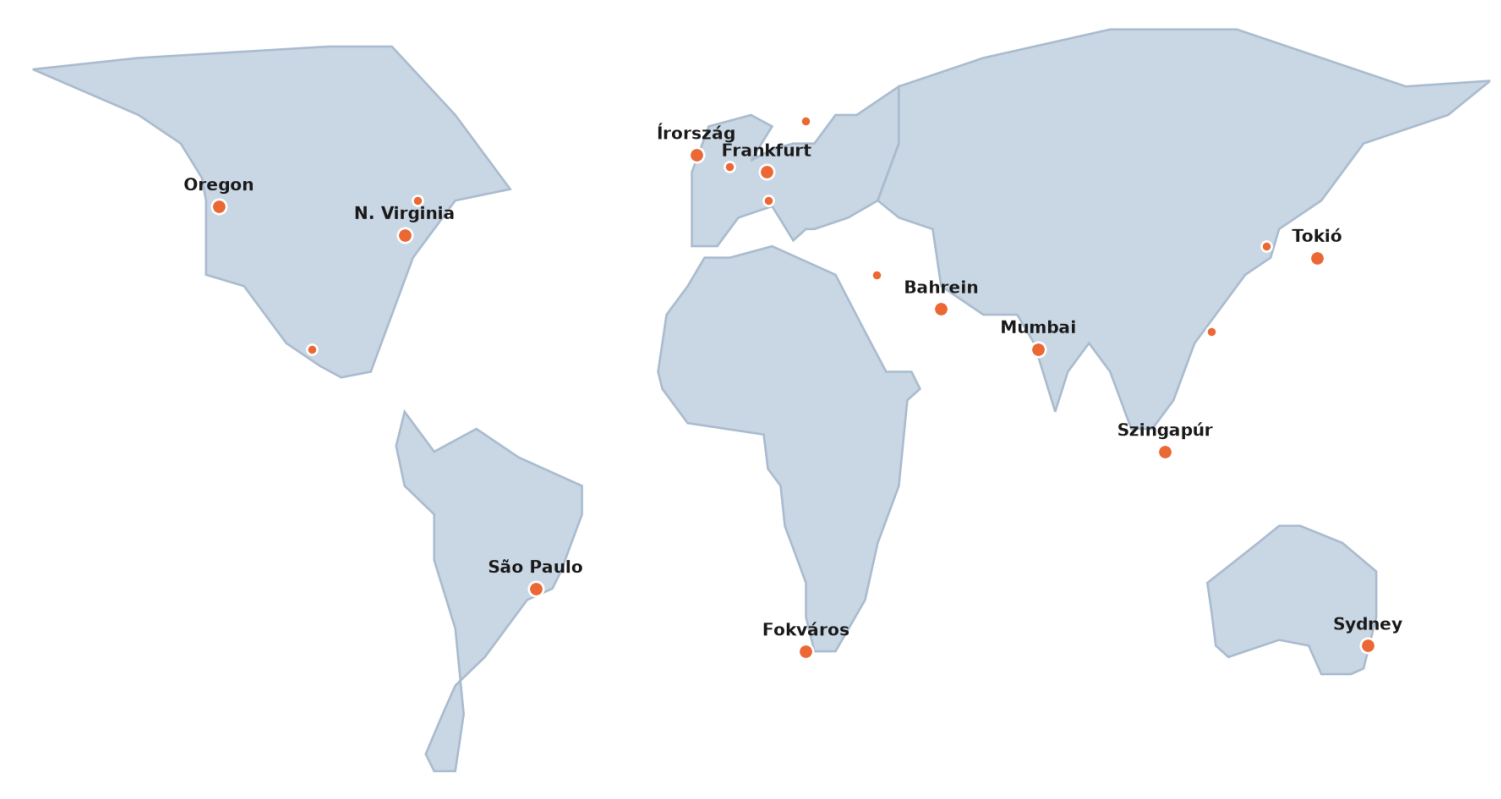
# Az AWS globális infrastruktúrája

Régiók, rendelkezésre állási zónák, adatközpontok — és hogy ez a felépítés miért nem csak mérnöki érdekesség, hanem a napi architektúrai döntéseitek alapja.

GLOBÁLIS INFRASTRUKTÚRA

# Mi az a régió?

Egy **régió** egy önálló földrajzi terület (pl. Írország, Frankfurt, Tokió), ahol az AWS saját adatközpontokat üzemeltet. Minden régió **teljesen független** a többitől — külön fizetés, külön elérhető szolgáltatások, és alapból **semmi nem másolódik át** egyik régióból a másikba.



GLOBÁLIS INFRASTRUKTÚRA

# Léptékben

**39+**
régió világszerte

**124+**
rendelkezésre állási zóna

**750+**
CloudFront edge helyszín

### Ez a szám folyamatosan nő

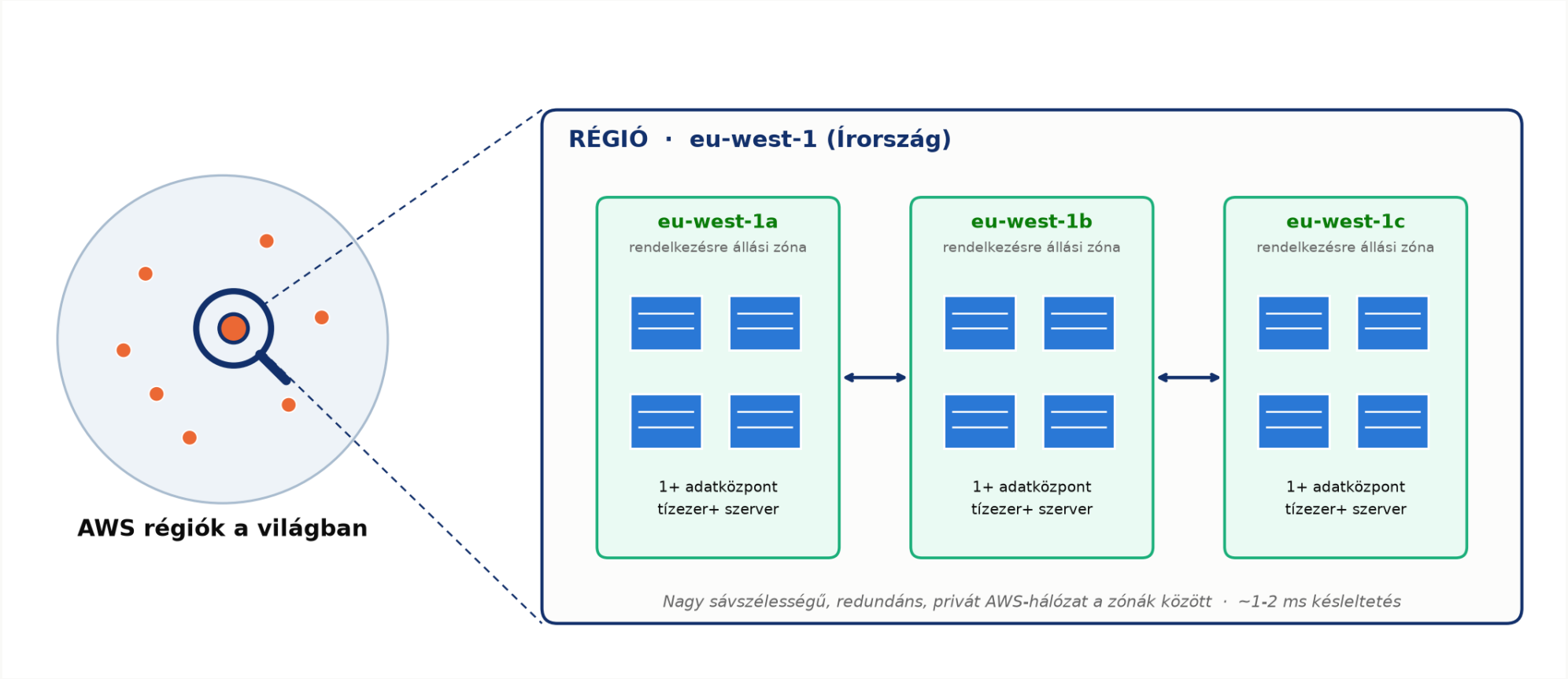
Az AWS rendszeresen jelent be új régiókat (jelenleg pl. Szaúd-Arábia és Chile van tervben) — a pontos, aktuális számot mindig az `aws.amazon.com/about-aws/global-infrastructure` oldalon érdemes ellenőrizni, ne memorizáljuk a mai számot vésett kőtáblaként.

### Hogyan válasszunk régiót egy projekthez?

Három szempont dönt tipikusan: **késleltetés** (melyik van közel a felhasználóinkhoz), **ár** (régiónként eltér), és **jogi megfelelőség** (pl. EU-s személyes adat maradjon EU-s régióban — GDPR).

GLOBÁLIS INFRASTRUKTÚRA

# Egy régió belülről: rendelkezésre állási zónák




Minden AWS régió legalább 3 rendelkezésre állási zónából (Availability Zone, AZ) épül fel.

GLOBÁLIS INFRASTRUKTÚRA

# Miért nem elég egyetlen (nagy) adatközpont?

Mert egyetlen adatközpont egyetlen hibaforrás. A zónák tudatosan úgy vannak szétválasztva, hogy **egy helyi probléma ne tudjon átterjedni** a szomszéd zónára:

### ⚡ Független áramellátás

Külön tápvonal, külön generátorok — egy áramkimaradás nem viszi le a szomszéd zónát is.

### ❄️ Független hűtés

Egy hűtési meghibásodás csak a saját zónáját érinti.

### 🌊 Földrajzi távolság

Elég messze vannak egymástól ahhoz, hogy tűz, árvíz ne érintsen kettőt egyszerre — de elég közel a gyors hálózathoz.

### 🔌 Független hálózati bekötés

Külön internetkilépési pontok — egy szolgáltatói hiba nem izolálja az egész régiót.

**A lényeg**

Két zóna egyszerre történő kiesése statisztikailag elenyésző eséllyel fordul elő — **ezért ér annyit**, ha egy alkalmazást két zónában futtatunk egy helyett.

GLOBÁLIS INFRASTRUKTÚRA

# Zónák egymás közt: gyors. Régiók egymás közt: semmi.

### ✅ Egy régión belül (zóna ↔ zóna)

- Dedikált, redundáns, privát AWS-fényvezeték köti össze őket
- ~1-2 ms késleltetés, nagyságrendileg tíz Tbps sávszélesség
- Gyakorlatilag úgy viselkedik, mintha egy hálózaton lennének

### ⛔ Két régió között

- Nincs alapból semmilyen kapcsolat, replikáció
- Ha kell, a nyilvános interneten (vagy külön fizetős backbone-on) megy a forgalom
- Minden régióközi adatmozgatás **explicit, tudatos döntés** - erről még lesz szó a mai órán

**Gyakori kezdő tévhit**

„A felhőben van, tehát biztosan mindenhol elérhető." **Nem.** Egy erőforrás pontosan ott létezik, ahova létrehoztuk — egy adott zónában, egy adott régióban. Semmi nem mozdul el onnan magától.

GLOBÁLIS INFRASTRUKTÚRA

# A teljes kép: régiókon túl

A régió/zóna páros a **számítási és tárolási** réteg. Emellett az AWS-nek van egy sokkal sűrűbb, világszerte szétszórt hálózata is — ez viszont más célt szolgál:

750+ helyszín

### CloudFront Edge Location

Statikus tartalmat (kép, videó, cache) visz nagyon közel a felhasználóhoz — nem EC2/RDS fut itt, hanem egy CDN gyorsítótár.

helyi kiterjesztés

### Local Zone

Egy régió "kihelyezett" mini-verziója, nagyvárosokhoz közel, extra alacsony késleltetéshez (pl. Los Angeles).

### Ami a mai órán számít

A projektfeladatotokhoz elsősorban a **régió/zóna** réteg lesz releváns (EC2, RDS, VPC). Az Edge/Local Zone réteget csak azért érdemes ismerni, hogy tudjátok elhelyezni a térképen, ha később találkoztok vele.

A MAI ÓRA FŐ KÉRDÉSE

# Mi replikálódik automatikusan — és mi nem?

Ugyanabban a régióban futtatva a szolgáltatásaitokat, néhány dolog "magától" túlél egy zóna-kiesést. A többségük viszont nem — hacsak ti nem terveztek rá tudatosan.

AUTOMATIKUS VS. MANUÁLIS REDUNDANCIA

# Amit NEKÜNK kell megoldanunk

**1 EC2 = 1 zóna**

## EC2

Egy EC2 példány **pontosan egy** rendelkezésre állási zónában létezik. Ha az a zóna kiesik, az a példány elérhetetlenné válik — az AWS nem indítja újra automatikusan egy másik zónában.

**Megoldás:** Auto Scaling Group, amit több zónára állítunk be, + Load Balancer, ami tudja, melyik példány érhető el.

**AZ-hez kötött**

## EBS (Elastic Block Store)

Egy EBS kötet **csak a saját zónáján belül** replikálódik (több másolat, de mind ugyanabban a zónában). Ha a zóna elérhetetlen, a kötet is az — átmenetileg vagy tartósan.

**Megoldás:** rendszeres snapshot (ami már régió-szinten, S3 mögött tárolódik), amiből másik zónában is új kötet indítható.

AUTOMATIKUS VS. MANUÁLIS REDUNDANCIA

# Amit AZ AWS automatikusan megold

alapból multi-AZ

## S3 (Simple Storage Service)

Minden feltöltött objektum **automatikusan, kérés nélkül** több zóna között replikálódik ugyanazon a régión belül. Innen az emlegetett „11 kilences" (99,999999999%) tartóssági garancia.

**Mi a teendőnk?** Semmi extra — ez a szolgáltatás alapviselkedése.

alapból multi-AZ

## DynamoDB

Az AWS natív NoSQL adatbázisa szintén automatikusan több zóna között tartja szinkronban az adatot a régión belül, teljesítményveszteség nélkül.

**Mi a teendőnk?** Semmi extra — ez is beépített garancia, nem kapcsoló, amit be kell kattintani.

### Észrevehető minta

A **tárolási** szolgáltatások (S3, DynamoDB) jellemzően automatikusan multi-AZ-osak. A **számítási** szolgáltatás (EC2) soha nem az — mert az AWS nem tudja, hogy a bennük futó alkalmazásotoknak van-e állapota, amit nem szabad csak úgy máshol újraindítani.

AUTOMATIKUS VS. MANUÁLIS REDUNDANCIA

# A köztes eset: RDS Multi-AZ

Az RDS egy külön kategória: alapból **ugyanolyan sebezhető, mint egy EC2** (egyetlen zónában fut) — de egyetlen kapcsolóval "félig-automatikussá" tehető.

alapértelmezett

### Single-AZ RDS

Egy adatbázis-példány, egy zónában. Zóna-kiesés esetén az adatbázis elérhetetlen, amíg a zóna vissza nem áll.

bekapcsolható

### Multi-AZ RDS

Az AWS **automatikusan** létrehoz és szinkronban tart egy készenléti (standby) másolatot egy másik zónában. Kiesés esetén az AWS **automatikusan átvált** a standby-ra — nekünk csak a bekapcsolás a dolgunk.

**Fontos árnyalat**

A Multi-AZ **nem** ugyanaz, mint egy Read Replica (ami teljesítmény-skálázásra való, külön kell kérni, és nem automatikus failoverhez készült). A kettőt gyakran összekeverik vizsgán is.

# Összefoglalva: mit intéz az AWS, mit intézünk mi

| | | |
|---|---|---|
| **EC2** | MANUÁLIS | 1 zóna. Multi-AZ redundanciához saját ASG + Load Balancer kell. |
| **EBS** | MANUÁLIS | Csak a saját zónáján belül replikálódik. Más zónába: snapshot kell. |
| **RDS** | FÉLIG-AUTOMATIKUS | Alapból 1 zóna — Multi-AZ opció bekapcsolásával AWS kezeli a failovert. |
| **S3** | AUTOMATIKUS | Minden objektum alapból több zóna között replikálódik. |
| **DynamoDB** | AUTOMATIKUS | Alapból több zóna között szinkronban tartott adat. |
| **Régiók között** | MINDIG MANUÁLIS | Egyetlen szolgáltatás sem lép át régióhatáron a ti kifejezett kérésetek nélkül. |

GYAKORLATBAN

# Forgatókönyv: kiesik az egyik zóna

Képzeljük el, hogy futtatjuk a féléves projektünket, és a régiónk egyik zónája (pl. `eu-west-1a`) átmenetileg elérhetetlenné válik. Mi történik?

**EC2 (1 példány, nincs ASG)**

**Leáll.** Amíg a zóna vissza nem tér, nem érhető el — nincs automatikus áthelyezés.

**EC2 (ASG, 2+ zóna)**

**Túléli.** A Load Balancer a másik zónában futó példányok felé irányítja a forgalmat, az ASG pótolja a hiányzó kapacitást.

**EBS kötet abban a zónában**

**Elérhetetlen** a zóna helyreállásáig, hacsak nincs friss snapshot másik zónában visszaállítva.

**RDS (Multi-AZ)**

**Automatikus failover** a standby-ra, jellemzően ~1-2 perces kieséssel.

**S3 bucket**

**Észre sem vesszük.** A többi zónában lévő másolatokból szolgál ki.

**A másik régió (pl. Frankfurt)**

**Teljesen érintetlen** — a régiók pontosan emiatt vannak elkülönítve egymástól.

GYAKORLATBAN

# Hogyan építsünk erre tudatosan?

- **Állapotmentes komponensek (UI, API) → Auto Scaling Group, legalább 2 zónában**, Load Balancer mögött — ezt fogjuk a gyakorlaton is felépíteni
- **Adatbázis → RDS Multi-AZ**, ha valódi rendelkezésre állás a cél, nem csak teljesítmény
- **Fájlok, statikus tartalom → S3**, ahol a multi-AZ redundancia "ingyen" jár
- **EBS-en tárolt adat → rendszeres snapshot**, különösen, ha az adat nem tartható újra elő máshonnan
- **Katasztrófa-terv (Disaster Recovery) → tudatos, régiók közötti** döntés — csak akkor, ha a kockázat ezt indokolja (lásd következő dia)

**Ökölszabály**

Zóna-szintű redundancia szinte mindig megéri (olcsó, gyors, sok helyen "ingyen" jár). Régió-szintű redundancia drága és bonyolult — csak akkor vezessük be, ha a projekt tényleg megköveteli.

GYAKORLATBAN

# Ha mégis régiók között kell mozognunk

Mivel régiók között **semmi nem történik automatikusan**, minden ilyen megoldás explicit beállítást igényel:

### S3 Cross-Region Replication

Bekapcsolható funkció: minden új objektum automatikusan átmásolódik egy másik régióba is — de csak, ha külön beállítjuk.

### AMI másolása

Egy EC2 image (AMI) manuálisan átmásolható másik régióba, hogy ott is tudjunk belőle szervert indítani.

### RDS cross-region read replica

Az adatbázisról aszinkron másolat tartható fenn egy másik régióban — tipikusan olvasási terhelés elosztására vagy DR-célra.

### Ne felejtsük

Ezekért mind **ti fizettek külön** (adatátvitel + tárolás a másik régióban is) — pont ezért nem alapértelmezett egyik sem.

ÖSSZEFOGLALÁS

# Amit ma haza kell vinni

- Egy **régió** = önálló földrajzi terület, teljesen független a többitől
- Egy régió **≥3 rendelkezésre állási zónából** épül fel — mindegyik saját áram, hűtés, hálózat
- A zónák egymással **gyorsan, ingyen** kommunikálnak; a régiók egymással **egyáltalán nem**, hacsak nem kérjük
- **Tárolási szolgáltatások (S3, DynamoDB)** jellemzően automatikusan multi-AZ-osak
- **Számítási szolgáltatások (EC2) és a nyers blokktárolás (EBS)** alapból nem azok — ezt nekünk kell megterveznünk (ASG, LB, snapshot)

KÖVETKEZŐ ÓRA: ADATKÖZPONTOK BELSŐ HÁLÓZATA

# Kérdések?

Gyakorlaton hamarosan mi magunk is Load Balancert állítunk be több zónára — ha ez a dia most elméletnek tűnt, két hét múlva már a saját kezetekkel fogjátok látni, miért számít.