🧠 AWS SAA-C03 з абсолютного нуля — підручник

Глибоке навчання: 47 розділів з образами, ілюстраціями, історією пекарні «Хмаринка», словничками й міні-тестами (відповіді — одразу під питаннями). Інтерактивна версія з підказками — на сайті.

Незалежний навчальний курс. Не пов'язаний з Amazon Web Services і не схвалений AWS. AWS і назви сервісів — торговельні марки Amazon.com, Inc. або її афілійованих осіб. Практичні питання — оригінальні, це не реальні питання іспиту.

Зміст

1 🧭 Як пройти цей курс і не загубитися

2 💻 Що всередині комп'ютера: процесор, пам'ять і диск

3 🧩 Програми, операційна система і файли

4 🔌 Мережа: як комп'ютери розмовляють

5 🌐 Інтернет і вебсайти: що відбувається, коли відкриваєш сторінку

6 🔢 IP-адреси, підмережі й CIDR без страху

7 🏗️ Як влаштований сучасний застосунок

8 🗄️ Бази даних і кеш з нуля

9 🏭 Сервери й дата-центри: де живе інтернет

10 🪆 Віртуалізація, контейнери і serverless: як один сервер стає сотнею

11 ☁️ Що таке хмара і чому всі туди переїжджають

12 🗺️ AWS по всьому світу: регіони, зони доступності й edge

13 🔑 Як виглядає AWS зсередини: акаунт, консоль, API і рахунки

14 ⚖️ Словник архітектора: надійність, швидкість, безпека, ціна

15 🪪 Хто ти і що тобі можна: IAM з нуля

16 🎫 Ролі й тимчасові ключі: безпечніше за паролі

17 🏢 Багато акаунтів і багато людей: Organizations, SCP, вхід для працівників і клієнтів

18 🔐 Шифрування з нуля: замки, ключі й конверти

19 🛡️ Атаки й охорона: від DDoS до розслідувань

20 🏘️ VPC: будуємо власне містечко в хмарі

21 🚧 Охоронці мережі: Security Groups і Network ACL

22 🛣️ Приватні дороги: VPC endpoints, PrivateLink, Flow Logs і вхід адміна

23 🌉 Як з'єднати мережі: peering, Transit Gateway, VPN і Direct Connect

24 🧭 Route 53, CloudFront і Global Accelerator: швидко для всього світу

25 🖥️ EC2: орендуємо віртуальний комп'ютер

26 💳 Як платити за сервери і не переплатити

27 ⚖️ Коли клієнтів стає багато: балансувальники й автомасштабування

28 ⚡ Serverless: Lambda і API Gateway — код без серверів

29 📦 Контейнери: ECS, EKS, Fargate, Beanstalk і Batch

30 📨 Черги й події: як частини системи розмовляють без поспіху

31 🗃️ Три види сховищ: блоки, файли й об'єкти

32 🪣 S3 докладно: безрозмірний склад

33 💿 Диски й файлові системи: EBS, instance store, EFS, FSx

34 🚚 Переїзд у хмару і гібрид: Storage Gateway, DataSync, Snowball, DMS, MGN

35 🧱 RDS: керована реляційна база

36 🚀 Aurora: база, створена для хмари

37 🗃️ DynamoDB: база без серверів на мільйони запитів

38 🧠 Кеш і спеціальні бази: ElastiCache, DocumentDB, Neptune та інші

39 🌊 Потоки даних: Kinesis, Firehose, Flink і MSK

40 📊 Аналітика: озеро даних, Glue, Athena, Redshift і дашборди

41 🤖 Штучний інтелект без магії: готові AI-сервіси, SageMaker і Bedrock

42 📈 Приладова панель, відеореєстратор і пульт: моніторинг і автоматизація

43 🛟 Коли все ламається: висока доступність і план на випадок аварії

44 💰 Розумний гаманець: з чого складається рахунок і як платити менше

45 ⏳ Від одного сервера до двох регіонів: як мислить архітектор

46 🎯 Іспит зсередини: формат, анатомія питання і розбори

47 🏁 Що далі: від книжки до сертифіката — і після нього

🧭 Як пройти цей курс і не загубитися

👴 Простими словами

Це не підручник, який треба зубрити, а дорога з табличками. Читай по одному розділу на день, проходь міні-тест у кінці — і за кілька тижнів дійдеш від нуля до іспиту.

🎯 Що ти дізнаєшся

- що таке AWS і хто такий Solutions Architect — простими словами
- який шлях веде від нуля до сертифіката і скільки часу він займе
- як читати розділи, щоб запам'ятовувати без зубріння
- хто така Оля і навіщо нам її пекарня

🥐 **Знайомся: Оля.** У Львові є маленька пекарня «Хмаринка». Її власниця Оля пече такі круасани, що зранку під дверима стоїть черга.

Якось постійна клієнтка питає: «Олю, чому я не можу замовити у вас онлайн? Я б брала щодня!» Оля ніколи не мала справи з комп'ютерами. Вона не знає, що таке сервер, хмара чи база даних. Зовсім як ти зараз.

У цьому курсі ми пройдемо шлях Олі: від «а що взагалі всередині комп'ютера?» до інтернет-пекарні, яка приймає замовлення з усього світу, не падає в Чорну п'ятницю і не розоряє власницю рахунками за хмару. А допомагатиме їй племінник Тарас — студент, який трохи розуміється на комп'ютерах.

Що таке AWS — одним абзацом

Компанія Amazon колись була просто інтернет-магазином книжок. Щоб магазин витримував мільйони покупців, вона збудувала величезні приміщення, повні комп'ютерів. А потім зрозуміла: ці комп'ютери можна **здавати в оренду** всім охочим — через інтернет, на хвилини, з оплатою лише за використане.

Так з'явився **AWS** (читається «ей-дабл-ю-ес», Amazon Web Services). Сьогодні це найбільший у світі «прокат» комп'ютерних ресурсів: понад 200 окремих послуг — **сервісів**. Один сервіс зберігає файли, інший — дані клієнтів, третій розсилає повідомлення, четвертий захищає від хакерів. Замість купувати власні комп'ютери, компанія бере в AWS рівно те, що потрібно, і платить, як за електрику — за спожите.

Такий прокат через інтернет називають **хмарою** (cloud, «клауд»). Чому саме хмара — розберемо в окремому розділі.

Хто такий Solutions Architect

Згадай архітектора будинку. Він сам не кладе цеглу. Він вирішує, **з чого** складатиметься будинок і **як** частини поєднаються, щоб будинок стояв століттями, був безпечним, зручним і вклався в бюджет.

Solutions Architect (читається «солюшнз архітект», «архітектор рішень») робить те саме з комп'ютерними системами в AWS. Замовник каже: «Хочу інтернет-пекарню». Архітектор обирає з двохсот «будматеріалів» AWS потрібні й поєднує їх так, щоб система:

- **не зламали** — безпека;
- **не падала й не губила дані** — надійність;

- **працювала швидко** — продуктивність;
- **коштувала розумно** — вартість.

Ці чотири побажання ти зустрічатимеш у кожному розділі. На них побудований і сам іспит.

Іспит SAA-C03 — коротко

SAA-C03 — код іспиту AWS Certified Solutions Architect – Associate («асоушіейт» — рівень «спеціаліст»). Головне, що варто знати вже зараз:

- **65 питань за 130 хвилин**, прохідний бал — **720 з 1000**. Питання англійською (української немає), але на кожне є пояснення в цьому курсі.
- Якщо англійська не рідна, **перед записом** на іспит можна попросити **+30 хвилин** (ESL +30). Як це зробити — в останній частині курсу.
- Майже кожне питання — міні-історія: «Компанія має… Потрібно… Яке рішення найкраще?». Команди й кнопки напам'ять знати **не треба**. Треба **розуміти, для чого кожен сервіс, і вміти обрати**.

Саме тому цей курс не змушує зубрити. Він пояснює, як усе працює, — а вибір потім стає очевидним.

Маршрут: шість кроків до сертифіката

Сайт побудований як дорога. Кожен наступний крок спирається на попередній — тож іди по порядку.



Маршрут: від глибокого розуміння до іспиту. Час — орієнтовний, при 1–1,5 години на день.

1. 🧠 **Глибоке навчання** (ця вкладка). Кожна тема з нуля: образи, картинки, історія «Хмаринки», міні-тести. Тут закладається розуміння.

2. 🎓 **Навчання.** 24 короткі уроки з тими самими образами, але стисло, плюс «як про це питають на іспиті». Після глибокого навчання вони читаються легко — як повторення знайомого.
3. 📘 **Шпаргалка.** Усі факти й цифри, розкладені по поличках, і **тригери** — пари «бачиш у питанні → відповідь».
4. 🎲 **Квіз.** Тренуєш тригери, поки 9 з 10 не будуть «знав».
5. 📝 **Тест.** 65 питань у стилі іспиту, з таймером. Ціль — **стабільно 80% і більше** в режимі «Іспит».
6. 🏆 **Іспит.** Записуєшся й складаєш.

Скільки це займе? Якщо починати зовсім з нуля й приділяти 1–1,5 години на день, реалістично — **6–10 тижнів**. Хтось швидше, хтось повільніше, і це нормально. Важливо не поспішати з глибоким навчанням: кожна година тут економить дві на повторенні.

Як читати кожен розділ

Усі розділи побудовані однаково, тож скоро ти почуватимешся в них як удома:

- 🎯 **Що ти дізнаєшся** — кілька рядків угорі: навіщо цей розділ.

- 🥐 **Історія «Хмаринки»** — що сталося в Олі і чому тема знадобилася.
- **Пояснення** — невеликими кроками, з образами з життя й ілюстраціями.
- 📖 **Словничок** — нові слова розділу з перекладом і вимовою.
- 🧩 **Міні-тест** — кілька питань. Помилятися тут корисно: пояснення покаже, що перечитати.
- ✅ **Головне** — розділ у п'яти рядках.
- 🔁 **Закріпи** — посилання на відповідний короткий урок і розділ шпаргалки.

Ще два види вставок можна при першому читанні пропускати:

- 🔍 **Для допитливих** — глибші подробиці. Корисні, але не обов'язкові.
- 🧪 **Спробуй у справжньому AWS** — маленькі практичні вправи в безкоштовному акаунті. Дуже допомагають «відчути» сервіси руками.

П'ять звичок, які роблять навчання легким

1. **Уявляй.** Кожен образ — «кухня», «склад», «охоронець» — намалюй у голові. Картинка запам'ятовується краще за визначення.

2. **Переказуй своїми словами.** Після розділу поясни уявному другові (або бабусі), про що він. Якщо не виходить — перечитай один підрозділ, а не весь розділ.
3. **Не пропускай міні-тести.** Навіть якщо здається, що все ясно.
4. **Повторюй «Головне» наступного дня.** Одна хвилина — і вчорашній розділ закріплено.
5. **Торкайся підкреслених слів.** Це 🤖 підказки: коротке пояснення будь-якого сервісу чи терміна прямо в тексті.

Прогрес (пройдені розділи, відповіді міні-тестів) зберігається на цьому пристрої. Сайт можна встановити як застосунок на телефон — він працюватиме й без інтернету.

🔍 **Для допитливих: Англійська абетка для абревіатур**

В AWS безліч скорочень: S3, EC2, IAM, VPC. Їх читають по літерах, англійською. Ось як звучать літери:

A ей · **B** бі · **C** сі · **D** ді · **E** і · **F** еф · **G** джі · **H** ейч · **I** ай · **J** джей · **K** кей · **L** ел · **M** ем · **N** ен · **O** оу · **P** пі · **Q** к'ю · **R**

ар · **S** ес · **T** ті · **U** ю · **V** ві · **W** дабл-ю · **X** екс · **Y** вай · **Z** зі

Цифра в назві зазвичай рахує однакові літери. **S3** — Simple Storage Service: три слова на S, тому «ес-срі». **EC2** — Elastic Compute Cloud: одна E і дві C, тому «і-сі-ту». Вимову кожного сервісу ми даватимемо при першій зустрічі.

Не бійся слів

Спочатку незнайомих слів буде багато. Це нормально: навіть досвідчені інженери AWS щодня щось гуглять. Ми ніколи не використовуватимемо слово, не пояснивши його. А ті самі ідеї повертатимуться знову і знову, щоразу трохи глибше, — так вони й запам'ятаються.

📖 **Словничок**

AWS («ей-дабл-ю-ес») — Amazon Web Services, найбільший хмарний провайдер.

Хмара (cloud, «клауд») — оренда комп'ютерних ресурсів через інтернет з оплатою за використане.

Сервіс — окрема послуга AWS: зберігання файлів, база даних, розсилка повідомлень тощо.

Solutions Architect — фахівець, який обирає й поєднує сервіси під вимоги замовника.

SAA-C03 — код іспиту AWS Certified Solutions Architect – Associate.

Тригер — у цьому курсі: пара «бачиш у питанні → обираєш відповідь».

🧩 Міні-тест: перевір себе

? Що насамперед перевіряє іспит SAA-C03?

- **А** Як швидко ти набираєш команди
- **Б** Чи вмієш ти обрати сервіси AWS, які виконують вимоги: безпеку, надійність, швидкість і ціну
- **В** Чи пам'ятаєш ти ціни всіх сервісів
- **Г** Чи вмієш ти програмувати

Відповідь: Б. Іспит — про вибір. Питання описує ситуацію, а ти обираєш найкраще рішення. Команди й ціни напам'ять не потрібні.

? У якому порядку найкраще проходити сайт?

А Тест → Шпаргалка → Глибоке навчання

Б Одразу квіз, бо він найшвидший

В Глибоке навчання → Навчання → Шпаргалка → Квіз → Тест → Іспит

Г Будь-який порядок однаково добрий

Відповідь: В. Кожен крок спирається на попередній: спершу розуміння, потім стислі уроки, потім факти й тригери, і лише тоді тренування на питаннях.

? Що робити з блоками «🔍 Для допитливих» при першому читанні?

А Обов'язково вивчити напам'ять

Б Можна пропустити й повернутися пізніше

В Читати лише їх, бо вони найважливіші

Відповідь: Б. Це додаткові подробиці. Головне завжди в основному тексті.

? Який результат практичного тесту — добрий знак готовності до іспиту?

А 50%

Б Стабільно 80% і більше в режимі «Іспит»

В 100% з першої спроби

Г Будь-який, головне — пройти тест хоч раз

Відповідь: Б. Прохідний бал іспиту — 720 з 1000. Стабільні 80%+ на практиці дають запас на хвилювання й незнайомі формулювання.

✅ **Головне**

AWS — прокат комп'ютерних ресурсів через інтернет: понад 200 сервісів, оплата за використане.

Solutions Architect обирає й поєднує сервіси так, щоб система була безпечна, надійна, швидка й недорога.

Іспит — про **вибір**, а не про команди: 65 питань, 130 хвилин, прохідний бал 720 з 1000.

Маршрут: Глибоке навчання → Навчання → Шпаргалка → Квіз → Тест → Іспит.

Уявляй образи, переказуй своїми словами, проходь міні-тести, повторюй «Головне» наступного дня.

💻 Що всередині комп'ютера: процесор, пам'ять і диск

👴 Простими словами

Комп'ютер — як кухня. Процесор — кухар, оперативна пам'ять — робочий стіл (швидкий, але все зникає, коли вимкнути світло), диск — холодильник (повільніше, зате зберігає надовго).

🎯 Що ти дізнаєшся

- з яких трьох головних частин складається будь-який комп'ютер — від телефона до сервера AWS
- чим оперативна пам'ять відрізняється від диска і чому це так важливо
- що таке біт, байт, мегабайт і терабайт — на прикладах з життя
- що саме ти «замовлятимеш», коли орендуватимеш комп'ютер в AWS

🥐 Оля вирішила: у «Хмаринки» буде сайт, де можна замовити круасани. Перше, що вона питає Тараса: «А на чому цей сайт працюватиме? На моєму ноутбуці?»

«На сервері, — усміхається Тарас. — Але сервер — теж комп'ютер, тільки потужніший. Давай спершу розберемося, що всередині будь-якого комп'ютера. Тоді все інше стане на свої місця».

Комп'ютер — це кухня

Телефон, ноутбук, величезний сервер в AWS — усі вони влаштовані однаково. Уяви кухню ресторану. У ній є три головні речі:

- **Процесор** — це **кухар**. Він виконує всю роботу: рахує, порівнює, перекладає, малює картинку на екрані. Кухар неймовірно швидкий — мільярди простих дій за секунду.
- **Оперативна пам'ять** — це **робочий стіл** кухаря. Усе, з чим він працює просто зараз, лежить на столі під рукою. Брати зі столу — миттєво. Але стіл невеликий, і коли кухня закривається (комп'ютер вимикається), стіл прибирають начисто.

- **Диск** — це **холодильник і комора**. Тут усе зберігається надовго: рецепти, продукти, записи. Місця багато, і вимкнене світло нічого не зіпсує. Але щоб узяти щось із комори, треба сходити й принести — це помітно довше, ніж зі столу.



Кухар ніколи не готує просто в холодильнику. Спершу продукти переносять на стіл, і лише там з ними працюють. Так само й комп'ютер: коли ти відкриваєш фото, воно спочатку копіюється з диска в оперативну пам'ять, і вже там процесор його показує, повертає чи

прикрашає. Коли натискаєш «Зберегти», результат повертається в комору — на диск.

Тепер зрозуміло, звідки беруться знайомі скарги:

- **«Комп'ютер гальмує, бо мало пам'яті»** — стіл замалий, кухар постійно бігає до холодильника.
- **«Слабкий процесор»** — кухар повільний або один на велике замовлення.
- **«Закінчилося місце»** — комора забита вщерть, нові продукти нікуди покласти.

Процесор: кухар, що рахує мільярдами

Процесор англійською — **CPU** (читається «сі-пі-ю», Central Processing Unit — «центральний обчислювальний пристрій»).

Два слова про нього знадобляться весь курс:

- **Ядра** (cores, «корз»). У сучасному процесорі не один кухар, а кілька — їх називають ядрами. Чотири ядра — четверо кухарів, які готують одночасно. У серверів ядер бувають десятки.
- **Частота** (вимірюється в гігагерцах, ГГц). Це те, як швидко рухається кожен кухар. 3 ГГц — приблизно три мільярди кроків за секунду.

В AWS ти не бачитимеш справжніх процесорів. Тобі виділятимуть **vCPU** (читається «ві-сі-пі-ю», virtual CPU) —

«віртуального кухаря», частку справжнього процесора. Замовляєш 2 vCPU — отримуєш двох кухарів на свою кухню.

А ще процесори бувають різних «шкіл кулінарії» — **архітектур**. Одна школа — Intel і AMD (у більшості ноутбуків). Інша — ARM (у телефонах). AWS робить власні процесори на ARM — **Graviton** («гравітон»). Вони готують ту саму страву дешевше, тож на іспиті Graviton часто з'являється у відповідях про економію.

Оперативна пам'ять: робочий стіл

Оперативна пам'ять англійською — **RAM** (читається «рем», Random Access Memory). Її вимірюють у гігабайтах: у телефоні 6–12 ГБ, у ноутбуці 8–32 ГБ, у великих серверах — сотні гігабайтів і навіть терабайти.

Дві властивості, які треба запам'ятати:

1. **RAM дуже швидка** — у сотні й тисячі разів швидша за диск.
2. **RAM тимчасова**. Вимкнули живлення — усе на столі зникло. Тому програми просять «зберегти» роботу: зберегти — означає перенести зі столу в комору.

Ця пара властивостей — основа цілого класу сервісів AWS. Коли щось треба віддавати блискавично, дані тримають саме в пам'яті — це називається **кеш**

(«записник на столі»). А ще є сервери з «величезним столом» — для задач, де весь масив даних мусить лежати під рукою.

Диск: холодильник і комора

Диск — це **постійне** сховище: дані лишаються, навіть якщо вимкнути живлення. Буває двох головних видів:

- **HDD** (читається «ейч-ді-ді», жорсткий диск) — усередині обертаються металеві пластини, а над ними бігає голівка, як у старому програвачі платівок. Дешевий і місткий, але повільний, особливо коли треба дістати багато дрібних речей з різних полиць.
- **SSD** (читається «ес-ес-ді») — без рухомих частин, як велика флешка. Набагато швидший, але кожен гігабайт коштує дорожче.

В AWS ти обиратимеш диски під задачу: швидкі SSD — для баз даних і системи, дешеві HDD — для великих архівів, які читають підряд. Детально — у розділі про диски.

> 🔍 **Для допитливих: Чому диск повільніший за пам'ять, якщо SSD такий швидкий?**
>
> Навіть найкращий SSD відповідає за десятки–сотні мікросекунд (мільйонних часток секунди), а

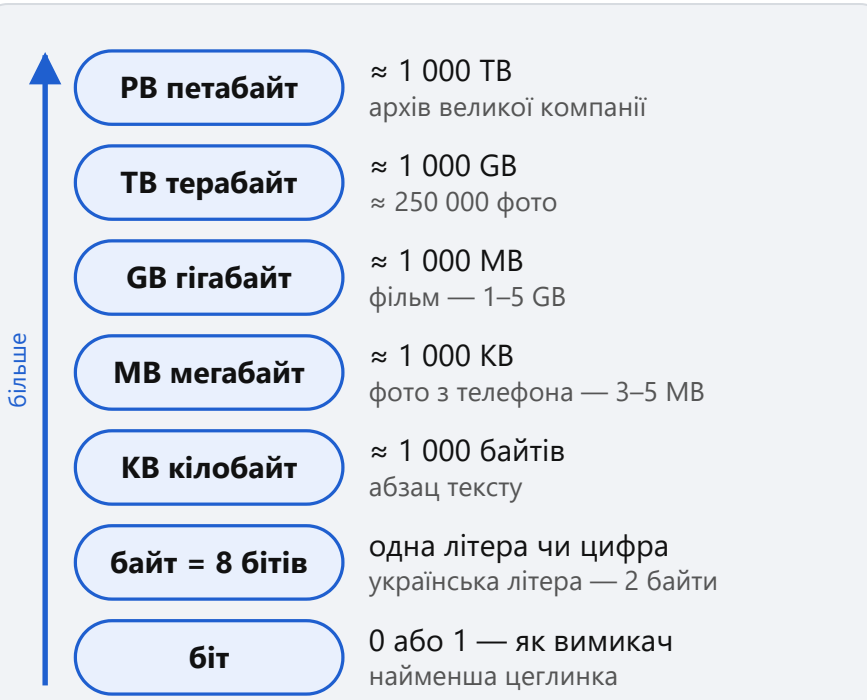
оперативна пам'ять — приблизно за сотню наносекунд (мільярдних). Різниця — сотні й тисячі разів. Для людини обидва варіанти миттєві, але процесор за час очікування диска встигає зробити сотні тисяч дій. Тому все «гаряче» тримають у пам'яті, а на диску — все, що має пережити вимкнення.

Біти й байти: як комп'ютер міряє інформацію

Усередині комп'ютера все — текст, фото, музика — записано нулями й одиницями. Одна така позначка, 0 або 1, називається **біт** (bit). Уяви вимикач світла: увімкнено — 1, вимкнено — 0.

Один біт мало що вміщує, тому біти збирають у групи по вісім. Вісім бітів — це **байт** (byte). Одного байта досить, щоб записати одну англійську літеру чи цифру. Українська літера в поширеному кодуванні займає два байти.

Далі — як з грамами й кілограмами, тільки крок приблизно в тисячу разів:



Ці одиниці ти бачитимеш скрізь: «об'єкт у сховищі — до 50 TB», «диск на 500 GB», «повідомлення — до 1 MB».

🔍 Для допитливих: 1000 чи 1024? KB і KiB

Комп'ютери рахують степенями двійки, тому «чесний» кілобайт — 1024 байти. Щоб не плутатися, придумали окремі назви: KiB, MiB, GiB (кібібайт, мебібайт, гібібайт) — це саме 1024-кратні одиниці, а KB, MB, GB — тисячні. AWS у документації часто пише GiB і TiB (наприклад, для дисків і пам'яті).

Різниця — кілька відсотків, для іспиту вважай їх однаковими.

Біти для швидкості, байти для розміру

Тут новачки часто плутаються. **Розмір** файлів міряють у **байтах** (MB, GB). А **швидкість** інтернету — у **бітах за секунду**: Mbps («мегабіт за секунду»), Gbps («гігабіт за секунду»). Зверни увагу на маленьку «b» — біти, і велику «B» — байти.

Правило перекладу просте: **швидкість у бітах ділимо на 8** — отримуємо байти.

- Домашній інтернет 100 Mbps → приблизно 12 мегабайтів за секунду.
- Фільм на 3 GB (3 000 MB) такий канал завантажить приблизно за 250 секунд — трохи більше чотирьох хвилин.

Це знадобиться пізніше, коли рахуватимемо, скільки днів займе перенесення даних компанії в хмару через інтернет — і коли вигідніше відвезти диски фізично.

GPU: бригада помічників

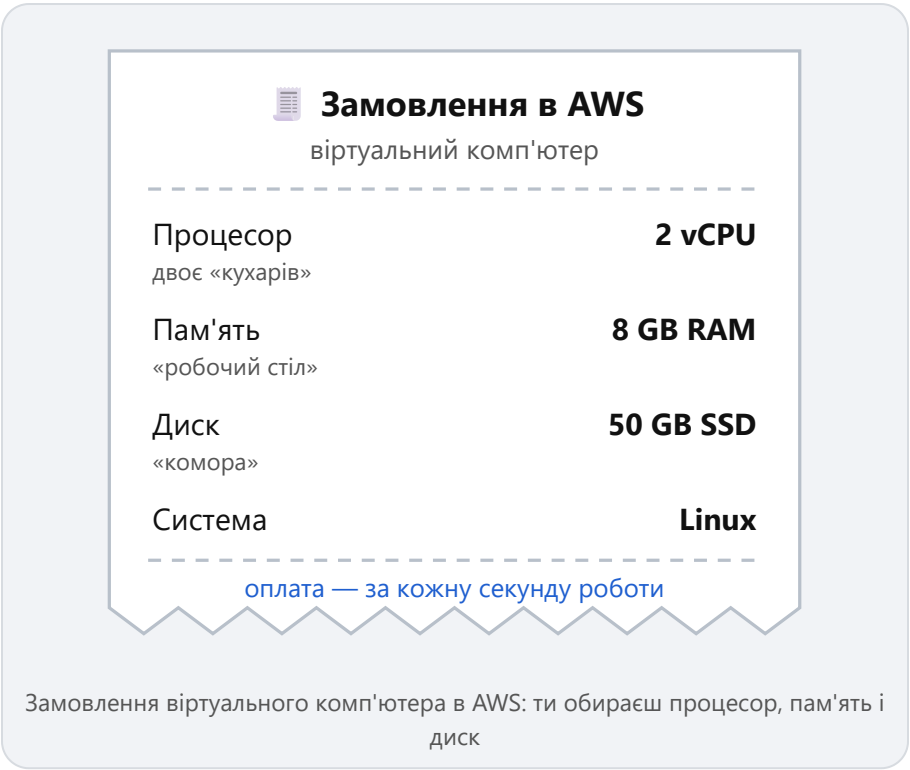
Є ще один вид «кухаря» — **GPU** (читається «джі-пі-ю», Graphics Processing Unit, графічний процесор).

Звичайний процесор — кілька майстрів, кожен уміє все. GPU — тисячі простих помічників, які вміють лише нарізати, зате всі разом і одночасно.

Така бригада ідеальна там, де одну просту дію треба повторити мільйони разів: малювання графіки, обробка відео, **навчання штучного інтелекту**. В AWS для цього є окремі сервери з GPU.

Що ти «замовлятимеш» в AWS

Сервер — той самий комп'ютер: кухар, стіл і комора, тільки без екрана й клавіатури, дуже потужний і ввімкнений цілодобово. Коли ти орендуватимеш комп'ютер в AWS, замовлення виглядатиме майже як чек у ресторані:



Замовлення віртуального комп'ютера в AWS: ти обираєш процесор, пам'ять і диск

Тепер ти розумієш кожен рядок цього «чека». Скільки кухарів (vCPU), який завбільшки стіл (RAM), яка комора (диск) — і платиш лише за час, поки кухня працює. Про те, як саме такий комп'ютер стає «віртуальним», поговоримо в розділі про віртуалізацію.

📖 **Словничок**

Процесор, CPU («сі-пі-ю») — «кухар»: виконує обчислення.

Ядро (core, «кор») — окремий «кухар» усередині процесора.

vCPU («ві-сі-пі-ю») — віртуальний процесор: частка справжнього, яку дає хмара.

Graviton («гравітон») — процесори AWS на архітектурі ARM: та сама робота дешевше.

Оперативна пам'ять, RAM («рем») — «робочий стіл»: дуже швидка, але тимчасова.

Диск — «холодильник і комора»: повільніший, зате зберігає назавжди.

SSD («ес-ес-ді») — швидкий диск без рухомих частин.

HDD («ейч-ді-ді») — дешевий місткий диск з обертовими пластинами.

Біт / байт — 0 або 1 / вісім бітів.

KB, MB, GB, TB, PB — кіло-, мега-, гіга-, тера-, петабайт; кожен ≈ у 1000 разів більший.

Mbps, Gbps — мегабіти й гігабіти за секунду: швидкість мережі.

GPU («джі-пі-ю») — «бригада помічників» для графіки та штучного інтелекту.

🧩 Міні-тест: перевір себе

? Комп'ютер раптово вимкнули. Що зникне?

- **А** Усі файли на диску
- **Б** Те, що було лише в оперативній пам'яті й не встигло зберегтися на диск
- **В** Процесор
- **Г** Нічого, комп'ютер пам'ятає все

Відповідь: Б. Оперативна пам'ять — робочий стіл: коли кухня закривається, його прибирають. Диск — комора: там усе лишається.

? Сайт гальмує, бо «кухар» постійно бігає до холодильника: даних, з якими йде робота, більше, ніж уміщує стіл. Чого бракує найімовірніше?

А Місця на диску

Б Оперативної пам'яті

В Моніторів

Г Клавіатури

Відповідь: Б. Коли стіл (RAM) замалий, дані доводиться весь час підтягувати з повільної комори (диска). Більше пам'яті — менше пробіжок.

? Скільки приблизно мегабайтів в одному гігабайті?

А 8

Б 100

В Близько 1000

Г Мільйон

Відповідь: В. Кожна наступна одиниця приблизно в 1000 разів більша: 1 GB ≈ 1000 MB.

? Інтернет-канал — 100 Mbps. Скільки приблизно мегабайтів файлу пройде за секунду?

А 100 MB

Б 800 MB

В Близько 12 MB

Г 1 GB

Відповідь: В. Швидкість мережі рахують у бітах. Ділимо на 8: 100 / 8 ≈ 12,5 мегабайта за секунду.

? Для чого найкраще підходить GPU?

А Для зберігання фото

Б Для задач, де однакову просту дію треба повторити мільйони разів: графіка, відео, навчання штучного інтелекту

В Для підключення до Wi-Fi

Г Для збереження даних після вимкнення

Відповідь: Б. GPU — тисячі простих помічників, які працюють одночасно. Саме це потрібно для графіки й штучного інтелекту.

✅ **Головне**

Будь-який комп'ютер — кухня: **процесор** (кухар), **оперативна пам'ять** (стіл), **диск** (холодильник і комора).

Пам'ять — швидко, але тимчасово. Диск — повільніше, зате назавжди.

Біт → байт (8 бітів) → KB → MB → GB → TB → PB, кожен крок ≈ ×1000.

Швидкість мережі — у **бітах** (Mbps), розмір — у **байтах** (MB): ділимо на 8.

В AWS ти орендуватимеш саме такі «кухні»: обираєш vCPU, пам'ять і диск.

🧩 Програми, операційна система і файли

👴 Простими словами

Операційна система — як адміністратор у будинку: роздає програмам місце, пам'ять і доступ до диска. Програми — мешканці, а файли — їхні речі, складені по шафах-папках.

🎯 Що ти дізнаєшся

- що таке програма і чому вона схожа на рецепт
- навіщо комп'ютеру операційна система і чому сервери майже завжди працюють на Linux
- що таке файли, папки й формати
- чому оновлення і «патчі» — це питання безпеки
- як керують комп'ютером, до якого не можна підійти

🥐 Тарас ставить на Олин ноутбук програму для обліку замовлень. «Зачекай, — дивується Оля, — що означає "встановити"? І чому ти весь час щось оновлюєш?» Тарас відкладає ноутбук: «Добре. Спершу — що таке програма взагалі».

Програма — це рецепт

Пам'ятаєш кухню з попереднього розділу? Кухар (процесор) дуже швидкий, але сам нічого не вигадує. Йому потрібен **рецепт** — точна послідовність кроків: «візьми борошно, додай масло, вимішуй 5 хвилин».

Програма і є таким рецептом для комп'ютера — набором інструкцій. Пишуть програми люди-програмісти спеціальними мовами (Python, Java, JavaScript та інші). Браузер, у якому ти зараз читаєш, — програма. Калькулятор — програма. Сайт «Хмаринки» теж буде програмою, яка працює на сервері.

Коли рецепт не просто лежить у книжці, а за ним **прямо зараз** готують — це називається **процес**. На одному комп'ютері одночасно працюють сотні процесів: кухар блискавично перемикається між ними, і тобі здається, що все відбувається паралельно.

Операційна система — керуючий рестораном

Хтось має вирішувати, яку страву кухарі готують цієї миті, скільки місця на столі дати кожній і де в коморі що лежить. Цим займається **операційна система** (ОС, англійською OS — «оу-ес»).

ОС — найголовніша програма комп'ютера, **керуючий рестораном**:

- розподіляє процесор і пам'ять між програмами;
- тримає лад у коморі — керує файлами на диску;
- «розмовляє» з пристроями: екраном, мережею, принтером;
- пильнує, хто має право заходити на кухню, — керує користувачами й правами.



Шари комп'ютера: залізо внизу, над ним операційна система, над нею програми

Найвідоміші операційні системи: **Windows** (більшість домашніх комп'ютерів), **macOS** (комп'ютери Apple), **Android** та **iOS** (телефони) і **Linux** («лінукс»).

Linux — безкоштовна система з відкритим кодом: її може переглянути й удосконалити будь-хто. Вона стабільна, економна й добре працює без екрана, тому на **Linux працює більшість серверів у світі**, зокрема більшість серверів в AWS. **Windows Server** обирають, коли потрібні саме віндоусівські програми — наприклад, спільні папки для офісу з Windows чи корпоративний каталог користувачів Active Directory. Ця різниця ще

знадобиться на іспиті: «спільні файли для Linux» і «спільні файли для Windows» — різні сервіси AWS.

Файли, папки й формати

Файл — це окремий документ у коморі, у якого є ім'я: `krouasan.jpg`, `menu.pdf`, `zamovlennia.csv`. **Папки** (їх ще називають каталогами, directories) — полиці й коробки, у які файли розкладено. Шлях до файлу — ніби адреса полиці:

- у Windows: `C:\Users\Olia\Retsepty\krouasan.txt`
- у Linux: `/home/olia/retsepty/krouasan.txt`

Закінчення імені після крапки — **розширення** — підказує формат, тобто як усередині записані дані і якою програмою їх відкривати:

- `.txt` — простий текст; `.pdf` — документ; `.jpg`, `.png` — зображення; `.mp4` — відео;
- `.csv` — таблиця, записана простим текстом, де значення розділено комами;
- `.json` — дані для програм (познайомимося в розділі про вебсайти).

Формати ще повернуться в розділі про аналітику: там ти дізнаєшся, чому той самий масив даних у форматі

Parquet можна проаналізувати в десятки разів дешевше, ніж у CSV.

Встановлення, оновлення і патчі

Встановити програму — означає скопіювати її файли в комору й «познайомити» з керуючим: де вона лежить, як її запускати.

Програми пишуть люди, а люди помиляються. Деякі помилки — просто незручності, а деякі — **дірки в безпеці**: через них зловмисник може проникнути в комп'ютер. Це як виявити, що замок на задніх дверях відчиняється шпилькою.

Коли таку дірку знаходять, виробник випускає **патч** («латку») — оновлення, що її закриває. Сервер, на якому роками не ставили патчів, — будинок із розбитим вікном: рано чи пізно хтось залізе.

💡 Хто ставить патчі в хмарі — ти чи AWS, — залежить від сервісу. Якщо орендуєш «голий» віртуальний комп'ютер, патчі операційної системи — твоя турбота. Якщо береш керований сервіс (наприклад, готову базу даних), ОС латає AWS. Цей поділ називається моделлю спільної відповідальності — і на іспиті про неї питають постійно.

Користувачі й права

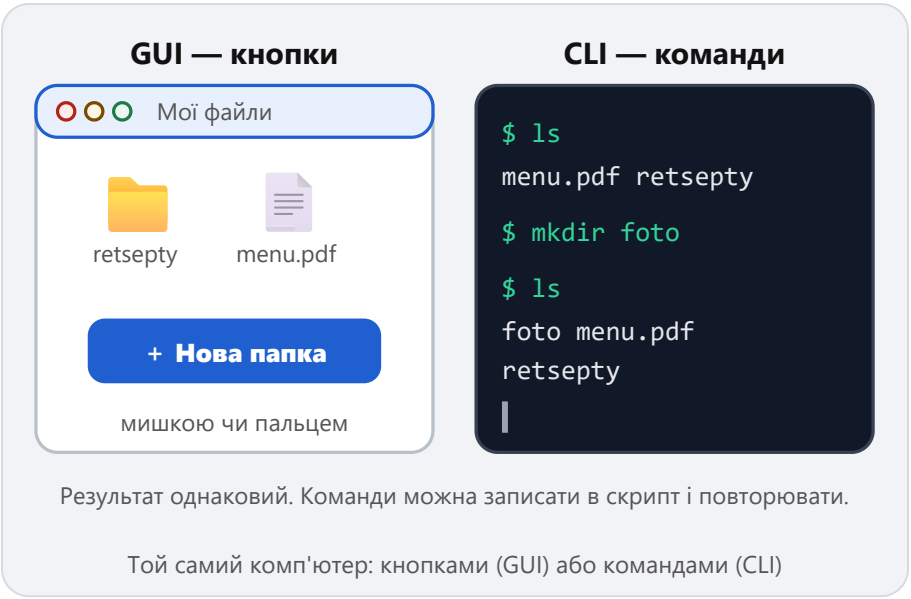
Операційна система знає, хто за нею працює. У кожного — свій **обліковий запис** (акаунт) з ім'ям і паролем, і свої права: бухгалтер бачить облік, але не може видалити систему.

Є й особливий запис з повною владою: **адміністратор** у Windows, **root** («рут») у Linux. Це майстер-ключ від усього будинку. Правило досвідчених людей: майстер-ключем не користуються щодня — лише для рідкісних особливих справ. Те саме правило буде і в AWS: там теж є «root» — і його теж ховають.

Два способи керувати: кнопки й команди

Найзвичніше керувати комп'ютером мишкою чи пальцем: вікна, кнопки, значки. Це **графічний інтерфейс** (GUI, «джі-ю-ай»).

Є й інший спосіб — **командний рядок** (CLI, «сі-ел-ай», command line interface). Ти набираєш коротку команду, комп'ютер виконує й відповідає текстом. Наприклад, у Linux команда `ls` показує файли в папці, а `mkdir retsepty` створює папку «retsepty».



Результат однаковий. Команди можна записати в скрипт і повторювати.

Той самий комп'ютер: кнопками (GUI) або командами (CLI)

Навіщо команди, якщо є кнопки? Три причини:

1. **Сервери не мають екрана.** Вони стоять у дата-центрі, і керують ними через мережу.
2. **Команди можна записати.** Список команд у файлі — **скрипт** — комп'ютер виконає сам, скільки завгодно разів, без помилок і втоми.
3. **Швидкість.** Налаштувати сто серверів кнопками — тиждень. Скриптом — хвилини.

Керувати на відстані: SSH і RDP

Щоб дістатися до сервера через мережу, адміністратори використовують:

- **SSH** («ес-ес-ейч») — захищений командний рядок для Linux. Працює через «двері» номер **22** (що таке номери дверей — портів, — у наступному розділі).
- **RDP** («ар-ді-пі») — віддалений робочий стіл Windows, ніби дивишся на екран сервера здалеку. Двері номер **3389**.

Відкриті для всього інтернету двері 22 чи 3389 — улюблена ціль зловмисників. В AWS є безпечніший спосіб зайти на сервер узагалі без відчинених дверей — Session Manager. Ми до нього дійдемо.

Скрипт за розкладом — і чому на іспиті він програє

Уяви: щоночі о 2:00 треба копіювати замовлення в архів. Можна написати скрипт і попросити керуючого запускати його за розкладом (у Linux розклад задач називається **cron** — «крон»).

Це працює. Але такий скрипт живе на одному сервері: сервер зламався — архівація тихо зупинилась, і ніхто не помітив. У хмарі для більшості таких задач є **керовані сервіси**, які роблять те саме надійніше й без твого нагляду. Тому на іспиті варіант «напишемо скрипт на сервері й запускатимемо cron» майже завжди **неправильний**, якщо поруч є керований сервіс.

📖 Словничок

Програма — набір інструкцій для комп'ютера, «рецепт».

Процес — програма, яка виконується просто зараз.

Операційна система, ОС («оу-ес») — головна програма-«керуючий»: розподіляє ресурси, керує файлами, пристроями й правами.

Linux («лінукс») — безкоштовна ОС, на якій працює більшість серверів.

Файл / папка — документ з ім'ям / полиця, на яку документи розкладено.

Розширення — закінчення імені файлу (`.jpg` , `.csv`), що вказує формат.

Патч — оновлення, яке виправляє помилку, часто — дірку в безпеці.

root / адміністратор («рут») — обліковий запис з повною владою, «майстер-ключ».

GUI («джі-ю-ай») — керування кнопками й вікнами.

CLI («сі-ел-ай») — керування текстовими командами.

Скрипт — записаний список команд, який комп'ютер виконує сам.

SSH («ес-ес-ейч», порт 22) / **RDP** («ар-ді-пі», порт 3389) — віддалене керування Linux / Windows.

cron («крон») — розклад задач у Linux.

🧩 Міні-тест: перевір себе

? Що робить операційна система?

- **А** Лише показує гарні шпалери на екрані
- **Б** Розподіляє процесор, пам'ять і диск між програмами, керує файлами, пристроями й правами
- **В** Замінює процесор, коли він перегрівається
- **Г** Нічого, це просто ще одна гра

Відповідь: Б. ОС — «керуючий рестораном»: без неї програми не змогли б поділити кухню між собою.

? На якій операційній системі працює більшість серверів в інтернеті та в AWS?

- А Windows 11
- Б macOS
- **В Linux**
- Г Android

Відповідь: В. Linux безкоштовна, стабільна й чудово працює без екрана — тому вона домінує на серверах.

? Навіщо встановлювати патчі безпеки?

- А Щоб комп'ютер став гарнішим
- **Б Щоб закрити знайдені в програмах «дірки», через які можуть зламати**
- В Щоб звільнити місце на диску
- Г Патчі потрібні лише для ігор

Відповідь: Б. Патч — латка на дірку. Сервер без патчів — будинок із розбитим вікном.

? Як адміністратор найчастіше керує Linux-сервером на відстані?

- **А** Через SSH — захищений командний рядок через мережу
- **Б** Приходить у дата-центр з мишкою
- **В** Надсилає лист поштою
- **Г** Ніяк, сервером не керують

Відповідь: А. SSH (порт 22) дає командний рядок сервера через мережу. Для Windows є схожий RDP (порт 3389).

? Щоночі треба архівувати дані. Який варіант на іспиті AWS зазвичай програє?

- **А** Керований сервіс AWS, що робить це сам
- **Б** Власний скрипт на одному сервері, запущений через cron
- **В** Готова функція резервного копіювання
- **Г** Автоматичне правило за розкладом у керованому сервісі

Відповідь: Б. Скрипт на одному сервері — єдина точка відмови і постійний нагляд. Керований сервіс надійніший і простіший — саме його обирає іспит.

✅ Головне

Програма — «рецепт», процес — «готування за рецептом просто зараз».

Операційна система — «керуючий»: ділить ресурси, тримає файли, стереже права.

Більшість серверів — на **Linux**; Windows Server — коли потрібні саме віндоусівські програми.

Патчі закривають дірки в безпеці; хто їх ставить у хмарі — залежить від сервісу.

Сервери без екрана керуються командами (SSH, RDP); скрипт на одному сервері на іспиті програє керованому сервісу.

🔌 Мережа: як комп'ютери розмовляють

👴 Простими словами

Мережа — це пошта між комп'ютерами. Кожен має адресу, повідомлення ріжуться на маленькі листи-пакети, а маршрутизатори, як поштові відділення, передають їх далі, поки ті не дійдуть.

🎯 Що ти дізнаєшся

- як дані подорожують мережею — маленькими посилками-пакетами
- що таке IP-адреса і порт — адреса поштомата і номер комірки
- що роблять роутер і комутатор
- чим TCP відрізняється від UDP
- чому «швидкий інтернет» і «швидка відповідь» — різні речі

🥐 У «Хмаринці» стоять ноутбук, касовий апарат і принтер для чеків. Оля натискає «друкувати» на ноутбуці — і чек виповзає з принтера в іншому кутку. «Як ноутбук передав чек, якщо вони нічим не з'єднані?» — «З'єднані, — каже Тарас, — через Wi-Fi. Це мережа. А інтернет — це та сама мережа, тільки на весь світ».

Мережа — це пошта між комп'ютерами

Мережа — це кілька пристроїв, з'єднаних так, щоб вони могли обмінюватися даними. Кабелем чи через Wi-Fi — неважливо.

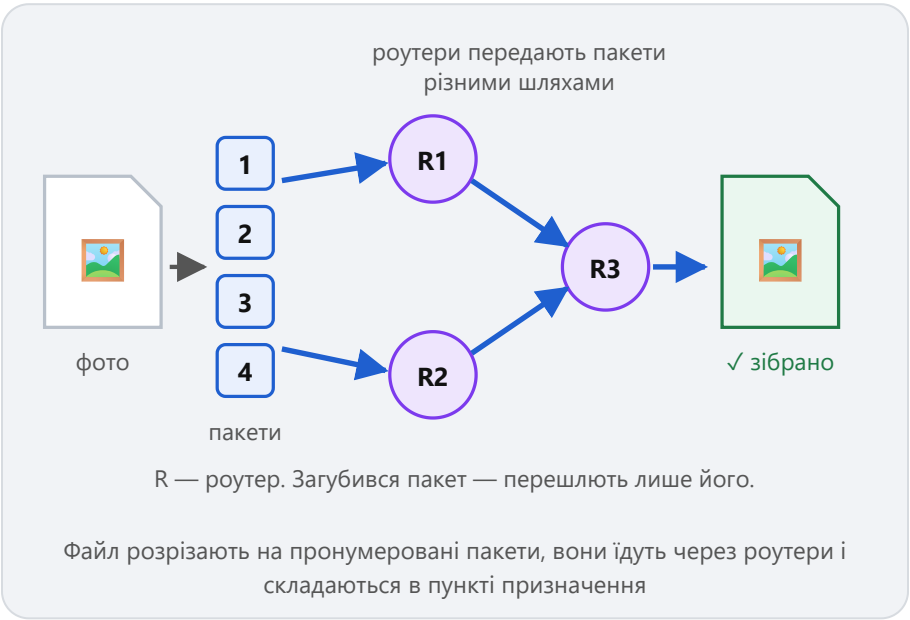
- Ноутбук, телефон, телевізор і принтер удома чи в пекарні утворюють **локальну мережу** (LAN, «лан»).
- **Інтернет** — це мережа мереж: мільйони локальних мереж по всьому світу, з'єднаних між собою.

Найкраще уявляти мережу як **пошту** — наприклад, Нову пошту. Дані — це посилки, у кожного пристрою є адреса, а по дорозі посилки проходять сортувальні центри.

Дані їдуть посилками — пакетами

Великий файл ніхто не везе одним шматком. Його розрізають на маленькі посилки — **пакети** (приблизно по півтори тисячі байтів). На кожному пакеті — адреса відправника, адреса одержувача й порядковий номер.

Пакети можуть їхати різними маршрутами й приходити не по черзі. Одержувач складає їх за номерами — як меблі, що приїхали в пронумерованих коробках. Загубилася одна коробка — перевозять лише її, а не всю шафу.



IP-адреса — адреса поштомата

Щоб пакет доїхав, у кожного пристрою в мережі має бути адреса. Це **IP-адреса** (читається «ай-пі», від Internet Protocol). Виглядає як чотири числа через крапку:

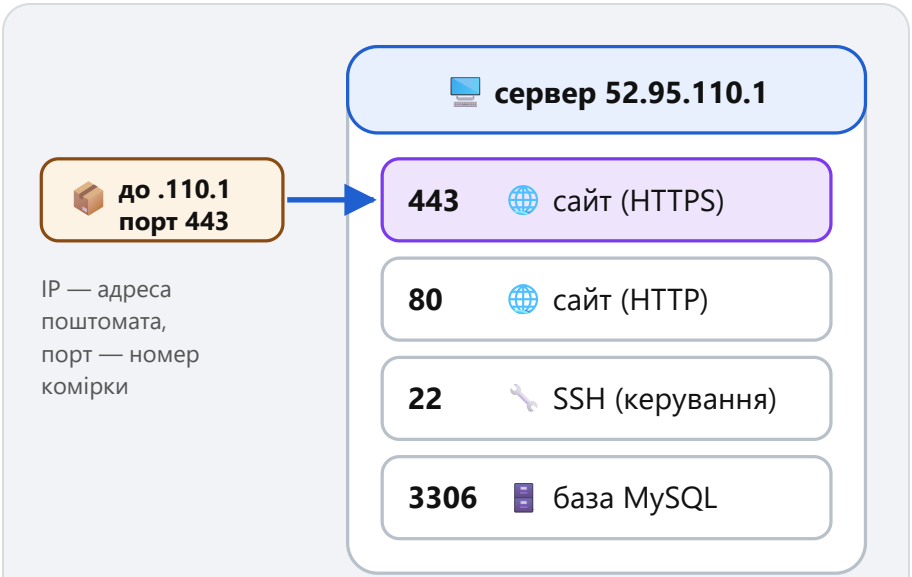
- `192.168.1.25` — типова адреса ноутбука в домашній мережі;
- `52.95.110.1` — адреса одного із серверів в інтернеті.

Без IP-адреси пакет нікуди не доставити. Як влаштовані адреси всередині і чому числа лише від 0 до 255 — у розділі про IP-адреси.

Порт — номер комірки поштомата

На одному комп'ютері одночасно працює багато програм, і кожна чекає свою пошту: сайт, пошта, база даних. Як пакет знає, **якій саме** програмі він призначений?

Для цього є **порт** — номер «дверей» чи комірки. Уяви поштомат: адреса поштомата — це IP, а номер комірки — порт. Посилка на `52.95.110.1:443` означає: «поштомат 52.95.110.1, комірка 443».



Один сервер — багато програм: кожна чекає пакети у своїй «комірці» — порті

Деякі номери домовлено використовувати для певних програм. Ці знадобляться найчастіше:

Порт	Хто там «живе»
80	вебсайти (HTTP)
443	захищені вебсайти (HTTPS)
22	SSH — керування Linux-сервером
3389	RDP — віддалений робочий стіл Windows
53	DNS — «довідкова» імен сайтів
3306	база даних MySQL
5432	база даних PostgreSQL

Навіщо це архітектору? У правилах захисту постійно пишуть саме порти: «пускати всіх на 443», «базу даних на

3306 — лише від наших серверів». Хто знає порти, той читає такі правила, як газету.

Роутер і комутатор: сортувальні центри

Пакети не летять напряму від тебе до сервера. Їх передають із рук у руки спеціальні пристрої:

- **Комутатор** (switch, «світч») — сортує пакети **всередині** однієї локальної мережі, як кур'єр у межах одного будинку.
- **Роутер**, або маршрутизатор (router, «раутер»), — сортувальний термінал **між мережами**. Він дивиться на адресу одержувача й вирішує, куди передати пакет далі: «на Київ, звідти — на Франкфурт».

Домашній Wi-Fi-роутер — теж роутер: він з'єднує твою маленьку мережу з мережею провайдера. А пакет зі Львова до сервера у Франкфурті по дорозі проходить десяток-другий роутерів.

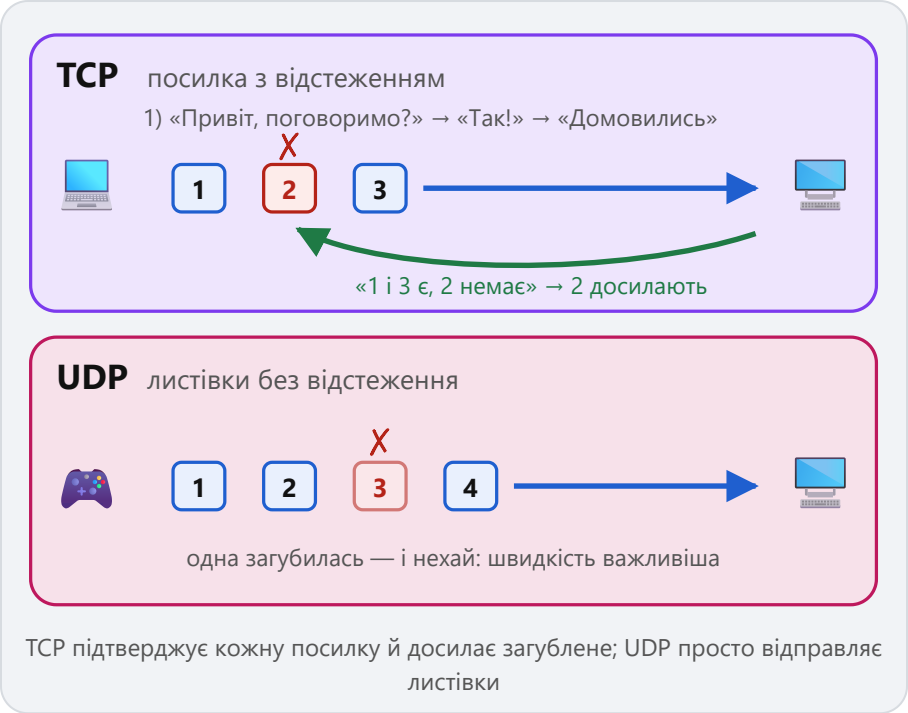
Щоб роутер знав, куди що передавати, у нього є **таблиця маршрутів** — «дорожні вказівники»: «пакети для 10.0.0.x — ліворуч, для всього іншого — на головну трасу». Запам'ятай цей образ: в AWS ти власноруч налаштовуватимеш такі таблиці.

Протокол — домовленість, як розмовляти

Щоб пошта працювала, усі мають однаково писати адресу, пакувати й підтверджувати доставку. Такі домовленості в мережах називають **протоколами**. Ти вже знаєш кілька: **IP** — як адресувати й доставляти пакети; далі будуть **HTTP** (як браузер розмовляє із сайтом) і **DNS** (як шукати адресу за ім'ям). А зараз — два найважливіші способи доставки.

TCP і UDP: посилка з відстеженням і листівка

- **TCP** («ті-сі-пі») — **посилка з відстеженням і підтвердженням**. Перед відправкою сторони «тиснуть руки» й домовляються про з'єднання. Кожен пакет пронумеровано, одержувач підтверджує отримання, загублене надсилають повторно, порядок відновлюють. Надійно, але трохи повільніше. Так працюють сайти, пошта, передавання файлів, бази даних.
- **UDP** («ю-ді-пі») — **листівка без відстеження**. Просто відправили — і все: ні рукостискання, ні підтверджень. Швидко й легко, але частина листівок може загубитися. Це нормально для відеодзвінків, онлайн-ігор і трансляцій: краще на мить втратити кадр, ніж чекати, поки його перешлють.



TCP підтверджує кожну посилку й досилає загублене; UDP просто відправляє листівки

🎯 На іспиті це з'являється в неочікуваних місцях. Наприклад: «онлайн-гра на **UDP**, потрібні статичні IP-адреси для всього світу» — це підказка, що правильна відповідь — сервіс, який уміє працювати з UDP. Ми до таких сервісів дійдемо.

Швидкість: ширина дороги і довжина дороги

Люди кажуть «швидкий інтернет», маючи на увазі дві різні речі:

- **Пропускна здатність** (bandwidth, «бендвідс») — **ширина дороги**: скільки даних проходить за секунду. Вимірюється в Mbps чи Gbps. Широка дорога — фільм завантажується швидко.
- **Затримка** (latency, «лейтенсі») — **довжина дороги**: скільки часу пакет їде від тебе до сервера й назад. Вимірюється в мілісекундах (мс). Навіть на найширшій трасі зі Львова до Токіо їхати далеко: відповідь прийде приблизно за чверть секунди, хоч би який швидкий був інтернет.



Звідси одне з головних правил хмарного архітектора: **щоб відповідь приходила швидко, дані й сервери треба розміщувати ближче до людей**. Цілі сервіси AWS існують лише для цього.

Файрвол — охоронець на вході

Коли пакети можуть приходити звідусіль, потрібен хтось, хто вирішує, кого впускати. Це **файрвол**, або брандмауер (firewall, «файрвол» — «вогняна стіна»). Він перевіряє кожен пакет за правилами: з якої адреси, на який порт. «На 443 — пускати всіх. На 22 — лише з адреси офісу. Решту — не пускати».

В AWS таких охоронців два, і різницю між ними обов'язково запитають на іспиті. Про них — у частині про мережу.

📖 **Словничок**

Мережа / LAN («лан») — з'єднані пристрої / локальна мережа будинку чи офісу.

Інтернет — мережа мереж на весь світ.

Пакет — маленька порція даних з адресами відправника й одержувача.

IP-адреса («ай-пі») — адреса пристрою в мережі, напр. `192.168.1.25`.

Порт — номер «комірки» для конкретної програми: 443 — HTTPS, 22 — SSH.

Роутер («раутер») — передає пакети між мережами за таблицею маршрутів.

Комутатор (switch, «світч») — передає пакети всередині однієї мережі.

Протокол — домовленість, як саме обмінюватися даними.

TCP («ті-сі-пі») — надійна доставка з підтвердженням. **UDP** («ю-ді-пі») — швидка без підтвердження.

Пропускна здатність (bandwidth) — скільки даних за секунду, «ширина дороги».

Затримка (latency) — скільки чекати відповіді, «довжина дороги»; у мілісекундах.

Файрвол («файрвол») — охоронець, що пропускає пакети лише за правилами.

🧩 Міні-тест: перевір себе

? Що таке IP-адреса?

А Адреса пристрою в мережі, за якою йому доставляють пакети

Б Пароль від Wi-Fi

В Назва сайту, наприклад khmarynka.ua

Г Швидкість інтернету

Відповідь: А. IP-адреса — як адреса поштомата: без неї пакет нікуди не доставити.

? На одному сервері працюють і сайт, і база даних. Як пакет «знає», якій програмі він призначений?

А За кольором пакета

Б За розміром пакета

В За номером порту

Г Ніяк, усі пакети отримують усі програми

Відповідь: В. Порт — номер комірки поштомата. Сайт чекає на 443, база даних MySQL — на 3306.

? Відеодзвінок: краще на мить втратити кадр, ніж чекати, поки його перешлють. Який протокол доставки зазвичай обирають?

А TCP

Б UDP

В Жодного, відео не передається мережею

Г Лише пошту

Відповідь: Б. UDP — «листівка без відстеження»: швидко, без підтверджень і повторів. Для живого відео це саме те.

? Клієнти в Японії скаржаться, що сайт, розміщений у Європі, відповідає повільно, хоча інтернет у них дуже швидкий. Що найімовірніше винне?

А Замала пропускна здатність у клієнтів

Б Затримка через велику відстань до сервера

В Японські браузери

Г Замало місця на диску сервера

Відповідь: Б. Широка дорога не скорочує відстань. Щоб відповіді були швидкими, сервери чи копії даних розміщують ближче до людей.

? Який порт зазвичай використовують захищені сайти (HTTPS)?

А 22

Б 80

В 443

Г 3306

Відповідь: В. 443 — HTTPS, 80 — звичайний HTTP, 22 — SSH, 3306 — MySQL.

✅ **Головне**

Дані їдуть мережею **пакетами**, які проходять через **роутери** за таблицями маршрутів.

IP-адреса — адреса поштомата, **порт** — номер комірки: 443 HTTPS, 80 HTTP, 22 SSH, 3389 RDP.

TCP — надійно з підтвердженням; **UDP** — швидко без підтверджень (ігри, відео, дзвінки).

Пропускна здатність (ширина дороги) і **затримка** (довжина дороги) — різні речі; затримку зменшує лише близькість.

Файрвол пропускає пакети за правилами «звідки і на який порт».

🌐 Інтернет і вебсайти: що відбувається, коли відкриваєш сторінку

👴 Простими словами

Коли ти відкриваєш сайт, браузер спершу питає «телефонну книгу» (DNS), де живе цей сайт, потім іде за адресою, просить сторінку — і сервер відповідає. Усе це за частку секунди.

🎯 Що ти дізнаєшся

- хто такі клієнт і сервер
- з чого складається адреса сайту — URL
- як DNS перетворює ім'я сайту на IP-адресу
- як браузер і сервер розмовляють по HTTP і що означають коди 200, 404, 503
- що таке HTTPS, cookies, API і JSON

🥐 Тарас набирає в браузері **khmarynka.ua** — і за секунду відкривається тестова сторінка пекарні з фото круасанів. «Що сталося за цю секунду?» — питає Оля. «Ціла подорож, — каже Тарас. — Зараз покажу по кроках».

Клієнт і сервер

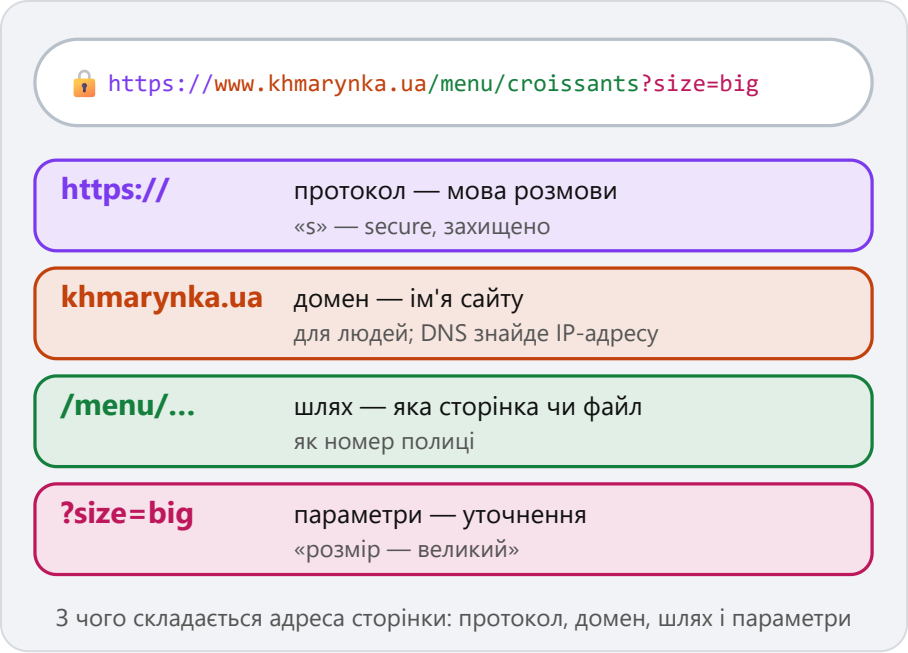
Будь-який сайт — це розмова двох сторін:

- **Клієнт** — той, хто питає. Твій браузер (Chrome, Safari, Firefox) чи мобільний застосунок.
- **Сервер** — той, хто відповідає. Комп'ютер десь у дата-центрі, на якому працює програма сайту.

Як у кав'ярні: гість (клієнт) робить замовлення, кухня (сервер) готує й віддає. Гостей багато, кухня одна — і вона має встигати всім. Запам'ятай слово **запит** (request, «ріквест») — це «замовлення» клієнта, і **відповідь** (response, «респонс»).

Адреса сайту — URL

Адреса, яку ти бачиш угорі браузера, називається **URL** (читається «ю-ар-ел»). Розберемо її по частинах:



З чого складається адреса сторінки: протокол, домен, шлях і параметри

- `https://` — **протокол**: мова розмови. Літера «s» означає secure — захищено.
- `www.khmarynka.ua` — **доменне ім'я**: назва «кав'ярні», зручна для людей.
- `/menu/croissants` — **шлях**: яку саме сторінку чи файл дати, ніби номер полиці.
- `?size=big` — **параметри**: уточнення до замовлення.

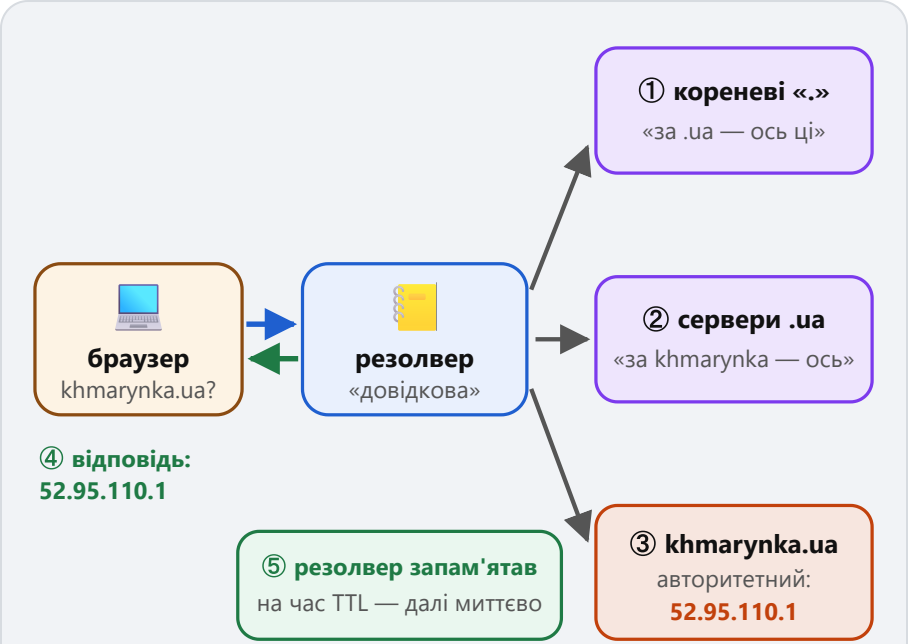
DNS — телефонна довідка інтернету

Комп'ютери знаходять одне одного за IP-адресами, а люди запам'ятовують імена. Хтось має перекласти `khmarynka.ua` у `52.95.110.1`. Це робить **DNS** (читається

«ді-ен-ес», Domain Name System — система доменних імен). Уяви телефонну довідку: називаєш ім'я — отримуєш номер.

Як це відбувається, коли адреси ще ніхто не знає:

1. Браузер питає **резолвер** — «довідкову» твого провайдера.
2. Резолвер питає **кореневі сервери**: «Хто відповідає за `.ua`?»
3. Потім сервери зони `.ua`: «Хто відповідає за `khmarynka.ua`?»
4. І нарешті **авторитетний сервер** домену (його налаштовує власник сайту): «Яка адреса в `khmarynka.ua`?» — «`52.95.110.1`».
5. Резолвер повертає адресу браузеру й **запам'ятовує** її на певний час — **TTL** («ті-ті-ел», time to live). Наступного разу відповідь миттєва.



Як DNS знаходить адресу за ім'ям — і запам'ятовує відповідь на час TTL

Дві речі звідси знадобляться в AWS:

- Щоб мати ім'я сайту, домен **реєструють** (купують на рік чи кілька) у реєстратора. А записи «ім'я → адреса» тримають у DNS-сервісі. В AWS і те, і те вміє **Route 53**.
- Через TTL зміни в DNS розходяться світом **не миттєво**: резолвери ще якийсь час пам'ятають стару адресу. На іспиті це пояснює, чому перемикання «через DNS» не завжди найшвидше.

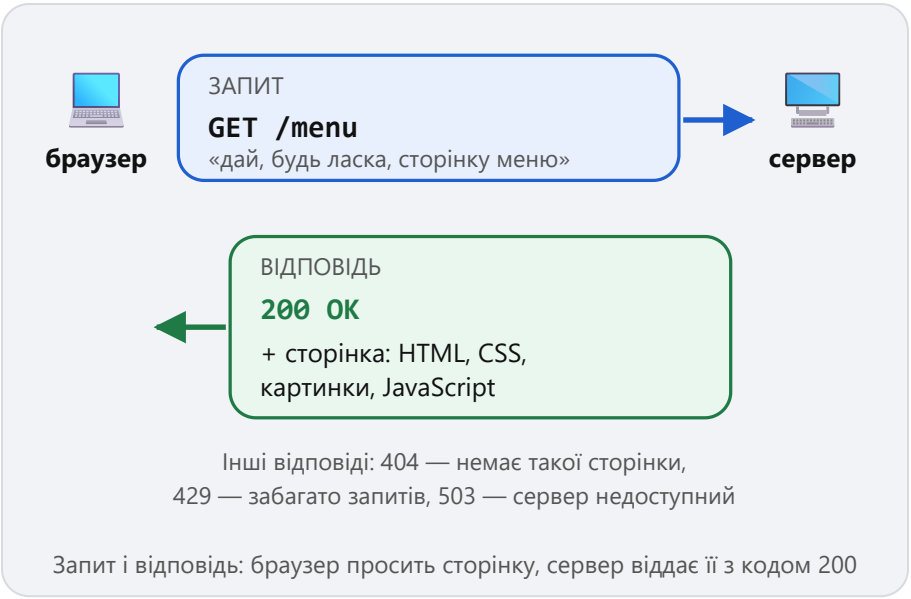
HTTP — як браузер і сервер розмовляють

Коли адресу знайдено, браузер відправляє серверу запит мовою **HTTP** («ейч-ті-ті-пі», HyperText Transfer Protocol). Запит складається з **методу** й шляху:

- `GET /menu` — «дай мені сторінку меню» (отримати);
- `POST /orders` — «ось моє замовлення, прийми» (надіслати дані, наприклад форму);
- ще бувають `PUT` (замінити), `DELETE` (видалити) та інші.

Сервер відповідає **кодом стану** і самим вмістом (сторінкою, картинкою). Коди варто знати за першою цифрою:

Код	Що означає	Приклад
2xx	усе добре	**200 OK** — ось твоя сторінка
3xx	іди за іншою адресою	**301** — сторінка переїхала
4xx	помилка з боку клієнта	**403** — заборонено, **404** — не знайдено, **429** — забагато запитів
5xx	помилка з боку сервера	**500** — збій програми, **502/504** — не дочекалися відповіді, **503** — сервіс недоступний



💡 Коди стану ще повернуться: сервіси AWS відповідають **429**, коли клієнт перевищує ліміт запитів, а балансувальник вважає сервер хворим, коли той відповідає 5xx чи мовчить.

HTTPS і замочок

Звичайний HTTP — як **листівка**: будь-хто на шляху (наприклад, у кав'ярняному Wi-Fi) може прочитати, що там написано, навіть пароль.

HTTPS — той самий HTTP, але в **замкненому броньованому конверті**. Дані шифруються (технологія TLS), тож по дорозі їх неможливо прочитати. А

сертифікат сайту — ніби паспорт — підтверджує, що це справді `khmarynka.ua`, а не підробка. Замочок біля адреси в браузері означає: конверт замкнено, паспорт перевірено.

Як працює шифрування і де брати сертифікати в AWS — в окремому розділі про шифрування.

З чого складається сторінка

Сторінка — це не один файл, а десятки:

- **HTML** — зміст і структура: заголовки, текст, кнопки;
- **CSS** — оформлення: кольори, шрифти, розташування;
- **JavaScript** — поведінка: що станеться, коли натиснеш кнопку;
- картинки, шрифти, відео.

Частина з них **статична** — однакова для всіх: логотип, фото круасанів, стилі. Їх можна просто зберігати як файли й роздавати дуже дешево й швидко — пізніше ти дізнаєшся, що в AWS для цього поєднують сховище S3 і мережу доставки CloudFront.

Інша частина **динамічна** — своя для кожного: «твій кошик», «твої замовлення». Її створює програма на сервері, беручи дані з бази даних.

Cookies і сесії: як сайт тебе пам'ятає

HTTP сам нічого не пам'ятає: кожен запит для сервера — як від нової людини. Як же сайт знає, що ти вже в особистому кабінеті і що в кошику три круасани?

Сервер видає браузеру **cookie** («кукі») — маленький талончик з номером, як **номерок у гардеробі**. Браузер показує цей номерок у кожному наступному запиті, а сервер за номером знаходить твою **сесію**: хто ти й що в кошику.

Питання «де сервер тримає сесії» стане дуже важливим, коли серверів буде багато: якщо сесія лежить в пам'яті одного сервера, а наступний запит потрапить на інший, кошик «зникне». Розв'язок — у розділі про застосунки.

API і JSON: меню й бланк замовлення для програм

Не лише люди спілкуються із сервером. Мобільний застосунок «Хмаринки» теж питає сервер — але не сторінки, а **дані**. Для цього сервер має **API** (читається «ей-пі-ай», Application Programming Interface) — перелік дозволених запитів і їхній формат. Образ: **меню з правилами замовлення** для програм.

Наприклад, застосунок надсилає `GET /api/menu`, а сервер відповідає даними у форматі **JSON** («джейсон»):

```
{
  "name": "Круасан з мигдалем",
  "price": 45,
  "inStock": true,
  "tags": ["випічка", "хіт"]
}
```

JSON — простий текст: назви полів у лапках, двокрапка, значення. Його легко читати і людям, і програмам. Зверни на нього увагу вже зараз: в AWS у JSON записують **правила доступу** (хто що може робити), налаштування й події. Ти читатимеш JSON постійно.

Де тут хмара

Усе, що відбувається «з того боку» — сервер із програмою, база даних, DNS, сертифікат, сховище для фото, — можна не купувати, а **орендувати в AWS**. Саме цим і займеться Оля. Але спершу ще кілька основ.

📖 **Словничок**

Клієнт / сервер — той, хто питає / той, хто відповідає.

Запит / відповідь (request / response) — «замовлення» клієнта і «страва» від сервера.

URL («ю-ар-ел») — повна адреса сторінки: протокол, домен, шлях, параметри.

Домен — ім'я сайту, напр. `khmarynka.ua`; його реєструють на певний строк.

DNS («ді-ен-ес») — «довідкова»: перетворює ім'я на IP-адресу.

TTL («ті-ті-ел») — скільки часу відповідь DNS можна пам'ятати.

HTTP / HTTPS — мова браузера й сервера / те саме, але в зашифрованому конверті.

Метод — тип запиту: `GET` отримати, `POST` надіслати.

Код стану — трицифрова відповідь сервера: 200 добре, 404 не знайдено, 503 недоступно.

Сертифікат — «паспорт» сайту, що підтверджує справжність.

Статичний / динамічний вміст — однаковий для всіх / створений під конкретного користувача.

Cookie («кукі») / **сесія** — номерок у гардеробі / записи сервера про тебе.

API («ей-пі-ай») — «меню» запитів для програм. **JSON** («джейсон») — текстовий формат даних.

🧩 Міні-тест: перевір себе

? Що робить DNS?

А Шифрує сторінку

Б Перетворює ім'я сайту на IP-адресу

В Зберігає фото з сайту

Г Прискорює процесор

Відповідь: Б. DNS — телефонна довідка інтернету: називаєш ім'я — отримуєш адресу.

? Сервер відповів кодом 404. Що це означає?

А Усе гаразд

Б Сервер зламався

В Сторінку не знайдено

Г Забагато запитів

Відповідь: В. 4xx — помилки з боку клієнта. 404 — такої сторінки немає.

? Сайт відповідає кодом 503. Де проблема?

А У браузері користувача

Б На боці сервера: він тимчасово недоступний або перевантажений

В У клавіатурі

Г Проблеми немає

Відповідь: Б. Коди 5xx — проблеми сервера. 503 — «сервіс зараз недоступний».

? Чим HTTPS відрізняється від HTTP?

А Нічим, це те саме

Б Дані шифруються дорогою, а сертифікат підтверджує, що сайт справжній

В HTTPS працює лише на телефонах

Г HTTPS швидше завантажує відео

Відповідь: Б. HTTPS — HTTP у замкненому конверті з перевіреним «паспортом» сайту.

? Навіщо сайту cookies?

А Щоб пекти печиво

Б Щоб прискорити інтернет

В Щоб «пам'ятати» користувача між запитами: вхід, кошик

Г Щоб блокувати рекламу

Відповідь: В. HTTP сам нічого не пам'ятає. Cookie — номерок у гардеробі, за яким сервер знаходить твою сесію.

? Що таке JSON?

А Мова програмування для ігор

Б Простий текстовий формат для обміну даними між програмами

В Назва браузера

Г Вид сертифіката

Відповідь: Б. JSON — поля в лапках і значення через двокрапку. В AWS у ньому пишуть правила доступу, налаштування й події.

✅ **Головне**

Сайт — розмова **клієнта** (браузер, застосунок) і **сервера**: запит → відповідь.

URL = протокол + домен + шлях + параметри; **DNS** перекладає домен в IP і запам'ятовує відповідь на **TTL**.

HTTP: `GET` отримати, `POST` надіслати; коди **2xx** добре, **4xx** помилка клієнта (404, 429), **5xx** помилка сервера (500, 503).

HTTPS — шифрування + сертифікат-«паспорт».

Статичне можна роздавати з простого сховища; **динамічне** створює програма з базою даних. **Cookie** — номерок до твоєї сесії. **API** + **JSON** — як програми розмовляють між собою.

🔢 IP-адреси, підмережі й CIDR без страху

👴 Простими словами

IP-адреса — це номер будинку в мережі. Підмережа — вулиця, на якій стоять кілька будинків. Запис CIDR (наприклад /24) просто каже, скільки будинків поміщається на вулиці.

🎯 Що ти дізнаєшся

- як влаштована IP-адреса і чому числа в ній лише від 0 до 255
- чим приватні адреси відрізняються від публічних
- як один роутер ховає цілу родину за однією адресою — NAT
- як читати записи на кшталт 10.0.0.0/16 і швидко рахувати, скільки в них адрес
- чому мережі з однаковими адресами не можна з'єднати

🥐 Для сайту «Хмаринки» Тарас хоче збудувати в AWS власну мережу. У налаштуваннях його просять ввести «діапазон адрес», і він пише `10.0.0.0/16`. Оля дивиться на це, як на шифровку: «Що це за дріб? Навіщо слеш?» — «Це просто спосіб записати цілий район адрес одним рядком, — каже Тарас. — Зараз розкладемо».

💡 Цей розділ — найбільш «математичний» у курсі. Не лякайся: знадобляться лише додавання і степені двійки, а таблиця з готовими відповідями буде під рукою. Після нього мережеві розділи читатимуться набагато легше.

Будова IP-адреси

Ти вже знаєш, що IP-адреса — це чотири числа через крапку: `192.168.1.25`. Кожне з чотирьох чисел займає рівно **один байт** — вісім бітів. А вісьмома бітами (нулями й одиницями) можна записати лише 256 різних значень: від 0 до 255. Тому в IP-адресі не буває чисел на кшталт 300.

Усього в адресі 4 × 8 = **32 біти**. Це дає приблизно 4,3 мільярда різних адрес. Здається, багато — але пристроїв

у світі давно більше. Тому придумали дві хитрості: **приватні адреси** з **NAT** і нову, величезну версію адрес — **IPv6**. Звичні адреси з чотирма числами називаються **IPv4**.



IPv4-адреса: чотири числа, кожне — один байт (вісім бітів), тому від 0 до 255

Публічні й приватні адреси

- **Публічна адреса** унікальна на весь інтернет — як повна поштова адреса з країною, містом і вулицею. Саме за нею тебе знаходять будь-звідки.
- **Приватна адреса** діє лише всередині однієї мережі — як **внутрішній номер телефону в офісі** («набери 214»). Такі самі номери одночасно використовують мільйони інших офісів і квартир, і нікому це не заважає.

Для приватних мереж домовилися використовувати три діапазони:

Діапазон	Від — до
`10.x.x.x`	10.0.0.0 — 10.255.255.255
`172.16–31.x.x`	172.16.0.0 — 172.31.255.255
`192.168.x.x`	192.168.0.0 — 192.168.255.255

Домашні роутери зазвичай роздають адреси `192.168.x.x`, а у своїх мережах в AWS найчастіше беруть `10.x.x.x`.

NAT — одна адреса на всю родину

Провайдер дає твоєму домашньому роутеру **одну** публічну адресу. А вдома — телефон, ноутбук, телевізор, усі з приватними адресами `192.168.x.x`. Як вони всі ходять в інтернет?

Роутер працює як **ресепшн в офісі**. Коли телефон надсилає запит на сайт, роутер замінює на пакеті зворотну адресу на свою публічну й записує в журнал: «відповідь від цього сайту — для телефона». Відповідь приходить на ресепшн, і роутер передає її потрібному пристрою. Ця підміна адрес називається **NAT** («нат», Network Address Translation — «переклад мережевих адрес»).



Корисний побічний ефект: ззовні **не можна** напряму звернутися до твого телефона — ресепшн не пропустить лист, якого ніхто не чекав. Вийти назовні можна, зайти ззовні — ні. Запам'ятай цю «односторонність»: в AWS є сервіс **NAT Gateway**, який робить рівно те саме для серверів, що мають оновлюватися з інтернету, але мусять лишатися недосяжними ззовні.

CIDR: як записати цілий район адрес

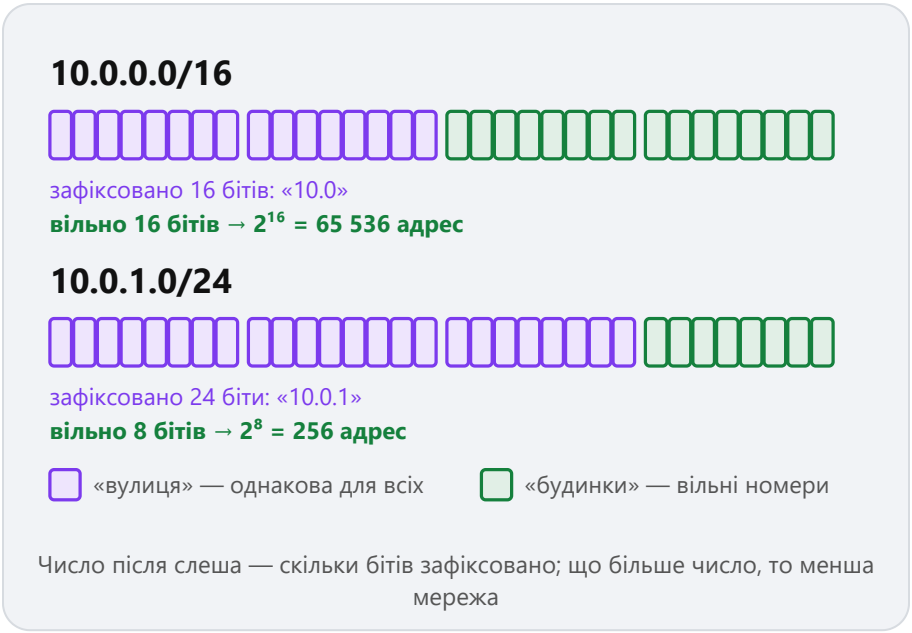
Мережі рідко складаються з однієї адреси. Зазвичай це цілий **діапазон** — «район» адрес. Записувати його з

першої до останньої незручно, тому користуються записом **CIDR** (читається «сайдер»):

`10.0.0.0/16`

Число після слеша означає: **скільки бітів ліворуч зафіксовано**. Зафіксована частина — ніби назва вулиці, однакова для всіх будинків району. Решта бітів — вільні: це номери будинків усередині.

- `/16` — зафіксовано перші **16** бітів, тобто перші **два** числа: `10.0.x.x`. Вільних бітів — 32 − 16 = 16, отже адрес 2^{16} = **65 536**.
- `/24` — зафіксовано перші **три** числа: `10.0.1.x`. Вільних бітів 8 → **256** адрес.
- `/32` — зафіксовано все: рівно **одна** адреса.



Формула одна: **кількість адрес = 2 у степені (32 – число після слеша)**. А найпростіше — запам'ятати кілька рядків:

Запис	Адрес	Образ
`/16`	65 536	ціле місто
`/20`	4 096	великий район
`/24`	256	квартал
`/26`	64	вулиця
`/28`	16	один будинок
`/32`	1	одна квартира

⚠️ Головна пастка: **що більше число після слеша, то менша мережа**. `/28` — крихітна (16 адрес), `/16` — величезна (65 536).

Два особливі записи варто знати напам'ять:

- `0.0.0.0/0` — зафіксовано **нуль** бітів, тобто **будь-яка адреса**, «увесь інтернет». У таблицях маршрутів рядок `0.0.0.0/0 → ворота в інтернет` означає: «усе, що не належить до наших районів, — на головну трасу».
- `x.x.x.x/32` — одна конкретна адреса. У правилах захисту так пишуть: «пускати лише адресу офісу `203.0.113.10/32`».

Підмережі: ділимо місто на квартали

Великий діапазон зазвичай ділять на менші — **підмережі** (subnets, «сабнетс»), як місто на квартали. З `10.0.0.0/16` можна зробити, наприклад:

- `10.0.1.0/24` — квартал для того, що має бути видно з інтернету;
- `10.0.2.0/24` — квартал для серверів із програмою;
- `10.0.3.0/24` — квартал для баз даних.

Кожен такий квартал — 256 адрес, а всього їх у `/16` поміститься 256 штук.



Мережа 10.0.0.0/16 поділена на квартали-підмережі /24

Саме так ти будуватимеш мережу в AWS. Дві деталі звідти, які люблять на іспиті:

- розмір мережі в AWS — від `/16` (найбільша) до `/28` (найменша);
- у **кожній** підмережі AWS забирає собі **5 адрес** (для мережі, роутера, DNS, запасу й «широкомовлення»), тож у `/24` вільних лишиться 251.

Чому адреси не можуть перетинатися

Уяви два міста з однаковими адресами: і там, і там є «вулиця Шевченка, 5». Якщо з'єднати їх однією поштою, листоноша не знатиме, в яке місто нести лист.

З мережами так само. Якщо мережа офісу — `10.0.0.0/16`, а мережа в AWS — теж `10.0.0.0/16`, з'єднати їх напряму **неможливо**: пакет для `10.0.1.5` не знатиме, куди йти. Тому досвідчені архітектори одразу дають мережам **різні** діапазони: офіс — `192.168.0.0/16`, мережа A — `10.0.0.0/16`, мережа B — `10.1.0.0/16`. Навіть якщо з'єднувати їх поки не планують.

> 🎯 Типове питання: «Дві мережі не вдається з'єднати напряму. Чому?» — бо їхні діапазони CIDR **перетинаються**.

IPv6 коротко

IPv6 — нова версія адрес: 128 бітів замість 32. Виглядає так: `2001:db8:1f70::5c`. Адрес стільки, що кожен пристрій на планеті може мати власну публічну — NAT уже не потрібен.

А як тоді дозволити пристроям виходити назовні, але не пускати гостей усередину? В AWS для IPv6 є окремі «ворота лише на вихід» — **Egress-only Internet Gateway**.

Поки просто запам'ятай: для IPv4 цю роль грає NAT, для IPv6 — egress-only ворота.

🔍 **Для допитливих: Як швидко рахувати CIDR у голові**

Кожен зайвий біт праворуч від слеша ділить мережу навпіл: `/24` — 256 адрес, `/25` — 128, `/26` — 64, `/27` — 32, `/28` — 16. І навпаки, мінус один біт — удвічі більше: `/23` — 512, `/22` — 1024.

Корисно пам'ятати «круглі» межі: `/8`, `/16`, `/24` — це рівно одне, два або три зафіксовані числа адреси.
`10.0.0.0/8` — усі адреси, що починаються з 10;
`10.20.0.0/16` — усі, що починаються з 10.20;
`10.20.30.0/24` — усі, що починаються з 10.20.30.

📖 **Словничок**

IPv4 — звичні адреси з чотирьох чисел від 0 до 255 (32 біти).

IPv6 — нові 128-бітні адреси, наприклад `2001:db8::5c`.

Публічна адреса — унікальна на весь інтернет.

Приватна адреса — діє лише всередині мережі: `10.x`, `172.16-31.x`, `192.168.x`.

NAT («нат») — підміна адрес: багато пристроїв виходять назовні з однієї публічної адреси, ззовні до них не зайти.

CIDR («сайдер») — запис діапазону: `10.0.0.0/16`; адрес = 2^(32 − число після слеша).

Підмережа (subnet, «сабнет») — менший діапазон усередині мережі, «квартал».

0.0.0.0/0 — будь-яка адреса, «увесь інтернет».

/32 — рівно одна адреса.

Перетин CIDR — коли діапазони двох мереж спільні; такі мережі напряму не з'єднати.

🧩 Міні-тест: перевір себе

? Яке число НЕ може бути частиною IPv4-адреси?

- **А** 0
- **Б** 127
- **В** 255
- **Г** 300

Відповідь: Г. Кожне число — один байт, тобто від 0 до 255. Числа 300 бути не може.

? Яка з цих адрес приватна?

- **А** 8.8.8.8
- **Б** 10.0.5.20
- **В** 52.95.110.1
- **Г** 203.0.113.10

Відповідь: Б. Приватні діапазони: 10.x.x.x, 172.16–31.x.x і 192.168.x.x. Адреса 10.0.5.20 — приватна.

? Скільки адрес у мережі /24?

А 24

Б 256

В 65 536

Г 16

Відповідь: Б. Вільних бітів 32 − 24 = 8, а 2^8 = 256.

? Яка мережа більша?

А 10.0.0.0/16

Б 10.0.0.0/24

В 10.0.0.0/28

Г Вони однакові

Відповідь: А. Що менше число після слеша, то більше вільних бітів і адрес. /16 — 65 536 адрес, /24 — 256, /28 — 16.

? Що означає 0.0.0.0/0 у таблиці маршрутів?

А Жодної адреси

Б Лише адресу роутера

В Будь-яку адресу — «увесь інтернет»

Г Помилку в налаштуваннях

Відповідь: В. Зафіксовано нуль бітів — отже, підходить будь-яка адреса. «Усе інше — на головну трасу».

? Мережу офісу 10.0.0.0/16 не вдається напряму з'єднати з мережею в хмарі 10.0.0.0/16. Чому?

А Бо в них різні паролі

Б Їхні діапазони адрес перетинаються, і пакети не знатимуть, у яку мережу йти

В Бо хмара не підтримує офісні мережі

Г Бо /16 — занадто мало адрес

Відповідь: Б. Дві мережі з однаковими адресами — як два міста з однаковими вулицями. Діапазони треба планувати різними з самого початку.

✅ **Головне**

IPv4 — чотири числа від 0 до 255 (32 біти); IPv6 — 128 бітів, адрес вистачить усім.

Приватні адреси (`10.x` , `172.16–31.x` , `192.168.x`) — «внутрішні номери»; **публічні** — унікальні на весь інтернет.

NAT випускає багато пристроїв назовні з однієї адреси й не пускає непрохану пошту всередину.

CIDR: адрес = 2^(32 − N); `/16` = 65 536, `/24` = 256, `/28` = 16; що більше N, то менша мережа. `0.0.0.0/0` — увесь інтернет.

Мережу ділять на **підмережі**; діапазони мереж, які можуть з'єднуватися, **не повинні перетинатися**.

🏗️ Як влаштований сучасний застосунок

👴 Простими словами

Сучасний застосунок — як ресторан: зала (те, що бачить клієнт), кухня (логіка) і комора (база даних). Частини розмовляють між собою, і кожну можна збільшити окремо.

🎯 Що ти дізнаєшся

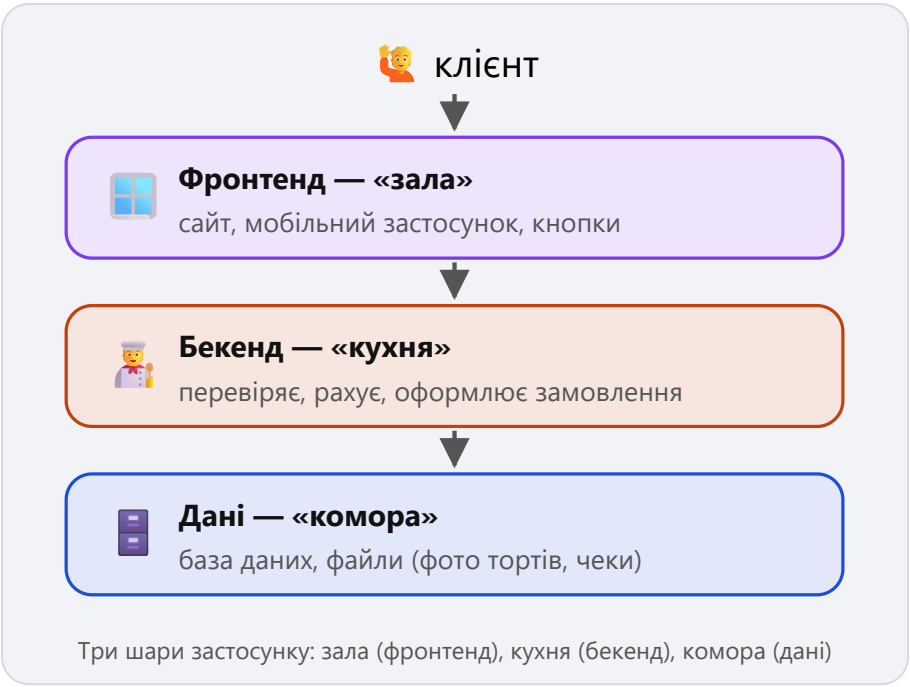
- з яких трьох шарів складається майже будь-який застосунок
- чим «моноліт» відрізняється від мікросервісів
- що означають stateful і stateless — і чому в хмарі це вирішальне
- навіщо частинам системи черги й події замість прямих «дзвінків»
- що таке масштабування, логи і єдина точка відмови

🥐 Тарас малює на серветці план майбутнього сайту «Хмаринки»: «Ось що бачить клієнт. Ось де рахується замовлення. А ось де все записується». Оля уважно дивиться: «Схоже на мою пекарню: зала, кухня і комора». — «Саме так! — радіє Тарас. — Більшість застосунків світу влаштовані як ресторан».

Три шари: зала, кухня, комора

Майже будь-який сучасний застосунок — інтернет-магазин, банк, соцмережа — складається з трьох шарів:

1. **Презентація, або фронтенд** (frontend, «фронтенд») — **зала ресторану**. Те, що бачить і чого торкається клієнт: сайт у браузері, мобільний застосунок, кнопки, фото, форма замовлення.
2. **Логіка, або бекенд** (backend, «бекенд») — **кухня**. Програма на сервері, яка приймає замовлення, перевіряє, чи є товар, рахує ціну, оформлює доставку. Клієнт її не бачить, але вона робить усю справжню роботу.
3. **Дані — комора й журнал замовлень**. База даних, де зберігаються клієнти, товари, замовлення, і сховище для файлів (фото тортів, чеки).



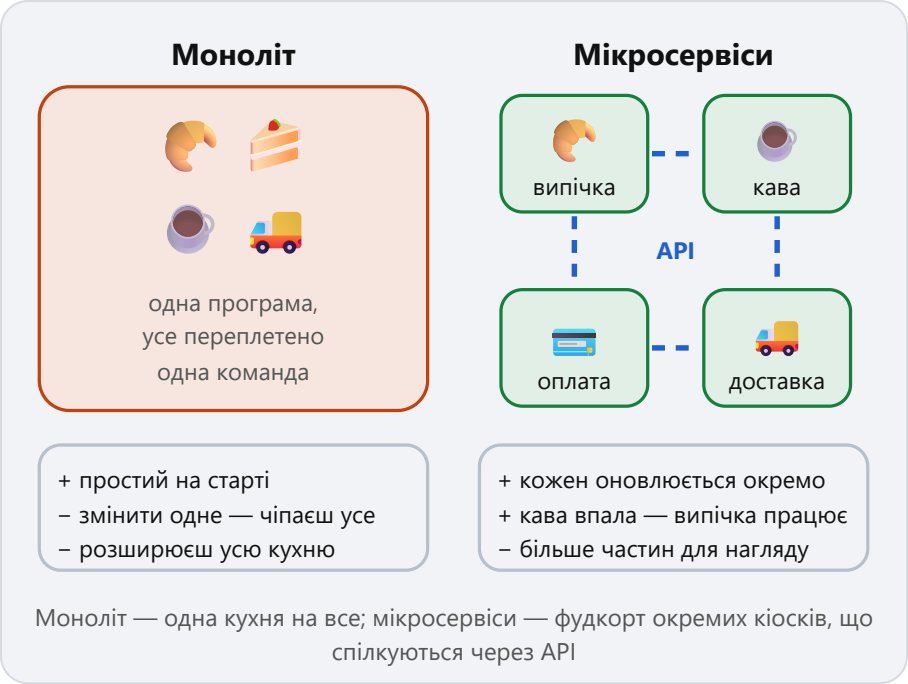
Три шари застосунку: зала (фронтенд), кухня (бекенд), комора (дані)

Цей малюнок — «трирівнева архітектура» (three-tier). Запам'ятай його: в AWS ти будуватимеш його знову і знову, лише щоразу надійніше й масштабніше. На іспиті він виглядає так: **балансувальник → сервери з програмою → база даних**.

Моноліт і мікросервіси

Кухню можна організувати двома способами.

- **Моноліт** — одна велика кухня, де ті самі кухарі печуть торти, варять каву й пакують доставку. Програма одна, велика, усе в ній переплетено. На початку це просто. Але щоб змінити рецепт тістечка, доводиться «зупиняти» всю кухню. А коли кава стала популярною, розширювати доводиться кухню цілком, хоча бракує лише кавоварок.
- **Мікросервіси** — **фудкорт**: окремі маленькі кіоски — випічка, кава, доставка, оплата. Кожен має свою команду, оновлюється й розширюється окремо, а між собою кіоски спілкуються через API. Зламалася кавоварка — випічку однаково продають. Ціна за гнучкість: більше «рухомих частин», за якими треба стежити.



Моноліт — одна кухня на все; мікросервіси — фудкорт окремих кіосків, що спілкуються через API

На іспиті мікросервіси часто йдуть поруч зі словами **decouple** («розв'язати» частини), контейнери і serverless — про них у частині про обчислення.

Stateful і stateless: офіціант, що пам'ятає, і дошка замовлень

Це одна з найважливіших ідей усього курсу.

- **Stateful** («стейтфул», зі станом) — сервер **пам'ятає** щось про клієнта між запитами. Як офіціант, що тримає твоє замовлення в голові: зручно, але до цього офіціанта доведеться звертатися щоразу, а якщо він піде додому — замовлення зникне.

- **Stateless** («стейтлес», без стану) — сервер **нічого не пам'ятає** між запитами. Усе, що треба пам'ятати (кошик, вхід), лежить у спільному місці — на **дошці замовлень** на кухні (база даних чи швидке сховище в пам'яті). Клієнт приносить лише номерок (cookie), і **будь-який** офіціант знайде за ним замовлення на дошці.



Stateful: замовлення в голові одного офіціанта. Stateless: усі записи на спільній дошці — обслужить будь-хто

Чому це вирішальне в хмарі? Бо **stateless-сервери можна вільно додавати й прибирати**. Прийшов натовп — поставили ще десять серверів, натовп пішов —

прибрали. Один сервер зламався — нічого не загубилося: наступний запит обслужить інший. З stateful-серверами так не вийде: разом із сервером зникають і кошики клієнтів.

> 🎯 На іспиті: «користувачі втрачають кошик або вхід, коли запит потрапляє на інший сервер» → зберігати сесії **поза сервером**, у спільному швидкому сховищі. Назви сервісів дізнаєшся пізніше.

Синхронно чи асинхронно

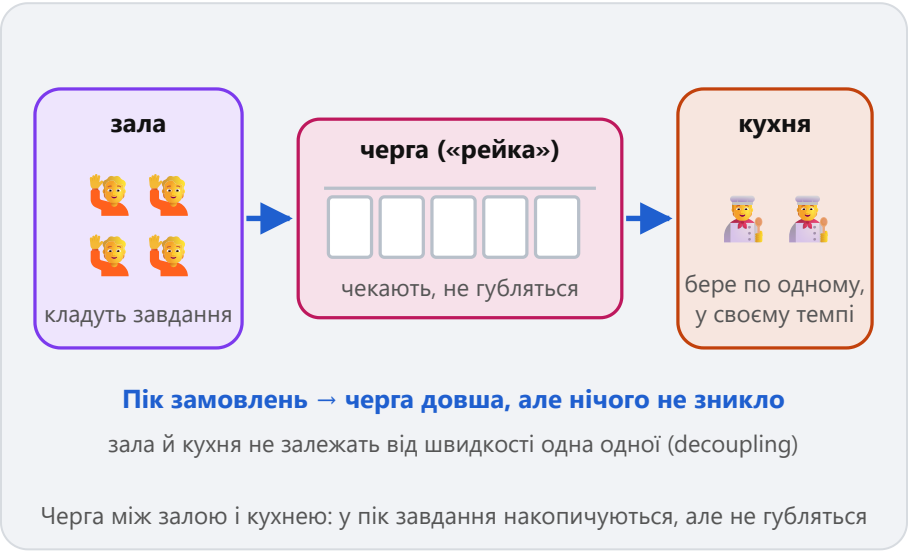
- **Синхронно** — як **телефонний дзвінок**: ти питаєш і чекаєш на лінії, поки не відповідять. Так перевіряють оплату: не можна підтвердити замовлення, поки банк не сказав «так».
- **Асинхронно** — як **повідомлення в месенджері**: написав і займаєшся своїми справами, відповідь прийде пізніше. Так надсилають лист-підтвердження, генерують PDF-чек, оновлюють складський облік.

Що більше роботи можна зробити асинхронно, то швидше система відповідає клієнтові й то менше повільна частина гальмує решту.

Черги й події: як частини системи не заважають одна одній

Уяви кухню в пікову годину. Офіціанти чіпляють папірці із замовленнями на **рейку**, а кухарі знімають їх по одному у своєму темпі. Замовлень стало вдвічі більше — папірців на рейці побільшало, але жоден не загубився, і ніхто не кричить на кухарів.

Ця рейка — **черга** (queue, «к'ю»). Одна частина системи кладе в неї завдання, інша забирає, коли готова. Вони не залежать від швидкості одна одної: це називається **розв'язування** (decoupling, «декаплінг»).



А **подія** (event, «івент») — це оголошення «щось сталося»: «нове замовлення №512!». Хто зацікавлений,

той реагує: кухня починає пекти, бухгалтерія записує, клієнтові йде SMS. Той, хто оголосив, навіть не знає, скільки слухачів у нього є — і це добре: нового слухача можна додати, нічого не змінюючи.

Черги й події — **величезна тема іспиту**. Слова «decouple», «buffer», «spikes» (сплески) у питанні майже завжди ведуть до черги.

Масштабування: більша піч чи більше печей

Клієнтів побільшало. Є два шляхи:

- **Вертикальне масштабування** (scale up, «скейл ап») — купити **більшу піч**: потужніший сервер. Просто, але є межа (найбільша піч колись закінчиться), і на час заміни піч доведеться вимкнути.
- **Горизонтальне масштабування** (scale out, «скейл аут») — поставити **більше печей**: більше серверів, і адміністратора, який розподіляє замовлення між ними (це **балансувальник навантаження**). Межі практично немає, і одна зламана піч не зупиняє пекарню.

Хмара обожнює горизонтальне масштабування — і воно працює лише зі **stateless**-серверами. Бачиш, як усе пов'язано?

Єдина точка відмови

Якщо вся пекарня тримається на одній печі, то її поломка зупиняє все. Така деталь називається **єдиною точкою відмови** (single point of failure, «сингл пойнт оф фейлюр»).

Головна звичка архітектора — шукати такі точки й **дублювати** їх: дві печі в різних приміщеннях, дві копії бази даних, два входи в будівлю. Про це — ціла частина курсу про надійність.

Логи й метрики: щоденник кухні

Як дізнатися, що щось пішло не так? Досвідчені кухні ведуть **журнал**:

- **Логи** (logs, «логз») — записи подій з часом: «12:03 прийнято замовлення №512», «12:04 ПОМИЛКА: оплата не пройшла». Без логів, коли щось ламається, ти сліпий.
- **Метрики** (metrics) — числа в часі: скільки замовлень за хвилину, наскільки завантажений процесор, скільки помилок. На метрики ставлять **тривоги**: «якщо помилок більше 10 за хвилину — надіслати SMS».

В AWS цим займається окремий сервіс, схожий на приладову панель автомобіля. Ми до нього дійдемо.

📖 Словничок

Фронтенд / бекенд — «зала» (що бачить клієнт) / «кухня» (логіка на сервері).

Трирівнева архітектура (three-tier) — фронтенд + бекенд + дані.

Моноліт — одна велика програма на все.

Мікросервіси — багато маленьких незалежних сервісів, що спілкуються через API.

Stateful / stateless («стейтфул / стейтлес») — сервер пам'ятає клієнта / не пам'ятає, стан лежить окремо.

Сесія — дані про користувача між запитами: вхід, кошик.

Синхронно / асинхронно — чекати відповіді на лінії / надіслати й займатися своїм.

Черга (queue, «к'ю») — «рейка з замовленнями» між частинами системи.

Подія (event, «івент») — оголошення «щось сталося», на яке реагують зацікавлені.

Розв'язування (decoupling, «декаплінг») — частини системи не залежать від швидкості й збоїв одна одної.

Вертикальне / горизонтальне масштабування — більший сервер / більше серверів.

Балансувальник навантаження — «адміністратор», що розподіляє запити між серверами.

Єдина точка відмови — деталь, поломка якої зупиняє все.

Логи / метрики — записи подій / числа в часі.

🧩 **Міні-тест: перевір себе**

? З яких трьох шарів складається типовий застосунок?

А Екран, клавіатура, миша

Б Фронтенд (що бачить клієнт), бекенд (логіка), дані (база й файли)

В Процесор, пам'ять, диск

Г Wi-Fi, роутер, інтернет

Відповідь: Б. Зала, кухня, комора: фронтенд, бекенд і дані. Процесор, пам'ять і диск — це вже «залізо» всередині кожного комп'ютера.

? Чому в хмарі сервери бажано робити stateless?

А Бо так вони гарніше виглядають

Б Тоді будь-який сервер можна додати, прибрати або втратити без втрати даних користувачів

В Бо stateless-сервери не потребують електрики

Г Бо stateful заборонено законом

Відповідь: Б. Якщо сервер нічого не пам'ятає сам, а стан лежить у спільному місці, сервери стають взаємозамінними — їх легко масштабувати й замінювати.

? У пікову годину замовлення надходять швидше, ніж кухня встигає їх виконувати, і частина губиться. Що допоможе найкраще?

А Купити більший монітор

Б Поставити чергу між прийманням замовлень і кухнею

В Вимкнути сайт у пікові години

Г Приймати замовлення лише телефоном

Відповідь: Б. Черга — «рейка з папірцями»: у пік завдання накопичуються й чекають, а кухня бере їх у своєму темпі. Нічого не губиться.

? Надіслати клієнтові лист-підтвердження після замовлення краще...

- **А** синхронно: хай клієнт чекає, поки лист піде
- **Б** асинхронно: клієнт одразу бачить «Дякуємо!», а лист піде за мить окремо
- **В** ніяк: листи не потрібні
- **Г** лише вручну

Відповідь: Б. Клієнтові не треба чекати, поки поштова система відповість. Такі задачі роблять асинхронно — часто через чергу.

? Що таке горизонтальне масштабування?

А Поставити сервер горизонтально

Б Додати більше серверів і розподіляти між ними навантаження

В Замінити сервер на потужніший

Г Зменшити кількість користувачів

Відповідь: Б. Більше печей замість однієї більшої. Межі майже немає, а поломка однієї печі не зупиняє роботу.

✅ **Головне**

Застосунок = **фронтенд** (зала) + **бекенд** (кухня) + **дані** (комора). На іспиті: балансувальник → сервери → база.

Моноліт — одна кухня на все; **мікросервіси** — фудкорт незалежних кіосків.

Stateless-сервери нічого не пам'ятають самі — тому їх легко додавати, прибирати й замінювати; сесії тримають окремо.

Черги й події розв'язують частини системи: у пік нічого не губиться, повільне не гальмує швидке.

Хмара любить **горизонтальне** масштабування; архітектор шукає й дублює **єдині точки відмови**; **логи й метрики** — очі системи.

🗄️ Бази даних і кеш з нуля

👴 Простими словами

База даних — акуратна шафа з підписаними полицями, де нічого не губиться. Кеш — кишеня: туди кладеш те, що береш найчастіше, щоб не бігати до шафи щоразу.

🎯 Що ти дізнаєшся

- що таке база даних і чим вона краща за купу файлів
- як влаштовані таблиці, ключі та мова SQL
- що таке транзакція і чому «все або нічого»
- які бувають NoSQL-бази і коли вони кращі за таблиці
- чим кеш, репліка і бекап відрізняються одне від одного
- що означають OLTP і OLAP

🥐 Досі Оля записувала замовлення в зошит. Коли їх стало сто на день, вона почала губити сторінки, а щоб знайти «що замовляла пані Ірина минулого місяця», треба було гортати годину. «Тобі потрібна база даних», — каже Тарас.

Навіщо база даних

Замовлення можна зберігати й у файлах — наприклад, у таблиці Excel. Поки їх мало, так і роблять. Але коли замовлень тисячі, а працюють з ними одночасно сайт, кухня й бухгалтерія, файли не справляються: хтось перезаписує чужі зміни, пошук триває вічність, дані губляться.

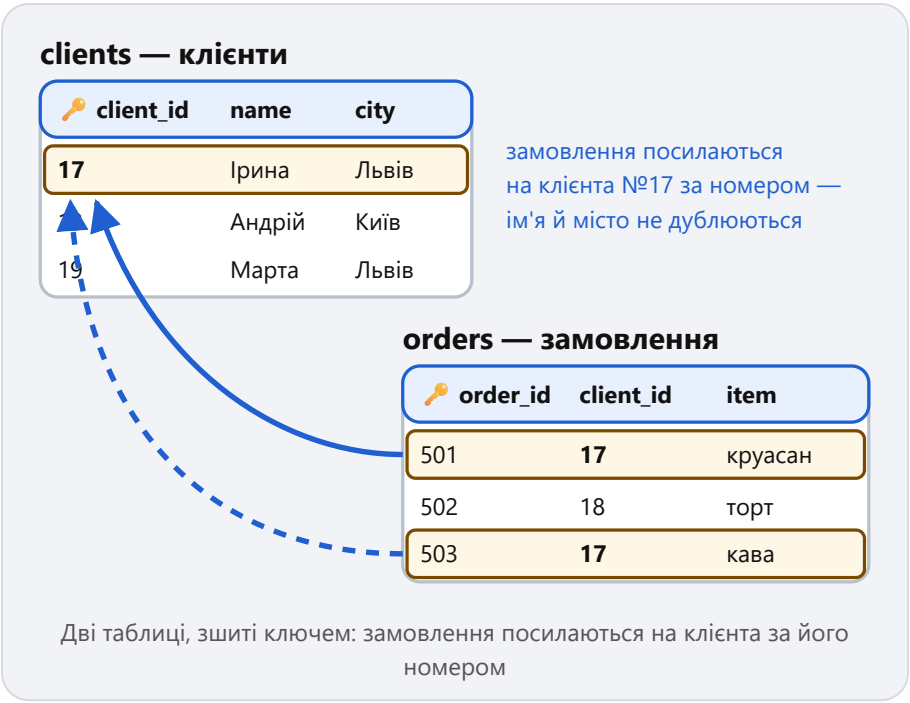
База даних — це **картотека з архіваріусом**. Дані розкладено за чіткими правилами, пошук миттєвий, а архіваріус стежить, щоб двоє людей не зіпсували одну картку одночасно і щоб ніхто не вписав нісенітницю. Самого «архіваріуса» — програму, яка керує базою, — називають **СКБД** (система керування базами даних). Найвідоміші: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Реляційна база: таблиці, як в Excel

Найпоширеніший вид баз — **реляційні** (relational, «рілейшнл»). Дані в них лежать у **таблицях**:

- **стовпці** — поля: ім'я, телефон, місто;
- **рядки** — записи: один рядок — один клієнт;
- **первинний ключ** (primary key) — унікальний номер кожного рядка, як номер паспорта: `client_id = 17`.

А найцікавіше — **зв'язки між таблицями**. У таблиці замовлень не пишуть щоразу ім'я й телефон клієнта — лише його номер `client_id`. Так дані не дублюються, а за потреби таблиці «зшивають» разом. Від цих зв'язків (relations) і походить назва «реляційна».



Дві таблиці, зшиті ключем: замовлення посилаються на клієнта за його номером

SQL — мова запитів

З реляційною базою розмовляють мовою **SQL** (читається «ес-к'ю-ел» або «сіквел», Structured Query Language). Вона дуже схожа на звичайні англійські речення:

```
SELECT name, phone
FROM clients
WHERE city = 'Львів';
```

Читається так: «**вибери** ім'я і телефон **з** таблиці клієнтів, **де** місто — Львів». Ще кілька слів: `INSERT` — додати запис, `UPDATE` — змінити, `DELETE` — видалити, `JOIN` —

зшити таблиці («покажи замовлення разом з іменами клієнтів»).

Писати SQL для іспиту **не треба**. Але треба знати: **SQL, таблиці, JOIN — це реляційні бази**.

Транзакція: все або нічого

Уяви переказ грошей з одного рахунку на інший. Це дві дії: зняти гроші з першого рахунку і додати на другий. А якщо світло вимкнеться між ними? Гроші зникнуть!

Щоб такого не траплялося, бази вміють **транзакції** — групу дій, яка виконується **повністю або ніяк**. Реляційні бази дають чотири гарантії, які називають **ACID** («ейсид»):

- **A**tomicity — атомарність: усе або нічого;
- **C**onsistency — узгодженість: правила бази не порушуються;
- **I**solation — ізольованість: одночасні операції не змішуються;
- **D**urability — довговічність: якщо база сказала «збережено», дані не зникнуть навіть при збої.

Тому гроші, замовлення, склад — класична робота для реляційних баз.

Схема: бланк, надрукований заздалегідь

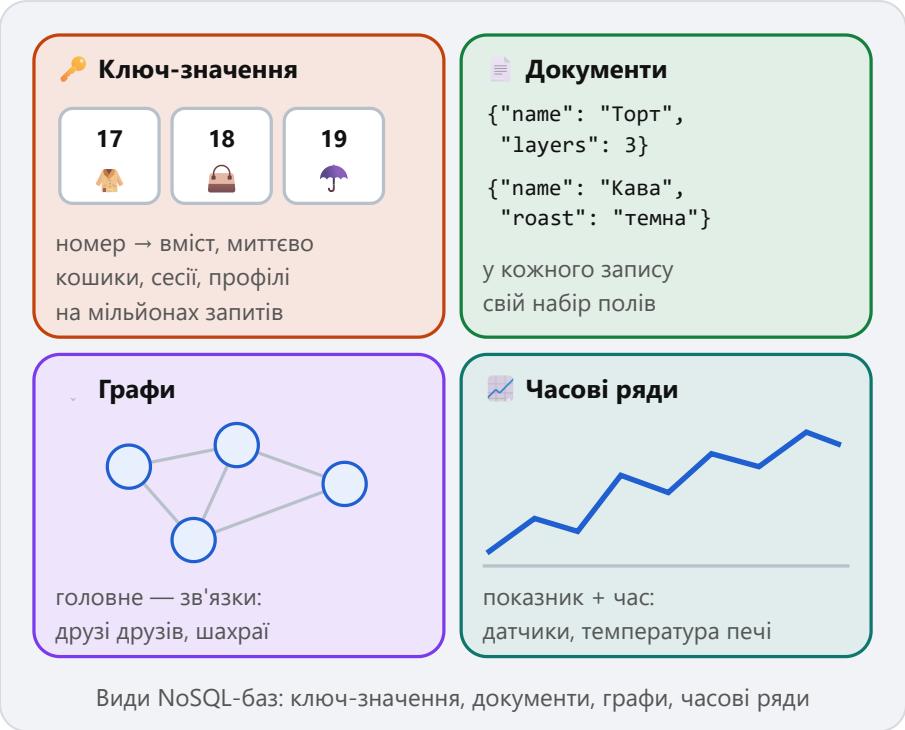
Реляційна таблиця вимагає наперед визначити стовпці — **схему**. Це як друкований бланк: є поле «ім'я», «телефон», «місто» — і нічого іншого вписати не можна. Порядок і передбачуваність — сила таких баз. Але якщо в одного товару 3 характеристики, а в іншого 30, бланк стає незручним.

NoSQL: коли таблиці тісні

NoSQL («ноу-ес-к'ю-ел», «не лише SQL») — загальна назва баз, які зберігають дані не в класичних таблицях. Найважливіші види:

- **Ключ-значення** (key-value) — **шафка з номерками в гардеробі**: даєш номер (ключ) — миттєво отримуєш вміст. Ідеально для мільйонів простих швидких запитів: кошики, сесії, профілі гравців.
- **Документні** — кожен запис — окремий **документ** (часто в JSON) зі своїм набором полів. У торта — «кількість ярусів», у кави — «ступінь обсмаження», і нікому це не заважає.
- **Графові** — **карта знайомств**: головне не записи, а зв'язки між ними. «Друзі друзів», «хто купував разом із цим», ланцюжки шахраїв.

- **Часових рядів** (time series) — показники з мітками часу: температура в печі щосекунди, дані датчиків.



Види NoSQL-баз: ключ-значення, документи, графи, часові ряди

Коротке правило вибору:

- складні зв'язки, JOIN, транзакції з грошима → **реляційна база**;
- величезний масштаб, прості запити за ключем, гнучка структура → **NoSQL**.

OLTP і OLAP: касир і бухгалтер

Бази даних працюють у двох дуже різних режимах:

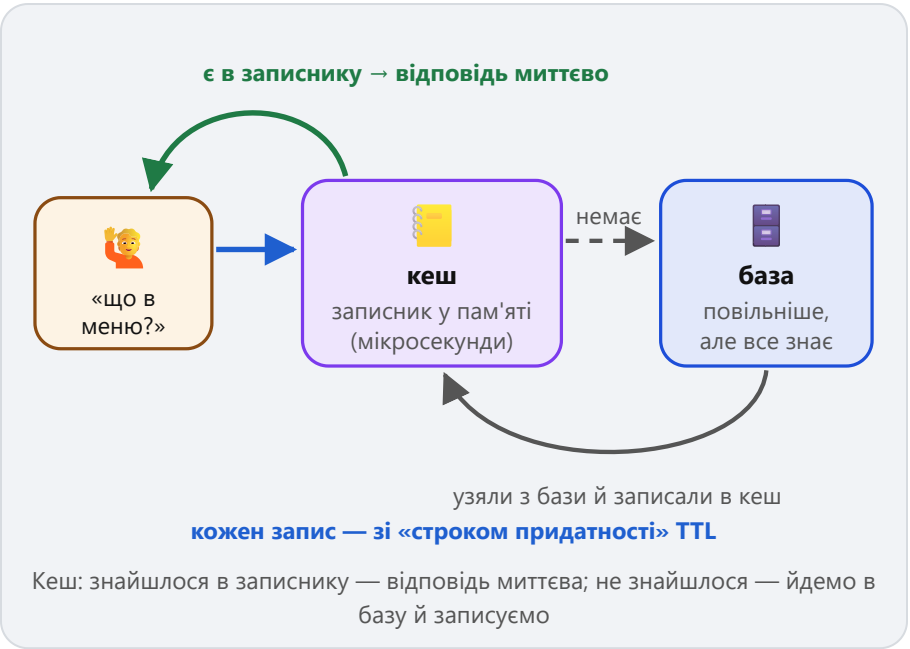
- **OLTP** («оу-ел-ті-пі», online transaction processing) — **касир**: тисячі маленьких швидких операцій щосекунди. «Продати круасан», «записати замовлення», «змінити адресу».
- **OLAP** («оу-лап», online analytical processing) — **бухгалтер наприкінці року**: небагато, але величезних запитів по всій історії. «Продажі кожного товару по місяцях за п'ять років».

Касирові й бухгалтерові потрібні різні інструменти. Для OLTP використовують звичайні бази (реляційні чи NoSQL), а для OLAP — спеціальні **сховища даних** (data warehouse), які вміють швидко перемелювати мільярди рядків. На іспиті: «аналітика по всій історії, складні звіти» → сховище даних, а не основна база сайту.

Кеш: записник на столі

Деякі питання до бази повторюються тисячі разів: «що сьогодні в меню?», «скільки коштує круасан?». Бігати щоразу в архів — марнувати час.

Кеш (cache, «кеш») — **записник на столі** з відповідями на часті питання. Він тримає дані в **пам'яті** (пам'ятаєш — швидкий «робочий стіл»), тому відповідає в десятки й сотні разів швидше за базу, а база відпочиває.



Слабке місце кешу — **застарілі дані**: ціна в базі змінилася, а в записнику ще стара. Тому записам у кеші дають «строк придатності» — **TTL**, як у DNS: минув строк — запис викидають і наступного разу беруть свіжий з бази.

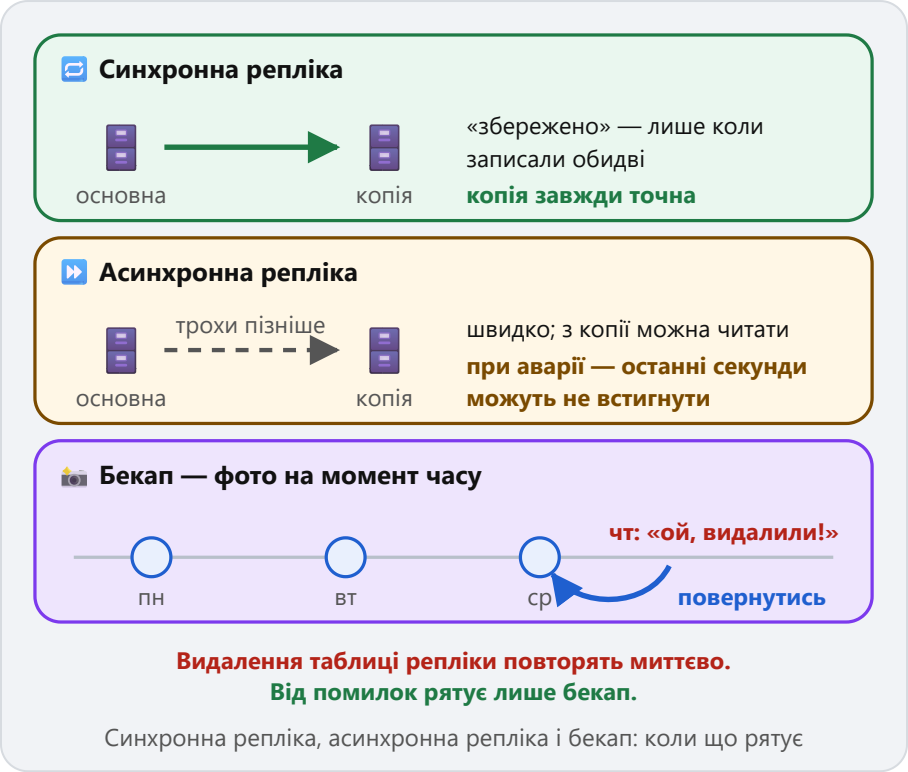
Репліка і бекап: копія копії не рівня

Дані дорогоцінні, тому їх копіюють. Але копії бувають принципово різні.

- **Репліка** (replica, «репліка») — **жива копія** бази на іншому сервері, яка постійно оновлюється. Як колега, що паралельно веде такий самий журнал. Зламався основний сервер — перемикаємося на копію. А ще з репліки можна **читати**, розвантажуючи основну базу.
- **Бекап** (backup, «бекап») — **фотографія** даних на певний момент, яка зберігається окремо. Сама вона не оновлюється.

Реплікація буває двох видів:

- **синхронна** — основна база чекає, поки копія теж запише, і лише тоді каже «збережено». Копія завжди точна, але кожен запис трохи повільніший;
- **асинхронна** — основна база записує й одразу відповідає, а копія наздоганяє з невеликим запізненням. Швидко, але при аварії останні секунди можуть не встигнути скопіюватися.



Синхронна репліка, асинхронна репліка і бекап: коли що рятує

⚠️ Головна пастка: **репліка не захищає від помилок**. Якщо хтось випадково видалив таблицю, репліка слухняно видалить її теж — за мілісекунди. Від помилок, вірусів-вимагачів і «ой, я не те натиснув» рятує лише **бекап** — фото з минулого, до якого можна повернутися.

Ця різниця — основа цілої групи питань іспиту: коли потрібна копія для виживання, коли — для швидкого читання, а коли — бекап.

📖 **Словничок**

База даних — впорядковане сховище даних зі швидким пошуком і правилами.

СКБД — програма-«архіваріус», що керує базою (MySQL, PostgreSQL, Oracle…).

Реляційна база («рілейшнл») — таблиці зі зв'язками між ними.

Таблиця / рядок / стовпець — «аркуш» / один запис / одне поле.

Первинний ключ (primary key) — унікальний номер запису.

SQL («ес-к'ю-ел», «сіквел») — мова запитів до реляційних баз.

JOIN — «зшити» таблиці в одному запиті.

Транзакція / ACID («ейсид») — група дій «все або нічого» / чотири гарантії надійності.

Схема — заздалегідь визначений набір стовпців, «бланк».

NoSQL — бази без класичних таблиць: ключ-значення, документи, графи, часові ряди.

OLTP / OLAP — «касир» (багато дрібних операцій) / «бухгалтер» (великі аналітичні запити).

Сховище даних (data warehouse) — база для важкої аналітики по всій історії.

Кеш — швидкий «записник» у пам'яті для частих відповідей; **TTL** — строк придатності запису.

Репліка — жива копія бази; **синхронна** (точна) чи **асинхронна** (з невеликим запізненням).

Бекап — знімок даних на момент часу; рятує від помилок і видалень.

🧩 **Міні-тест: перевір себе**

? Що таке первинний ключ?

А Пароль від бази даних

Б Унікальний номер запису в таблиці

В Найважливіший стовпець з грошима

Г Ключ від серверної

Відповідь: Б. Первинний ключ — як номер паспорта: за ним завжди знайдеш рівно один запис.

? Банківський переказ: гроші мають і списатися, і зарахуватися — або не зрушити взагалі. Що це гарантує?

А Кеш

Б Бекап

В Транзакція («все або нічого»)

Г Репліка

Відповідь: В. Транзакція виконує групу дій повністю або ніяк. Це буква A в ACID — атомарність.

? Каталог на мільйони товарів, у кожного свої характеристики, і сотні тисяч запитів за секунду за номером товару. Що підходить найкраще?

А Зошит

Б Excel-файл на спільному диску

В NoSQL-база (ключ-значення або документна)

Г Сховище даних для річних звітів

Відповідь: В. Гнучка структура й простий швидкий доступ за ключем на величезному масштабі — сильна сторона NoSQL.

? Співробітник випадково видалив важливу таблицю. Що допоможе її повернути?

А Синхронна репліка

Б Асинхронна репліка

В Бекап, зроблений до видалення

Г Кеш

Відповідь: В. Репліки слухняно повторюють усі зміни, зокрема видалення. Повернутися в минуле дозволяє лише бекап.

? Бухгалтер раз на місяць рахує продажі кожного товару за п'ять років. Це...

А OLTP

Б OLAP

В кешування

Г реплікація

Відповідь: Б. Великі аналітичні запити по всій історії — OLAP, «бухгалтер». Для цього використовують сховища даних.

? Навіщо кеш?

А Швидко віддавати часті відповіді з пам'яті й розвантажити базу

Б Зберігати дані назавжди замість бази

В Шифрувати дані

Г Робити бекапи

Відповідь: А. Кеш — «записник на столі»: миттєво відповідає на часті питання. Але це не заміна базі: у ньому лише копії, які мають строк придатності.

✅ **Головне**

База даних — «картотека з архіваріусом»: швидкий пошук, правила, захист від одночасних змін.

Реляційні бази: таблиці, ключі, **SQL**, **JOIN**, транзакції **ACID** — для грошей, замовлень, складних зв'язків.

NoSQL: ключ-значення, документи, графи, часові ряди — для величезного масштабу й гнучкої структури.

OLTP — «касир» (основна база застосунку); **OLAP** — «бухгалтер» (сховище даних для аналітики).

Кеш — швидкий записник у пам'яті з TTL; **репліка** — жива копія (синхронна або асинхронна); **бекап** — знімок у минулому. Від помилок рятує лише бекап.

🏭 Сервери й дата-центри: де живе інтернет

👴 Простими словами

Інтернет живе не «в повітрі», а у великих будівлях, повних комп'ютерів-серверів, з охороною, кондиціонерами й запасним живленням. Це і є дата-центри.

🎯 Що ти дізнаєшся

- як фізично виглядає сервер і що таке стійка
- з чого складається дата-центр і чому він такий дорогий
- чому «власний сервер у підсобці» — погана ідея для більшості бізнесів
- чому обладнання ламається завжди — і що з цим роблять

🥐 Оля знайшла оголошення: «Сервер для бізнесу, 5 000 доларів». «Куплю, поставлю в підсобці біля борошна — і хай працює», — каже вона. Тарас хитає головою: «Давай спершу подивимося, де живуть справжні сервери. Тоді порахуємо, скільки насправді коштуватиме твоя підсобка».

Як виглядає сервер

Сервер — це комп'ютер без екрана, клавіатури й миші. Зазвичай це плаский металевий ящик, трохи схожий на коробку від піци. Усередині те саме, що й у будь-якому комп'ютері, — процесори, пам'ять, диски, — тільки потужніше і з запасом:

- кілька процесорів з десятками ядер;
- сотні гігабайтів оперативної пам'яті;
- багато дисків;
- **два блоки живлення** — якщо один згорить, другий підхопить;
- кілька мережевих карт для швидкого зв'язку.

Сервери не стоять на столах. Їх вставляють у **стійки** (rack, «рек») — високі металеві шафи з полицями, по кілька

десятків серверів у кожній. Працює сервер цілодобово, роками.

Дата-центр: місто серверів

Дата-центр — будівля, в якій тисячі стійок. Звучить просто, але щоб сервери працювали без перерви, будівля мусить уміти дуже багато:

- **живлення** — дві незалежні лінії від електромережі, величезні акумулятори, щоб пережити секунди перемикання, і **дизельні генератори** на випадок, якщо світла немає годинами;
- **охолодження** — сервери гріються, як тисячі праскок, тож потрібні промислові кондиціонери;
- **мережа** — товсті оптоволоконні канали до інтернету, кілька незалежних;
- **охорона** — огорожа, камери, перепустки, окремі замкнені «клітки» для обладнання різних клієнтів;
- **пожежогасіння** — спеціальне, щоб не залити електроніку водою.



Дата-центр: стійки з серверами, живлення з генераторами, охолодження, мережа й охорона

Збудувати такий дата-центр коштує сотні мільйонів доларів. Утримувати — теж недешево. Тому своїми дата-центрами володіють переважно великі компанії й провайдери — такі як AWS.

«Свій сервер у підсобці» — on-premises

Коли компанія тримає сервери у **власному** приміщенні чи власному дата-центрі, це називають **on-premises** (читається «он-преміс», часто скорочують до «он-прем») — «на своїй території». Це слово ти зустрічатимеш на іспиті постійно: «компанія має on-premises дата-центр і хоче...».

Що насправді означає Олина підсобка:

- **гроші наперед** — тисячі доларів за залізо, ще до першого замовлення;
- **чекати** — доставка й налаштування займають тижні;
- **утримувати** — електрика, кондиціонер, надійний інтернет, бекапи, захист від злодіїв і пожежі;
- **людина на чергуванні** — сервер вирішить зламатися о третій ночі в неділю;
- **замінювати** — за кілька років залізо застаріє.

А найскладніше — **вгадати потужність**. Скільки серверів купувати? Замало — сайт упаде в Чорну п'ятницю, коли всі кинуться купувати торти. Забагато — решту року дорогі сервери гудітимуть вхолосту.



Власні сервери купують під пік — і платять за них увесь рік

Власні сервери: потужність фіксована. Над лінією — гроші на вітер, під нею в пік — сайт падає

Усе ламається — завжди

У великих системах поломка — не рідкісна неприємність, а **щоденна реальність**. Диски зношуються, блоки живлення згорають, мережеві карти глючать. Коли серверів тисячі, щось обов'язково зламане просто зараз. Технічний директор Amazon Вернер Фогельс сформулював це так: «усе ламається, весь час».

Тому досвідчені інженери не сподіваються, що нічого не зламається. Вони будують так, щоб поломка **нічого не зупинила**: два блоки живлення, кілька копій даних, запасні сервери, навіть запасні будівлі. Цей підхід —

дублювання (redundancy, «рідандансі») — ти зустрічатимеш у кожній частині курсу.

Кроки до хмари

Між «підсобкою» і хмарою є проміжні сходинки:

- **колокація** — ставиш своє залізо в чужий професійний дата-центр і платиш за місце, живлення й інтернет;
- **хостинг** — орендуєш цілий сервер у провайдера на місяці.

Хмара — наступний крок. Ти орендуєш не цілий сервер на рік, а **рівно стільки потужності, скільки треба, на хвилини чи секунди**, і керуєш нею через інтернет. Як саме провайдер «нарізає» свої сервери на шматочки для тисяч клієнтів — у наступному розділі.

Гібрид: трохи тут, трохи там

Багато компаній не переносять усе в хмару одразу. Частина систем лишається on-premises, частина працює в хмарі, і вони з'єднані між собою. Це називається **гібридна** інфраструктура (hybrid, «гайбрід»). На іспиті ціла група питань про те, як з'єднати офіс із хмарою, як переносити дані й як жити на два доми.

📖 Словничок

Сервер — потужний комп'ютер без екрана, що працює цілодобово.

Стійка (rack, «рек») — металева шафа з полицями для серверів.

Дата-центр — будівля з тисячами стійок, живленням, охолодженням, мережею й охороною.

On-premises («он-преміс», «он-прем») — власні сервери компанії на її території.

Колокація / хостинг — місце для свого заліза в чужому дата-центрі / оренда цілого сервера.

Дублювання (redundancy, «рідандансі») — запасні копії всього важливого, щоб поломка нічого не зупинила.

Гібридна інфраструктура («гайбрід») — частина систем on-premises, частина в хмарі.

🧩 **Міні-тест: перевір себе**

? Що означає on-premises?

А Сервери в хмарі AWS

Б Власні сервери компанії в її приміщенні чи її дата-центрі

В Сервери, які працюють лише вдень

Г Будь-який ноутбук

Відповідь: Б. «На своїй території»: компанія сама купує, ставить і утримує обладнання.

? Чому купувати власні сервери «під Чорну п'ятницю» — невигідно?

А Більшу частину року вони простоюватимуть, а гроші вже витрачено

Б Бо в Чорну п'ятницю сервери не працюють

В Бо сервери не можна купити

Г Це завжди вигідно

Відповідь: А. Потужність доводиться купувати під пік, а платити за неї весь рік. Хмара дозволяє брати більше лише на час піку.

? Навіщо дата-центру дизельні генератори?

А Щоб гріти серверну взимку

Б Щоб працювати, навіть якщо в місті годинами немає електрики

В Щоб заряджати телефони охорони

Г Для краси

Відповідь: Б. Живлення дублюють: дві лінії, акумулятори на секунди і генератори на години.

? Що з переліченого — неминучий факт для великих систем?

А Якісне обладнання ніколи не ламається

Б Обладнання ламається регулярно, тому все важливе дублюють

В Поломки бувають раз на сто років

Г Сервери ламаються лише від вірусів

Відповідь: Б. «Усе ламається, весь час». Надійність будують не на надії, а на запасі.

? Частина систем компанії працює в її дата-центрі, а частина — в AWS, і вони з'єднані. Як це називається?

А Колокація

Б Гібридна інфраструктура

В Хостинг

Г Резервна копія

Відповідь: Б. Гібрид — «життя на два доми». На іспиті це окрема велика тема.

✅ **Головне**

Сервер — потужний комп'ютер без екрана в **стійці**; **дата-центр** — тисячі стійок з живленням, охолодженням, мережею й охороною.

On-premises — власні сервери: гроші наперед, тижні очікування, утримання й чергування.

Потужність доводиться **вгадувати**: замало — падіння в пік, забагато — гроші на вітер.

Обладнання **ламається завжди**; надійність — це **дублювання** всього важливого.

Гібрид — частина on-premises, частина в хмарі.

🪆 Віртуалізація, контейнери і serverless: як один сервер стає сотнею

👶 Простими словами

Віртуалізація — як поділити великий будинок на квартири: один фізичний сервер стає багатьма «віртуальними». Контейнер — ще менша кімната, а serverless — коли кімнату тобі дають лише на хвилину, поки ти нею користуєшся.

🎯 Що ти дізнаєшся

- як один фізичний сервер ділять на багато віртуальних машин
- що таке гіпервізор і чому сусіди не бачать твоїх даних
- чим контейнер легший за віртуальну машину
- що означає serverless
- як змінюється «хто за що відповідає» від власного сервера до serverless

🥐 «Добре, дата-центр не для мене, — погоджується Оля. — Але як AWS дає мені "мій" сервер, якщо я ніколи його не побачу? І чому він коштує копійки на годину, а не 5 000 доларів?» — «Бо ти отримаєш не залізний сервер, а віртуальний», — відповідає Тарас.

Проблема: сервери нудьгують

Звичайний фізичний сервер більшу частину часу завантажений на 10–20%. Уяви величезний будинок, у якому живе одна людина: решта кімнат порожні, а платити за опалення доводиться за все.

А що, як один потужний сервер поділити між багатьма клієнтами — так, щоб кожен думав, що має власний окремий комп'ютер?

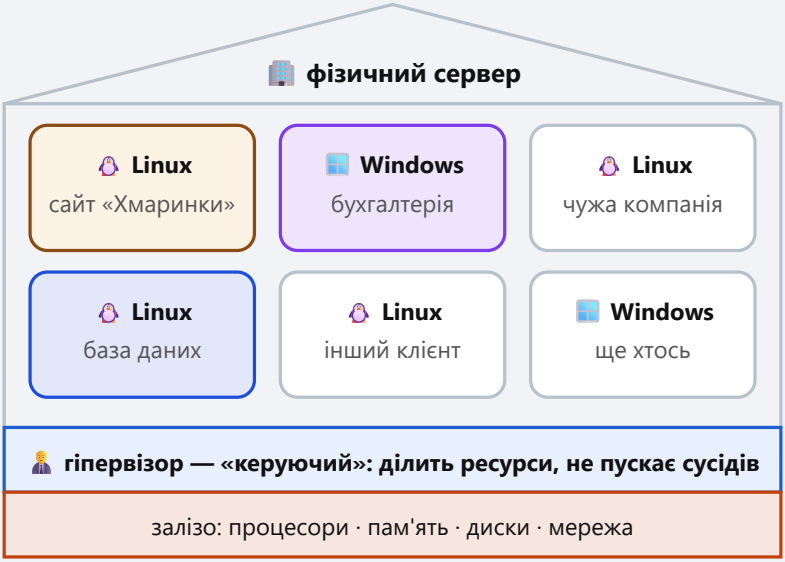
Гіпервізор і віртуальні машини: багатоквартирний будинок

Саме це робить **віртуалізація**. На фізичний сервер ставлять особливу програму — **гіпервізор** (hypervisor, «гіпервайзор»). Він ділить залізо на кілька **віртуальних машин** (VM, «ві-ем»):

- у кожної — свої віртуальні процесори (vCPU), своя пам'ять, свій диск;

- на кожній — своя операційна система: на одній Linux, на сусідній Windows;
- машини **ізольовані**: одна не бачить і не може зачепити іншу.

Образ — **багатоквартирний будинок**. Будівля одна (фізичний сервер), а квартир багато (віртуальні машини). У кожної квартири свої стіни, двері й замок. Керуючий будинку (гіпервізор) розподіляє воду й електрику (процесор і пам'ять) і стежить, щоб сусіди не лазили одне до одного.



кожна «квартира» — віртуальна машина зі своєю ОС і замком

Віртуалізація: один фізичний сервер — багато ізольованих віртуальних машин, як квартири в будинку

Віртуальний сервер в AWS називається **інстанс EC2** (instance — «екземпляр»; EC2 читається «і-сі-ту»). Гіпервізор AWS називається **Nitro** («найтро») — це поєднання спеціального заліза й дуже легкої програми, завдяки якому віртуальні машини працюють майже так само швидко, як «голе» залізо.

Що дає віртуалізація

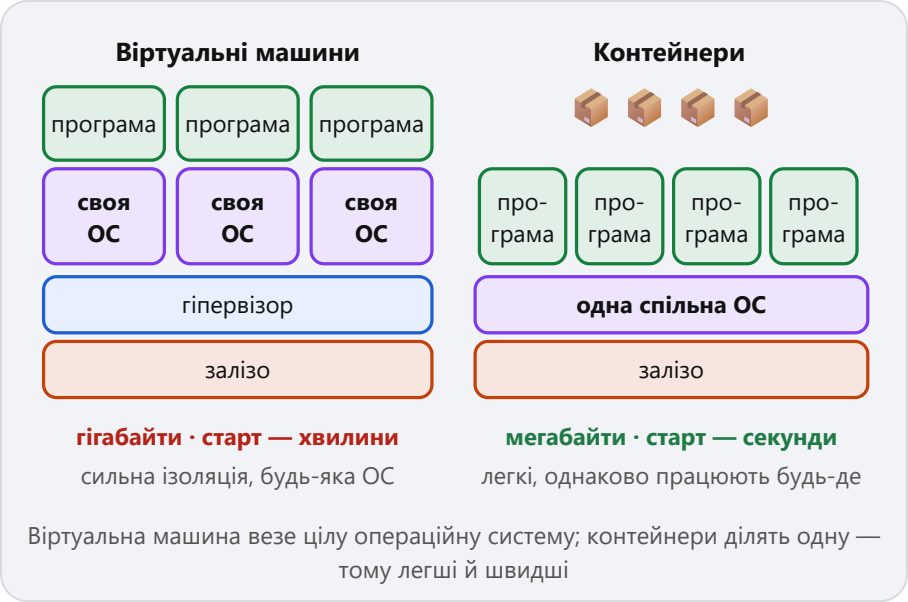
- **Швидкість.** Нова «квартира» готова за хвилини, а не за тижні доставки заліза.
- **Знімки й клони.** Стан цілої машини можна «сфотографувати» й створити з цього знімка десять однакових машин. В AWS такий шаблон називається AMI — до нього дійдемо.
- **Гнучкість.** Потрібно більше пам'яті — переїжджаєш у більшу «квартиру».
- **Економія.** Залізо завантажене щільно, тому кожна квартира коштує недорого.

Контейнери: морські контейнери для програм

Віртуальна машина — річ важка: кожна везе з собою **цілу** операційну систему (гігабайти), і запускається хвилинами. А якщо треба запустити сотні маленьких програм?

Тут з'являються **контейнери** (containers, «контейнерз»). Контейнер не має власної повної ОС — він користується **ядром спільної** операційної системи хоста й везе лише саму програму та все, що їй потрібно (бібліотеки, налаштування). Тому контейнер важить мегабайти і стартує за секунди.

Найкращий образ — **морський контейнер**. До його винаходу кожен вантаж завантажували по-своєму: мішки, бочки, ящики. Стандартний контейнер змінив усе: неважливо, що всередині, — кран, корабель і вантажівка працюють з ним однаково. Так і з програмами: контейнер **однаково працює будь-де** — на ноутбуці розробника, на сервері, у хмарі. Знаменита скарга «а в мене на комп'ютері все працювало» зникає.



Віртуальна машина везе цілу операційну систему; контейнери ділять одну — тому легші й швидші

Два слова, які варто розрізняти:

- **образ** (image, «імідж») — «креслення» контейнера: програма й усе потрібне, запаковане в один файл;
- **контейнер** — запущений образ, що працює просто зараз (як процес і програма).

Найвідоміший інструмент для контейнерів — **Docker** («докер»). В AWS є окремі сервіси, які запускають і доглядають тисячі контейнерів, — про них у частині про обчислення.

Serverless: код без турботи про сервери

Наступна сходинка — **serverless** («сервер-лес», «без серверів»). Ти не орендуєш ні машину, ні контейнер. Ти

даєш хмарі **лише код** — функцію — і кажеш, коли її запускати: «коли клієнт натисне "Замовити"», «коли з'явиться нове фото».

- Код запускається **лише на подію** і працює, поки виконує задачу.
- Платиш **за мілісекунди** роботи. Немає подій — немає витрат.
- Прийшла тисяча подій одночасно — хмара сама запустить тисячу копій.

«Без серверів» не означає, що серверів немає. Вони є, але **це не твоя турбота**: ні встановлювати, ні латати, ні масштабувати. В AWS головний serverless-сервіс називається **Lambda** («лямбда»).

Сходинки: хто за що відповідає

Зведемо все в одну картину — від найбільшого контролю до найменшого клопоту. Образ — житло:



Від власного будинку до таксі: що більше робить провайдер, то менше клопоту тобі

- **Власний сервер (on-premises)** — власний будинок: усе твоє, і все лагодиш ти.
- **Віртуальна машина** — орендована квартира: будинок доглядає власник, а меблі, замки й порядок усередині — твої. Ти ставиш і латаєш операційну систему.
- **Контейнери на керованій платформі** — номер у готелі: приходиш зі своїми речами (образом), а все інше вже готове.
- **Serverless** — таксі: не володієш машиною, не ремонтуєш, платиш лише за поїздку.

На іспиті ця драбинка з'являється як «**least operational overhead**» — «найменше клопоту з експлуатацією». Що

вище сходинка (ближче до serverless і керованих сервісів), то менше клопоту — і тим частіше саме така відповідь правильна.

🔍 **Для допитливих: Чому сусід по «будинку» не прочитає мої дані?**

Гіпервізор розподіляє пам'ять і диск так, що кожна віртуальна машина бачить лише своє. У Nitro більшість функцій гіпервізора винесено в окреме спеціалізоване залізо, а доступ операторів AWS до пам'яті клієнтських машин технічно обмежено. Крім того, дані можна шифрувати власними ключами. А для тих, кому потрібна повна ізоляція (наприклад, через ліцензії чи вимоги регулятора), AWS може виділити цілий фізичний сервер лише під одного клієнта — Dedicated Host. Про це — у розділі про оплату серверів.

📖 **Словничок**

Віртуалізація — поділ одного фізичного сервера на багато ізольованих віртуальних.

Гіпервізор («гіпервайзор») — програма-«керуючий будинку», що ділить залізо між VM.

Віртуальна машина, VM («ві-ем») — «квартира»: свої vCPU, пам'ять, диск і ОС.

Інстанс EC2 («і-сі-ту») — віртуальний сервер в AWS.

Nitro («найтро») — гіпервізор і спеціальне залізо AWS.

Контейнер — легка упаковка програми, що ділить ядро ОС з іншими; стартує за секунди.

Образ (image, «імідж») — «креслення» контейнера чи машини.

Docker («докер») — найвідоміший інструмент для контейнерів.

Serverless («сервер-лес») — даєш лише код, провайдер запускає його на подію; оплата за час роботи.

Lambda («лямбда») — головний serverless-сервіс AWS.

Operational overhead — «клопіт з експлуатацією»: патчі, масштабування, нагляд.

🧩 **Міні-тест: перевір себе**

? Що робить гіпервізор?

А Охолоджує сервер

Б Ділить фізичний сервер на кілька ізольованих віртуальних машин

В Прискорює інтернет

Г Шифрує паролі

Відповідь: Б. Гіпервізор — «керуючий багатоквартирного будинку»: ділить процесор і пам'ять і не пускає сусідів одне до одного.

? Чому контейнер легший і швидший за віртуальну машину?

- **А** Бо він менший фізично
- **Б** Він не везе власну повну операційну систему, а ділить ядро ОС хоста з іншими контейнерами
- **В** Бо контейнери не мають програм
- **Г** Контейнер важчий

Відповідь: Б. VM везе цілу ОС (гігабайти, хвилини на старт), контейнер — лише програму з бібліотеками (мегабайти, секунди).

? Що означає serverless?

А Серверів не існує взагалі

Б Серверами керує провайдер; ти даєш лише код, який запускається на подію, і платиш за час виконання

В Сервер стоїть у тебе вдома

Г Сайт працює без інтернету

Відповідь: Б. Сервери є, але не твоя турбота. Немає подій — немає оплати.

? Як в AWS називається віртуальний сервер?

А S3

Б Інстанс EC2

В Lambda

Г Гіпервізор

Відповідь: Б. EC2 — «Elastic Compute Cloud», оренда віртуальних серверів. Окремий сервер — інстанс.

? У питанні іспиту написано «LEAST operational overhead». Що це найчастіше підказує?

А Обрати власні фізичні сервери

Б Написати скрипти для кожної дії

В Обрати керований або serverless-сервіс, де провайдер бере на себе більше роботи

Г Узяти найпотужніший сервер

Відповідь: В. Що вище сходинка до serverless, то менше клопоту з експлуатацією.

✅ **Головне**

Гіпервізор ділить фізичний сервер на ізольовані **віртуальні машини** — «квартири» в будинку. В AWS VM — це **інстанс EC2**.

Контейнер — «морський контейнер» для програми: без власної ОС, легкий, стартує за секунди, працює однаково будь-де.

Serverless — даєш лише код; він запускається на подію, оплата за мілісекунди. В AWS — **Lambda**.

Драбинка: власний сервер → VM → контейнери на платформі → serverless — що вище, то менше клопоту.

«Least operational overhead» на іспиті → **керовані й serverless-сервіси**.

☁️ Що таке хмара і чому всі туди переїжджають

👴 Простими словами

Хмара — це оренда чужих комп'ютерів через інтернет замість купівлі своїх. Як таксі замість власної машини: береш, коли треба, платиш за поїздку, ремонт — не твоя турбота.

🎯 Що ти дізнаєшся

- точне визначення хмари і що означає «плата за використане»
- шість переваг хмари, які AWS називає офіційно
- чим еластичність відрізняється від масштабованості
- моделі IaaS, PaaS і SaaS — на прикладі пекарні
- що таке публічна, приватна й гібридна хмара

🥐 Тарас показує Олі два варіанти. Перший: сервер за 5 000 доларів і підсобка. Другий: віртуальний сервер в AWS — кілька доларів на місяць на старті, а коли клієнтів побільшає — більше. «А якщо бізнес не піде?» — питає Оля. «Тоді ти просто вимкнеш сервер і перестанеш платити», — каже Тарас. Оля задумливо відкладає оголошення.

Що таке хмара — точно

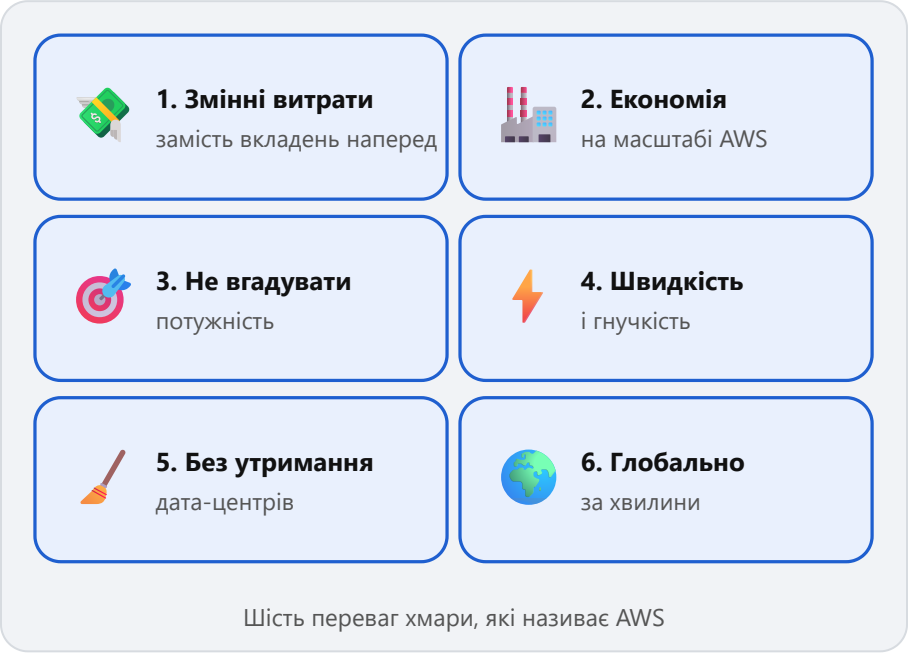
AWS визначає хмару так: **надання IT-ресурсів на вимогу через інтернет з оплатою за використане** (on-demand delivery of IT resources over the internet with pay-as-you-go pricing). Розберемо на шматочки:

- **IT-ресурси** — обчислювальна потужність, сховище, бази даних, мережа, готові сервіси.
- **На вимогу** — натискаєш кнопку (або програма робить запит) і отримуєш ресурс за хвилини. Без дзвінків, договорів і доставки.
- **Через інтернет** — керуєш усім здалеку, з браузера чи з коду.

- **Плата за використане** (pay-as-you-go, «пей-ез-ю-гоу») — як за електрику: ти не будуєш електростанцію, а платиш за кіловат-години. Вимкнеш — лічильник зупиниться.

Шість переваг хмари

AWS офіційно називає шість переваг. Їх корисно знати: вони пояснюють, чому правильні відповіді іспиту «думають» саме так.



Шість переваг хмари, які називає AWS

1. **Змінні витрати замість великих вкладень наперед.** Не купуєш сервери на три роки вперед (капітальні витрати, CapEx), а платиш щомісяця за спожите (операційні витрати, OpEx).
2. **Економія на масштабі.** AWS купує мільйони серверів і терабайти дисків дешевше, ніж будь-хто окремо, — і ця знижка потрапляє в ціни.
3. **Не треба вгадувати потужність.** Замало — додаєш за хвилини, забагато — прибираєш.
4. **Швидкість і гнучкість.** Нові ресурси — за хвилини, тому експериментувати дешево: спробуєш, не підійде — видалиш.
5. **Не витрачаєш гроші на дата-центри.** Кондиціонери, генератори й охорону бере на себе AWS, а ти зосереджуєшся на своєму бізнесі — випічці.
6. **Глобально за хвилини.** Відкрити сайт для клієнтів у Японії, не їдучи до Японії.

Масштабованість і еластичність

Ці слова схожі, але не однакові:

- **Масштабованість** (scalability, «скейлабіліті») — система **вміє рости**: витримати вдесятеро більше клієнтів, додавши ресурсів.

- **Еластичність** (elasticity, «еластісіті») — ресурси **автоматично** ростуть і **зменшуються** разом з попитом, як гармошка. Удень десять серверів, уночі — два, і платиш саме за це.



платиш за зелену лінію, а не за червону

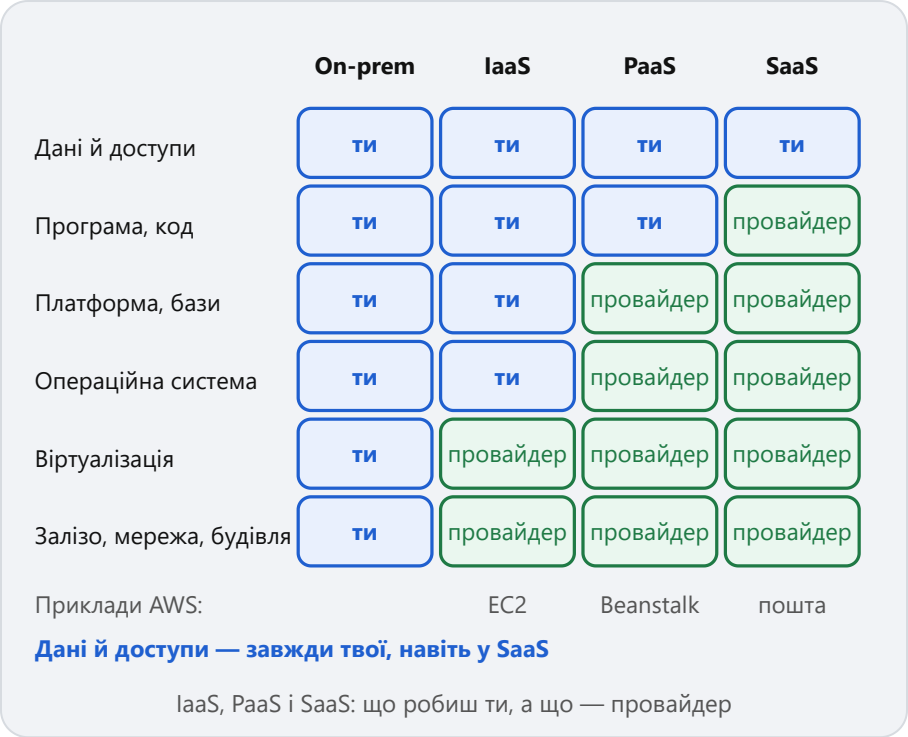
Еластичність: потужність у хмарі йде слідом за попитом, а власні сервери — фіксована лінія

Саме еластичність — головний «фокус» хмари, якого не дасть жодна підсобка. На іспиті вона ховається за словами «unpredictable traffic», «spikes», «scale automatically».

IaaS, PaaS, SaaS: скільки роботи бере провайдер

Хмарні послуги поділяють на моделі — за тим, скільки роботи провайдер бере на себе. Пояснимо на пекарні:

- **Власна пекарня (on-premises)** — усе власноруч: будуєш піч, купуєш обладнання, печеш і прибираєш.
- **IaaS** («ай-ес», Infrastructure as a Service — інфраструктура як послуга) — **орендуєш обладнану кухню**: піч, столи, електрика є, а тісто, рецепти й випічка — твої. В AWS: віртуальні сервери EC2 — ОС, програми й дані на тобі.
- **PaaS** («пес», Platform as a Service — платформа як послуга) — **орендуєш кухню з кухарями**: приносиш лише рецепт, решту роблять вони. В AWS: платформи, куди просто завантажуєш код, і керовані бази даних, які AWS встановлює й латає сам.
- **SaaS** («сас», Software as a Service — програма як послуга) — **купуєш готові круасани** в іншій пекарні: просто користуєшся. Приклади: пошта в браузері, відеозв'язок, онлайн-бухгалтерія.



IaaS, PaaS і SaaS: що робиш ти, а що — провайдер

Serverless-функції (як Lambda) іноді виділяють в окрему модель — FaaS («функція як послуга»): ти даєш тільки код однієї функції.

Публічна, приватна, гібридна

- **Публічна хмара** — AWS, Azure, Google Cloud: ресурси для всіх охочих, кожен клієнт ізольований.
- **Приватна хмара** — хмарні технології у власному дата-центрі компанії, лише для неї.

- **Гібридна** — поєднання: частина систем on-premises, частина в публічній хмарі, з'єднані між собою. Дуже поширено у великих компаніях і на іспиті.
- **Мультихмара** (multi-cloud) — кілька провайдерів одночасно.

Чого хмара не зробить за тебе

Хмара дає **будівельні блоки**: надійні, швидкі, безпечні. Але вона не збере з них правильну систему автоматично. Якщо поставити один сервер в одному дата-центрі й відкрити його всьому інтернету, хмара не врятує ні від падіння, ні від злому.

Будувати з блоків правильно — **робота архітектора**. Тобто твоя. Далі ми вивчатимемо блоки AWS один за одним і вчитимемося їх поєднувати.

> 🥐 Оля вирішила: «Хмаринка» стартує в хмарі. Платити доведеться кілька доларів на місяць, а якщо бізнес виросте — ресурси виростуть разом з ним. Тарас уже реєструє акаунт AWS.

📖 **Словничок**

Хмара (cloud, «клауд») — IT-ресурси на вимогу через інтернет з оплатою за використане.

Pay-as-you-go («пей-ез-ю-гоу») — платиш за фактичне споживання, як за електрику.

CapEx / OpEx («кепекс / опекс») — великі вкладення наперед / поточні витрати за спожите.

Масштабованість (scalability) — уміння рости під більше навантаження.

Еластичність (elasticity) — автоматичне збільшення й зменшення ресурсів разом з попитом.

IaaS («ай-ес») — оренда інфраструктури: віртуальні сервери, мережа, диски.

PaaS («пес») — платформа: приносиш код, решту робить провайдер.

SaaS («сас») — готова програма, якою просто користуєшся.

Публічна / приватна / гібридна хмара — для всіх / лише для однієї компанії / поєднання on-premises і хмари.

🧩 **Міні-тест: перевір себе**

? Що означає «плата за використане» (pay-as-you-go)?

А Оплата наперед за три роки

Б Платиш лише за фактично спожиті ресурси, як за електрику

В Усі сервіси безкоштовні

Г Фіксована щомісячна абонплата незалежно від використання

Відповідь: Б. Вимкнеш ресурс — лічильник зупиниться. Це основа економіки хмари.

? Сайт перевантажений щовечора, а вночі майже порожній. Яка властивість хмари тут найкорисніша?

А Економія на масштабі

Б Еластичність — автоматично додавати ресурси в пік і прибирати вночі

В Глобальність

Г Приватна хмара

Відповідь: Б. Еластичність — «гармошка»: ресурси йдуть слідом за попитом, і платиш саме за це.

? Орендуєш віртуальні сервери, а операційну систему й програми ставиш і оновлюєш самостійно. Яка це модель?

- **А** IaaS
- **Б** PaaS
- **В** SaaS
- **Г** On-premises

Відповідь: А. IaaS — «обладнана кухня в оренду»: інфраструктура провайдера, а все, що на ній, — твоє.

? Користуєшся готовою поштою в браузері й нічого не налаштовуєш. Яка це модель?

- **А** IaaS
- **Б** PaaS
- **В** SaaS
- **Г** Приватна хмара

Відповідь: В. SaaS — «готові круасани з іншої пекарні»: просто користуєшся.

? Яка перевага хмари означає «не купувати сервери на роки вперед, а платити щомісяця за спожите»?

- **А** Змінні витрати замість великих вкладень наперед
- **Б** Глобально за хвилини
- **В** Економія на масштабі
- **Г** Швидкість і гнучкість

Відповідь: А. Капітальні витрати (CapEx) замінюються поточними (OpEx).

✅ **Головне**

Хмара — IT-ресурси **на вимогу**, **через інтернет**, з **оплатою за використане**.

Шість переваг: змінні витрати, економія на масштабі, не вгадувати потужність, швидкість, без утримання дата-центрів, глобально за хвилини.

Еластичність — ресурси автоматично ростуть і зменшуються з попитом; **масштабованість** — уміння рости.

IaaS (обладнана кухня) → **PaaS** (кухня з кухарями) → **SaaS** (готові круасани).

Хмара дає блоки, а правильну систему з них будує **архітектор**.

🗺️ AWS по всьому світу: регіони, зони доступності й edge

👴 Простими словами

AWS — як мережа міст по всьому світу. Регіон — місто, зони доступності (AZ) — окремі райони міста з власним живленням. Якщо в одному районі біда, інші працюють.

🎯 Що ти дізнаєшся

- що таке регіон AWS і як читати його назву
- що таке зона доступності (AZ) і чому головне правило надійності — «щонайменше дві AZ»
- як обирати регіон — чотири причини
- що таке edge locations, Local Zones, Wavelength і Outposts
- які сервіси глобальні, які регіональні, а які прив'язані до однієї зони

🥐 Перші клієнти «Хмаринки» — зі Львова. Але Оля мріє про доставку тортів у Польщу, а її сестра в Канаді вже питає, чи можна замовити подарунок мамі онлайн. «Де ж тоді ставити сервер — у Львові, у Варшаві чи в Торонто?» — «AWS має дата-центри по всьому світу, — каже Тарас. — Треба лише зрозуміти, як вони згруповані».

Регіон — місто AWS

AWS розмістив свої дата-центри в різних куточках світу. Групу дата-центрів в одній географічній місцевості називають **регіоном** (region, «ріджн»). Образ з простого курсу: **регіон — окреме місто AWS**. Таких міст — понад три десятки, і їх постійно більшає.

Кожен регіон має код з трьох частин: *частина світу — напрямок — номер*:

Код	Де це
`eu-central-1`	Франкфурт (Німеччина)
`eu-west-1`	Ірландія
`eu-north-1`	Стокгольм (Швеція)
`us-east-1`	Північна Вірджинія (США)
`ap-northeast-1`	Токіо (Японія)

Регіону в Україні немає. Для українських проєктів найчастіше обирають **Франкфурт** (`eu-central-1`) — він близько й має всі сервіси.

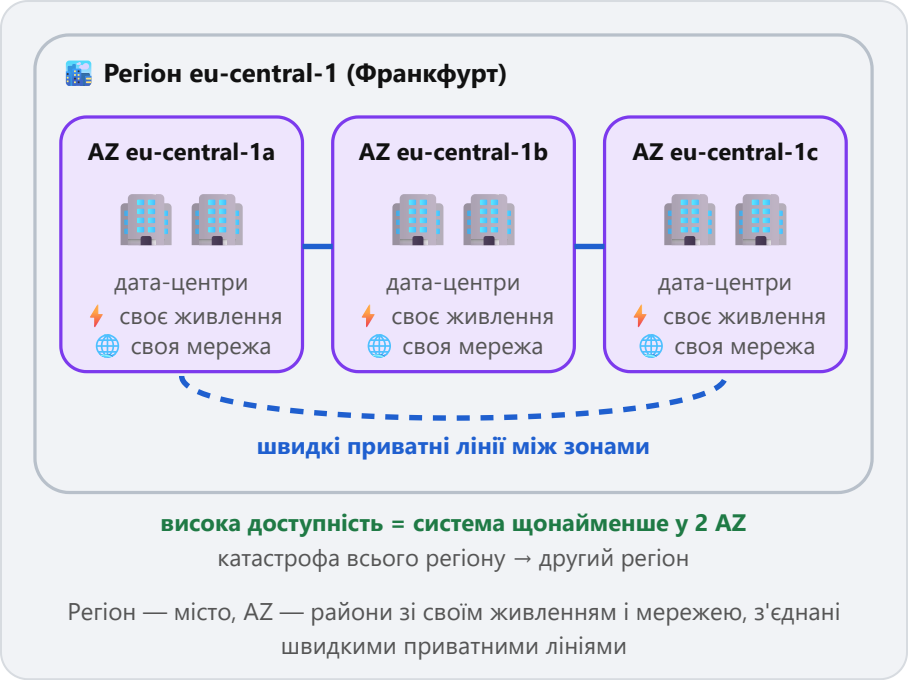
Важлива властивість: регіони **незалежні** один від одного. AWS **не переносить твої дані** з регіону в інший сам — лише коли ти це налаштуєш. Якщо дані мають лишатися в ЄС, достатньо обрати регіон у ЄС.

Зона доступності — окремий район міста

Усередині кожного регіону є кілька (зазвичай три й більше) **зон доступності** — **Availability Zones**, або скорочено **AZ** («ей-зі»). Кожна AZ — це один або кілька дата-центрів з **власним живленням, охолодженням і мережею**. Зони розташовані на відстані одна від одної — досить великій, щоб пожежа, повінь чи збій електромережі в одній не зачепили інші, і досить малій, щоб між ними була дуже швидка приватна мережа.

Образ: **регіон — місто, AZ — окремі райони зі своїми електростанціями**. Якщо в одному районі вимкнули світло, інші живуть як жили.

Зони називають кодом регіону з літерою в кінці: `eu-central-1a`, `eu-central-1b`, `eu-central-1c`.



З цього випливає **найголовніше правило надійності в AWS**:

- щоб пережити аварію одного дата-центру — розміщуй систему **щонайменше у двох AZ**;
- щоб пережити катастрофу цілого регіону (рідко, але буває) — потрібен **другий регіон**.

🎯 На іспиті слова «highly available» (висока доступність) майже завжди означають: **кілька AZ**.

Як обирати регіон

Чотири причини, які варто пам'ятати в такому порядку:

1. **Закон і вимоги** (compliance, «комплаєнс»). Якщо регулятор вимагає, щоб дані лишалися в країні чи в ЄС, — вибір уже зроблено.
2. **Близькість до користувачів** (latency). Ти пам'ятаєш: довга дорога — довга відповідь. Клієнти у Львові — Франкфурт, клієнти в Канаді — канадський регіон.
3. **Потрібні сервіси.** Не кожен сервіс є в кожному регіоні, особливо нові.
4. **Ціна.** У різних регіонах ціни трохи відрізняються.

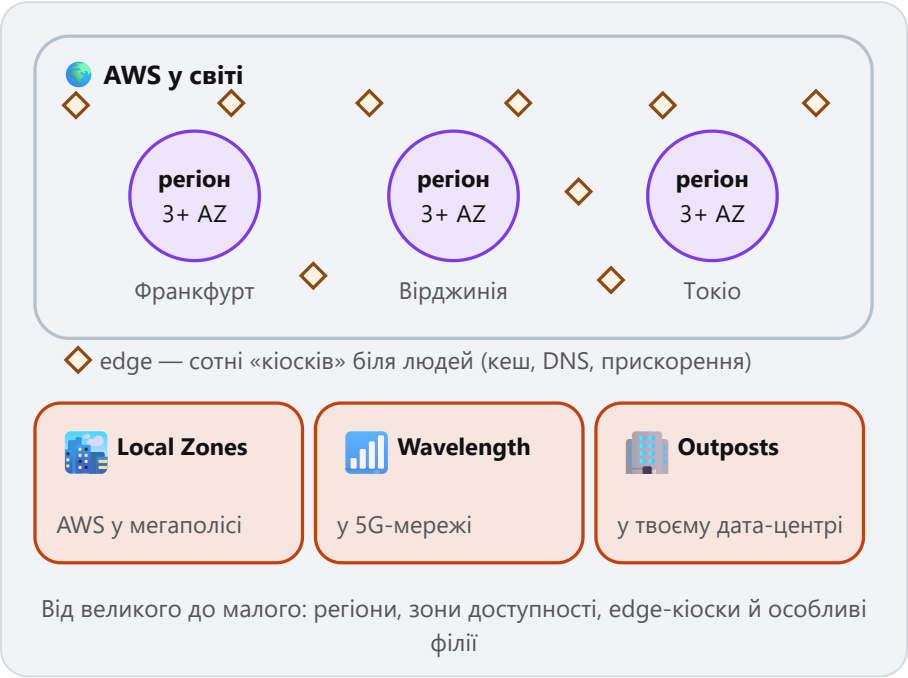
Edge locations — кіоски біля людей

Регіонів кілька десятків, а міст, де живуть люди, — тисячі. Тому AWS поставив ще й сотні маленьких точок присутності у великих містах — **edge locations** («едж локейшнз»), або просто edge («край»). Образ — **кіоски біля людей**: у них не запускають твої сервери, зате там **кешують** контент і **приймають** трафік якомога ближче до користувача.

На edge працюють сервіси для швидкості всього світу: мережа доставки контенту **CloudFront**, DNS **Route 53** і **Global Accelerator**. До них дійдемо в частині про мережу.

Особливі «філії» AWS

- **Local Zones** («локал зоунз») — **AWS у великому місті**: частина сервісів розміщена просто в мегаполісі, щоб затримка для мешканців була одноцифровою в мілісекундах. Для ігор, стрімінгу, обробки відео.
- **Wavelength** («вейвленс») — **AWS усередині 5G-мережі** мобільного оператора: наднизька затримка для телефонів і пристроїв.
- **Outposts** («аутпостс») — **стійка AWS у твоєму власному дата-центрі**: ті самі сервіси й інструменти, але обладнання фізично стоїть у тебе. Для випадків, коли дані мусять лишатися on-premises.



Від великого до малого: регіони, зони доступності, edge-кіоски й особливі філії

Глобальні, регіональні й зональні сервіси

Сервіси AWS живуть на різних «рівнях». Від цього залежить, де вони доступні й що переживуть.

- **Глобальні** — одні на весь світ: керування користувачами й правами (IAM), DNS (Route 53), доставка контенту (CloudFront).
- **Регіональні** — живуть в одному регіоні, але самі розподіляються по його зонах: сховище S3, бази DynamoDB, черги SQS, функції Lambda. Більшість сервісів — саме такі.
- **Зональні** — прив'язані до **однієї AZ**: віртуальний сервер EC2, його диск EBS, підмережа.

> ⚠️ Звідси типова пастка: диск EBS **не можна** підключити до сервера в іншій зоні — вони мусять бути в одній AZ. А «зональний» сервер падає разом зі своєю зоною — тому серверів ставлять кілька в різних AZ.

Особливий регіон us-east-1

Північна Вірджинія (`us-east-1`) — найстаріший і найбільший регіон AWS. Нові сервіси часто з'являються саме тут, а деякі глобальні речі «прописані» саме в ньому. Один факт звідси точно буде на іспиті: сертифікат

для HTTPS у CloudFront треба створювати в **us-east-1**. Поки просто запам'ятай, що цей регіон особливий.

📖 **Словничок**

Регіон (region, «ріджн») — «місто AWS»: група дата-центрів у одній місцевості, напр. `eu-central-1`.

Зона доступності, AZ («ей-зі») — «район»: один або кілька дата-центрів зі своїм живленням і мережею.

Висока доступність (high availability) — система працює, навіть коли одна частина зламалася; в AWS — кілька AZ.

Compliance («комплаєнс») — вимоги законів і регуляторів.

Edge location («едж локейшн») — «кіоск» біля людей для кешування й прийому трафіку.

Local Zones / Wavelength / Outposts — AWS у мегаполісі / у 5G-мережі / у твоєму дата-центрі.

Глобальний / регіональний / зональний сервіс — один на світ / в одному регіоні / в одній AZ.

🧩 **Міні-тест: перевір себе**

? Що таке Availability Zone?

А Окрема країна

Б Один або кілька дата-центрів у регіоні з окремим живленням і мережею, ізольованих від аварій в інших зонах

В Кіоск для кешування біля користувачів

Г Інший акаунт AWS

Відповідь: Б. AZ — «район міста зі своєю електростанцією». Аварія в одній зоні не зачіпає інші.

? Як захистити застосунок від аварії одного дата-центру?

А Купити потужніший сервер

Б Розгорнути його щонайменше у двох зонах доступності

В Переїхати в інший акаунт

Г Вимкнути його на ніч

Відповідь: Б. Головне правило: висока доступність = кілька AZ. Від катастрофи всього регіону — другий регіон.

? Закон вимагає, щоб дані клієнтів з ЄС лишалися в ЄС. Що зробити насамперед?

- **А** Обрати регіон у ЄС — AWS сам дані в інші регіони не переносить
- **Б** Обрати us-east-1, бо він найбільший
- **В** Нічого, AWS зберігає все в ЄС
- **Г** Використати edge locations

Відповідь: А. Регіони незалежні: дані покидають регіон, лише якщо ти це налаштуєш.

? Для чого потрібні edge locations?

- **А** Щоб запускати там свої сервери
- **Б** Щоб кешувати контент і приймати трафік ближче до користувачів
- **В** Щоб зберігати бекапи
- **Г** Щоб будувати приватні мережі

Відповідь: Б. Edge — «кіоски біля людей». Твої сервери там не живуть.

? Що з переліченого прив'язане до однієї зони доступності?

- **А** Сховище S3
- **Б** Керування правами IAM
- **В** Диск EBS
- **Г** DNS Route 53

Відповідь: В. EC2, EBS і підмережа — зональні. S3 — регіональний, IAM і Route 53 — глобальні.

? Потрібні сервіси AWS, але обладнання й дані мусять лишатися у власному дата-центрі компанії. Що обрати?

- **А** Local Zones
- **Б** Wavelength
- **В** AWS Outposts
- **Г** Інший регіон

Відповідь: В. Outposts — стійка AWS у твоєму дата-центрі. Local Zones — AWS у мегаполісі, Wavelength — у 5G-мережі.

✅ **Головне**

Регіон — «місто AWS» (`eu-central-1` — Франкфурт); дані не покидають регіон без твоєї команди.

AZ — «райони» з окремим живленням і мережею; висока доступність = **щонайменше дві AZ**, катастрофа регіону — **другий регіон**.

Регіон обирають за **законом**, **близькістю до людей**, **наявністю сервісів** і **ціною**.

Edge — кіоски біля людей; **Local Zones** — мегаполіс, **Wavelength** — 5G, **Outposts** — твій дата-центр.

Сервіси бувають **глобальні** (IAM, Route 53, CloudFront), **регіональні** (S3, DynamoDB, Lambda) і **зональні** (EC2, EBS).

🔑 Як виглядає AWS зсередини: акаунт, консоль, API і рахунки

👴 Простими словами

Акаунт AWS — твоя особиста квартира в хмарі. Консоль — вікно з кнопками, API — ті самі кнопки, але для програм. А рахунок приходить, як за комуналку: скільки спожив — стільки й платиш.

🎯 Що ти дізнаєшся

- що таке акаунт AWS і хто такий root
- три способи керувати AWS — і чому всі вони насправді одне й те саме
- що таке ARN, теги й квоти
- як AWS рахує гроші і як безкоштовно вчитися
- як не отримати рахунок-сюрприз

🥐 Тарас реєструє акаунт AWS для «Хмаринки»: пошта, пароль, банківська картка. «Картка?! — лякається Оля. — А якщо мені прийде рахунок на мільйон?» — «Не прийде, якщо знати кілька правил, — усміхається Тарас. — Зараз налаштуємо сторожа, який напише тобі, щойно витрати перевищать п'ять доларів».

Акаунт — твоя ділянка в місті AWS

Акаунт AWS (account) — це твоя окрема **ділянка** в місті AWS: усе, що ти створиш (сервери, сховища, бази), належить акаунту, а рахунок за це приходить йому. Акаунти ізольовані: з одного акаунта не видно ресурсів іншого, якщо спеціально не дозволити.

У кожного акаунта є номер з **12 цифр** — Account ID, наприклад `123456789012`.

Той, хто створив акаунт (email і пароль), — **root user** («рут юзер»): власник з **майстер-ключем** від усього. Пам'ятаєш правило з розділу про операційні системи? Майстер-ключем щодня не користуються. Для щоденної роботи створюють окремих користувачів з обмеженими правами — про це весь наступний розділ про безпеку. А

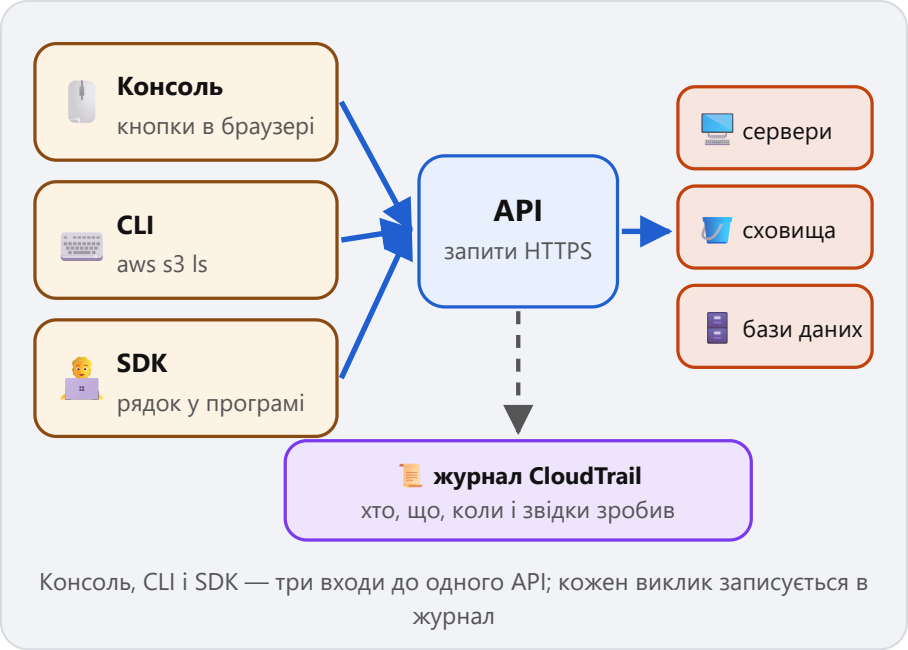
root захищають **MFA** — другим «замком» (кодом з телефона).

Великі компанії мають не один акаунт, а десятки й сотні: окремо для сайту, для тестів, для бухгалтерії. Як ними керувати разом — теж у частині про безпеку.

Три способи керувати AWS

1. **Консоль** (AWS Management Console) — вебсайт з кнопками, таблицями й майстрами налаштування. Зручно, щоб роздивитися й навчитися.
2. **CLI** («сі-ел-ай») — командний рядок. Наприклад, команда `aws s3 ls` покаже всі твої сховища S3.
3. **SDK** («ес-ді-кей») — бібліотеки для мов програмування (Python, JavaScript, Java…). Програма сама створює й змінює ресурси.

А тепер головне: **під капотом усі три способи роблять одне й те саме — викликають API** AWS. Кнопка в консолі, команда в CLI і рядок коду перетворюються на однаковий захищений запит HTTPS до AWS: «створи сервер», «покажи список файлів».



Консоль, CLI і SDK — три входи до одного API; кожен виклик записується в журнал

З цього випливають дві суперсили хмари:

- **Усе можна автоматизувати.** Будь-яку дію, яку можна зробити мишкою, можна записати в код і повторити тисячу разів.
- **Усе можна записати.** Кожен виклик API можна зафіксувати в журналі: хто, коли, звідки і що саме зробив. Сервіс, що веде цей журнал, називається **CloudTrail** — ти зустрінеш його в частині про безпеку.

Регіон у консолі: найчастіша пастка новачків

Угорі консолі є перемикач регіону. Усе, що ти створюєш, з'являється в **поточному** обраному регіоні.

Класична історія: «Вчора створили сервер, а сьогодні його немає!» — а він є, просто консоль відкрилася в іншому регіоні. Спершу перевір регіон — потім панікуй.

ARN — повна адреса ресурсу

Кожен ресурс в AWS має унікальну повну адресу — **ARN** («ей-ар-ен», Amazon Resource Name). Вона схожа на поштову адресу, де вказано все: країну, місто, вулицю, будинок.

`arn:aws:s3:::khmarynka-photos` — сховище S3 з назвою `khmarynka-photos`. `arn:aws:iam::123456789012:user/olia` — користувачка `olia` в акаунті `123456789012`.

Будова: `arn : розділ : сервіс : регіон : акаунт : ресурс`. Деякі частини можуть бути порожніми (у S3 немає регіону й акаунта в ARN, бо назви сховищ унікальні на весь світ).

Навіщо це? У **правилах доступу** пишуть саме ARN: «користувачці olia дозволено читати `arn:aws:s3:::khmarynka-photos/*`». Скоро ти читатимеш такі правила.

Теги — ярлики на ресурсах

До будь-якого ресурсу можна причепити **теги** (tags) — ярлики «ключ = значення»:

- `project = site`
- `env = prod` (робоче середовище) або `env = test`
- `owner = taras`

Теги допомагають **знайти** ресурси, **розподілити витрати** («скільки коштує сайт, а скільки тести?») і навіть **керувати доступом** («розробникам можна чіпати лише ресурси з `env = test`»).

Квоти — ліміти акаунта

У кожного сервісу є **квоти** (Service Quotas) — ліміти на акаунт у регіоні: скільки серверів можна запустити, скільки функцій одночасно тощо. Вони захищають і тебе (від випадкового запуску тисячі серверів), і AWS. Більшість квот — «м'які»: їх можна підняти запитом.

> 🎯 Корисний факт для іспиту: якщо плануєш у разі аварії запустити багато серверів у **запасному регіоні**, квоти там треба **підняти заздалегідь** — інакше в найгірший момент AWS не дасть запустити потрібну кількість.

Як AWS рахує гроші

У хмарі немає фіксованої абонплати. Рахунок складається з того, що **спожито**:

- **обчислення** — за секунди чи години роботи серверів і функцій;
- **зберігання** — за гігабайти на місяць;
- **запити** — за кожну тисячу чи мільйон звернень до деяких сервісів;
- **передача даних** — **вхід** в AWS безкоштовний, а **вихід** в інтернет і переміщення між зонами та регіонами — платні.

Рахунок приходить щомісяця, а поточні витрати видно в розділі **Billing and Cost Management**.

Як вчитися безкоштовно

Нові акаунти (з липня 2025 року) отримують **безкоштовний план** (Free plan) приблизно на **6 місяців**: **100 доларів кредитів** одразу і ще до 100 доларів — за виконання кількох навчальних кроків (запустити сервер, створити базу даних, функцію, спробувати генеративний AI і налаштувати бюджет). На безкоштовному плані AWS **не знімає грошей** з картки; коли кредити чи 6 місяців закінчуються, акаунт пропонують перевести на платний план, інакше його закривають. Деякі дорогі сервіси на безкоштовному плані недоступні. Умови можуть змінюватися — актуальні завжди на сторінці aws.amazon.com/free.

Звідси порада для навчання: **навчальний акаунт — для експериментів**, а не для важливих даних.

Як не отримати рахунок-сюрприз

1. **Налаштуй AWS Budgets** — «сторожа витрат», який напише на пошту, коли витрати чи їхній прогноз перевищать заданий поріг.
2. **Прибирай за собою.** Експеримент закінчено — видали сервер, базу, диск. Зупинений сервер може далі коштувати за диски.
3. **Увімкни MFA для root** і не користуйся root щодня.
4. **Ніколи не публікуй ключі доступу** (про них у розділі про безпеку). Боти цілодобово шукають у відкритих репозиторіях випадково викладені ключі AWS і за лічені хвилини запускають на чужі гроші майнинг криптовалют.

🧪 **Спробуй у справжньому AWS: Налаштуй «сторожа витрат» і захисти root (10 хвилин, безкоштовно)**

Ця вправа безпечна: вона нічого не запускає і нічого не коштує. На безкоштовному плані налаштування бюджету ще й додає кредити.

1. Увійди в консоль AWS: **console.aws.amazon.com**.

2. У рядку пошуку вгорі набери **Budgets** і відкрий **Billing and Cost Management** → **Budgets**.
3. Натисни **Create budget**, обери простий шаблон (**Use a template**) — **Monthly cost budget** або **Zero spend budget** («сповістити про будь-які витрати»).
4. Для місячного бюджету введи суму, наприклад **5 USD**, і свою пошту. Натисни **Create budget**.
5. Тепер захисти root: клацни на назву акаунта в правому верхньому куті → **Security credentials** → **Assign MFA device**.
6. Обери **Authenticator app**, відскануй QR-код застосунком-автентифікатором на телефоні (Google Authenticator, Microsoft Authenticator тощо) і введи два коди поспіль.

Готово: тепер AWS попередить про витрати, а без телефона в root-обліковку не зайти.

📖 **Словничок**

Акаунт AWS — «ділянка»: контейнер для всіх ресурсів і рахунків; номер з 12 цифр.

Root user («рут юзер») — власник акаунта з повною владою; захищають MFA і не використовують щодня.

MFA («ем-еф-ей») — багатофакторна автентифікація: пароль + код з телефона.

Консоль / CLI / SDK — кнопки / команди / бібліотеки для коду; усі викликають API.

API-виклик — запит до AWS «зроби те-то»; кожен можна записати в журнал.

ARN («ей-ар-ен») — повна унікальна адреса ресурсу.

Тег — ярлик «ключ = значення» на ресурсі.

Service Quotas — ліміти сервісів в акаунті й регіоні; більшість можна підняти.

AWS Budgets — «сторож витрат»: сповіщення про перевищення бюджету чи прогнозу.

Free plan — безкоштовний навчальний план для нових акаунтів з кредитами.

🧩 **Міні-тест: перевір себе**

? Хто такий root user?

А Власник акаунта з повною владою — той, чиїми email і паролем акаунт створено

Б Будь-який працівник з доступом до консолі

В Технічна підтримка AWS

Г Програма, що запускає сервери

Відповідь: А. Root має майстер-ключ. Його захищають MFA і не використовують для щоденної роботи.

? Вчора створено сервер, а сьогодні в консолі його не видно. Що перевірити насамперед?

А Чи не видалив його AWS

Б Чи в консолі обрано той самий регіон

В Чи оплачено інтернет

Г Чи не закінчилися квоти

Відповідь: Б. Консоль показує ресурси лише поточного регіону. Інший регіон — інший «список».

? Кнопка в консолі й команда в CLI — через що насправді виконується дія?

А Через електронну пошту

Б Через виклик API AWS

В Через телефонний дзвінок у підтримку

Г Кнопки й команди працюють по-різному

Відповідь: Б. Усі способи — лише різні входи до одного API. Тому все можна автоматизувати й записати в журнал.

? Навіщо ставити на ресурси теги на кшталт project = site?

А. Щоб ресурси працювали швидше

Б. Щоб знаходити ресурси, розподіляти витрати й керувати доступом

В. Щоб зашифрувати дані

Г. Теги потрібні лише для краси

Відповідь: Б. Теги — ярлики: за ними шукають, рахують витрати окремих проєктів і навіть дають права.

? Як вчасно дізнатися, що витрати перевищують заплановані?

А Чекати на рахунок наприкінці місяця

Б Налаштувати AWS Budgets зі сповіщенням на пошту

В Щодня вимикати всі ресурси

Г Попросити сусіда стежити

Відповідь: Б. Budgets — «сторож витрат»: пише, коли витрати чи прогноз перевищують поріг.

? Що таке ARN?

А Назва регіону

Б Унікальна повна «адреса» ресурсу в AWS

В Пароль від акаунта

Г Тип сервера

Відповідь: Б. ARN — як поштова адреса з усіма частинами. У правилах доступу вказують саме ARN.

✅ **Головне**

Акаунт — ізольована «ділянка» з усіма ресурсами й рахунками; **root** — майстер-ключ, під MFA і не щодня.

Консоль, CLI, SDK — три входи до одного **API**: усе можна автоматизувати й записати в журнал.

Ресурси створюються в **поточному регіоні**; **ARN** — повна адреса ресурсу; **теги** — ярлики для пошуку, витрат і доступу; **квоти** — ліміти, які можна підняти.

Платиш за **обчислення, зберігання, запити й вихідний трафік**; вхід в AWS безкоштовний.

Безпека гаманця: **Budgets**, прибирати за собою, MFA на root, ніколи не публікувати ключі.

⚖️ Словник архітектора: надійність, швидкість, безпека, ціна

👴 Простими словами

Архітектор постійно зважує чотири речі: щоб не ламалося, щоб працювало швидко, щоб було безпечно і щоб не коштувало зайвого. Майже кожне питання іспиту — про цей баланс.

🎯 Що ти дізнаєшся

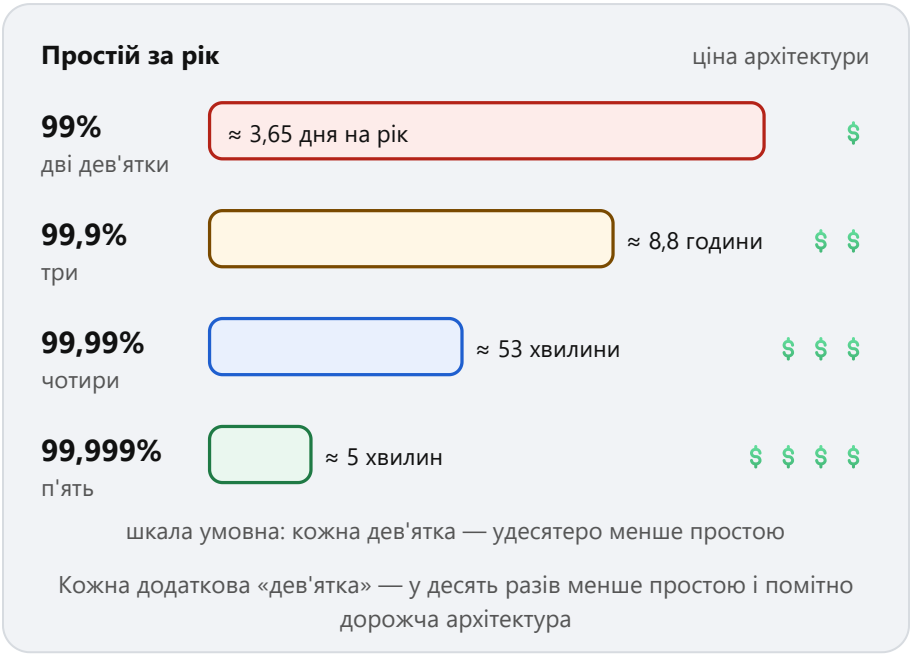
- що означають «три дев'ятки» і «одинадцять дев'яток» і чим доступність відрізняється від надійності зберігання
- чим висока доступність відрізняється від відмовостійкості
- що таке RPO і RTO
- хто за що відповідає в хмарі — модель спільної відповідальності
- шість стовпів Well-Architected, на яких стоїть увесь іспит

🥐 Оля формулює побажання до майбутнього сайту: «Щоб ніколи не падав. Щоб усе відкривалося миттєво. Щоб ніхто не зламав. І щоб недорого». Тарас записує й сміється: «Це чотири побажання кожного замовника у світі. І вони трохи сваряться між собою. Давай навчимося говорити про них точно — так, як говорять архітектори».

Доступність і «дев'ятки»

Доступність (availability, «евейлабіліті») — яку частку часу система працює. Її міряють у відсотках, а говорять «дев'ятками»:

Доступність	Простій за рік	Простій за місяць
99% («дві дев'ятки»)	≈ 3,65 дня	≈ 7,3 години
99,9% («три»)	≈ 8,8 години	≈ 44 хвилини
99,99% («чотири»)	≈ 53 хвилини	≈ 4,4 хвилини
99,999% («п'ять»)	≈ 5 хвилин	≈ 26 секунд



Кожна дев'ятка дається все дорожче: щоб з 99,9% стало 99,99%, треба дублювати вже не лише сервери, а й бази, мережу, цілі зони. Тому архітектор завжди питає: **скільки простою бізнес справді може собі дозволити?** Пекарні 99,9% вистачить з головою, банку — ні.

Обіцянку провайдера щодо доступності записують в **SLA** («ес-ел-ей», Service Level Agreement — угода про рівень обслуговування).

Надійність зберігання: чи не загубляться дані

Надійність зберігання (durability, «дюрабіліті») — наскільки ймовірно, що збережені дані **не загубляться**. Сховище S3 обіцяє **одинадцять дев'яток** — 99,999999999%. Образно: з 10 мільйонів файлів у середньому губиться один за 10 000 років.

> ⚠️ Не плутай: **доступність** — «двері складу відчинені просто зараз». **Надійність зберігання** — «речі на складі цілі». Склад може бути зачинений годину (доступність постраждала), але жодна коробка не зникне (надійність ціла). На іспиті «11 дев'яток» — це саме **durability**.

Висока доступність, відмовостійкість і аварійне відновлення

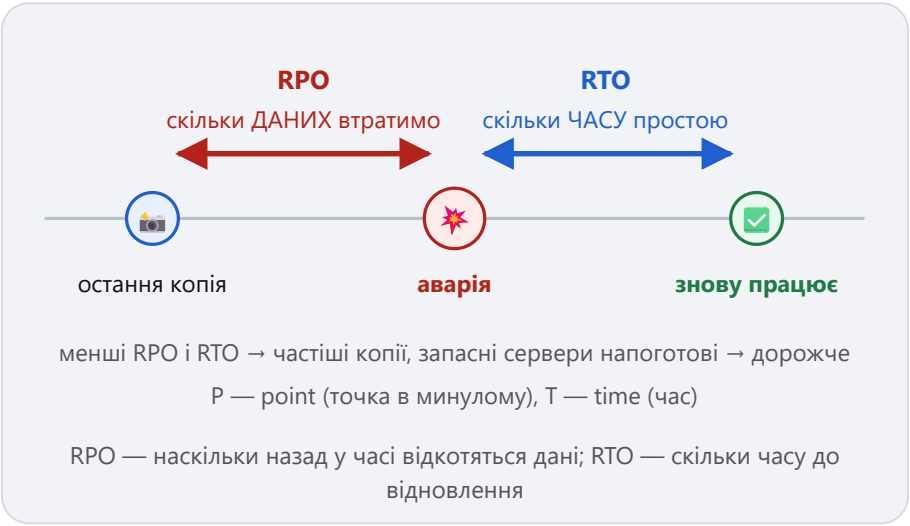
Три схожі, але різні поняття:

- **Висока доступність** (high availability, HA) — **запасне колесо**: щось зламалося, коротка зупинка — і їдемо далі. Користувач може помітити паузу в кілька секунд чи хвилину.
- **Відмовостійкість** (fault tolerance, «фолт толеранс») — **двомоторний літак**: один мотор відмовив, а політ триває взагалі без перерви. Дорожче, бо все працює в подвійному екземплярі постійно.

- **Аварійне відновлення** (disaster recovery, DR, «ді-ар») — **план на випадок пожежі в цілому місті**: як повернути систему, якщо зник цілий регіон.

Для DR є два числа, які ти зустрічатимеш у десятках питань:

- **RPO** («ар-пі-оу», Recovery Point Objective) — скільки **даних** можна втратити. «Не шкода останніх 15 хвилин замовлень» → RPO = 15 хвилин.
- **RTO** («ар-ті-оу», Recovery Time Objective) — скільки **часу** можна простояти. «За годину сайт має знову приймати замовлення» → RTO = 1 година.



Мнемоніка: **P — Point**, точка в минулому, до якої повернемося (дані). **T — Time**, час простою.

Масштабованість, продуктивність і IOPS

Ти вже знаєш **масштабованість** (уміння рости) й **еластичність** (рости й зменшуватися автоматично), **вертикальне** (більший сервер) і **горизонтальне** (більше серверів) масштабування. Для продуктивності додамо три міри:

- **затримка** (latency) — за скільки мілісекунд приходить відповідь;
- **пропускна здатність** (throughput, «сру-пут») — скільки запитів чи мегабайтів за секунду система переробляє;
- **IOPS** («ай-опс», input/output operations per second) — скільки окремих операцій читання чи запису за секунду робить диск. Образ: **скільки коробок за секунду комірник виносить із комори**. Для баз даних, де багато дрібних операцій, IOPS — головне.

Безпека: три букви CIA

Безпеку в IT часто описують трьома буквами **CIA** (не розвідка!):

- **Confidentiality** — **конфіденційність**: чужі не прочитають;
- **Integrity** — **цілісність**: ніхто не змінить непомітно;
- **Availability** — **доступність**: система працює, коли потрібна (наприклад, не впала від атаки).

І два принципи, які ти бачитимеш у правильних відповідях постійно:

- **мінімальні права** (least privilege, «ліст привілідж») — кожен отримує рівно ті права, що потрібні для роботи, і не більше;
- **захист на всіх рівнях** (defense in depth) — не одні двері з замком, а кілька рубежів: огорожа, двері, сейф, сигналізація.

Вартість: найдешевше, що виконує все

У хмарі платиш за використане, тому економія — це правильний вибір: потрібний розмір, потрібний спосіб оплати, вимикати непотрібне. Але на іспиті є важливе правило:

> ⚠️ «Найдешевше» (MOST cost-effective) — це найдешевший варіант **з тих, що виконують усі вимоги**. Дешевий варіант без потрібної доступності чи безпеки — неправильний, хоч би який він був дешевий.

Хто за що відповідає: модель спільної відповідальності

Згадай оренду квартири. Фундамент, стіни, дах і ліфт — турбота **власника будинку**. А хто має ключі від твоїх

дверей, що лежить у шафі й чи зачинене вікно — **твоя** відповідальність.

У хмарі так само. Це називається **модель спільної відповідальності** (Shared Responsibility Model):

- **AWS** відповідає за безпеку **хмари** (security **OF** the cloud): будівлі дата-центрів, залізо, мережу, гіпервізор, внутрішню «кухню» керованих сервісів.
- **Клієнт** відповідає за безпеку **в хмарі** (security **IN** the cloud): свої дані, хто має доступ, шифрування, налаштування мережевих охоронців, а на віртуальних серверах — ще й **патчі операційної системи**.

	EC2	Керована база	Lambda, S3
Дані, доступи, ключі	клієнт	клієнт	клієнт
Код і налаштування	клієнт	клієнт	клієнт
ОС і патчі	клієнт	AWS	AWS
Рушій бази, середовище	клієнт	AWS	AWS
Залізо, дата-центри	AWS	AWS	AWS

IN the cloud — клієнт: дані, доступи, налаштування

OF the cloud — AWS: залізо, дата-центри, внутрішня «кухня»

Межа відповідальності рухається: що більш керований сервіс, то менше роботи клієнту — але дані й доступи завжди твої

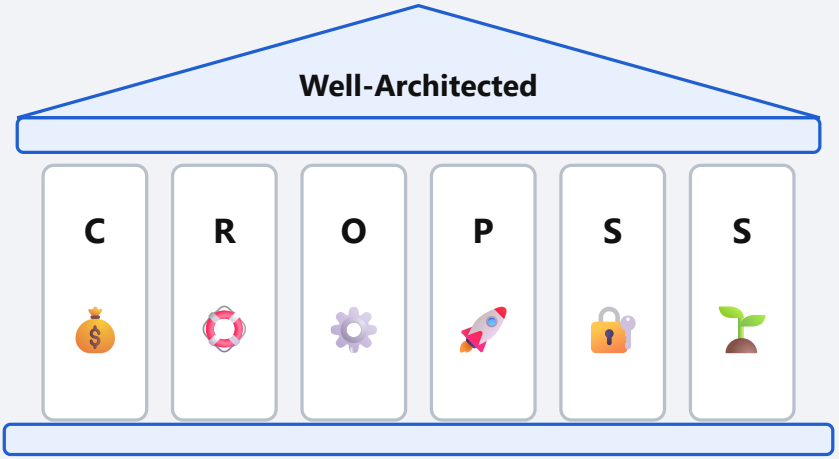
Межа рухається залежно від сервісу:

- **EC2** (віртуальний сервер) — ОС, патчі, програми, мережеві правила — на тобі.
- **Керована база даних** — ОС і рушій бази латає AWS; тобі лишаються доступи, бекап-політики, шифрування.
- **Serverless і сховища** (Lambda, S3, DynamoDB) — серверів ти не бачиш зовсім; тобі — код, дані, права й налаштування.

Але навіть у найбільш керованому сервісі **дані й доступи — завжди твої**. Якщо сховище стало публічним через неправильне налаштування, винен не AWS.

Well-Architected: шість стовпів

AWS зібрав досвід тисяч проєктів у **Well-Architected Framework** — набір питань і принципів, за якими оцінюють будь-яку архітектуру. Він стоїть на **шести стовпах**. Мнемоніка з простого курсу — **CROPS + S**:



C Cost Optimization — чи не переплачуємо?
R Reliability — чи відновимося після збою?
O Operational Excellence — автоматизація, уроки зі збоїв
P Performance Efficiency — правильні ресурси під задачу
S Security — захищені дані й доступи
S Sustainability — без зайвих ресурсів і енергії

мнемоніка: CROPS + S

Well-Architected: храм на шести стовпах — на них спирається кожна правильна архітектура і кожне питання іспиту

- **C**ost Optimization — **оптимізація вартості**: чи не переплачуємо? Плати за використане, прибирай зайве, розподіляй витрати тегами.
- **R**eliability — **надійність**: чи відновимося після збою? Автоматичне відновлення, кілька AZ, горизонтальне масштабування, перевірені бекапи.

- **O**perational Excellence — **операційна досконалість**: чи автоматизоване керування, чи бачимо, що відбувається, і чи вчимося на інцидентах? Інфраструктура як код, малі часті зміни.
- **P**erformance Efficiency — **ефективність продуктивності**: чи правильні ресурси під навантаження? Керовані й serverless-сервіси, близькість до людей, правильний тип бази.
- **S**ecurity — **безпека**: чи захищені дані й доступи? Мінімальні права, шифрування, журнали всіх дій, захист на всіх рівнях.
- **S**ustainability — **сталість**: чи ефективно використовуємо ресурси й енергію? Щільне завантаження, економні процесори, менше зайвих даних.

Іспит поділено на чотири **домени**, і вони майже збігаються зі стовпами: **безпека — 30%**, **відмовостійкість — 26%**, **продуктивність — 24%**, **вартість — 20%**. Операційна досконалість і сталість «розчинені» в питаннях усіх доменів.

📖 **Словничок**

Доступність (availability) — частка часу, коли система працює; «дев'ятки».

SLA («ес-ел-ей») — обіцянка провайдера щодо рівня обслуговування.

Надійність зберігання (durability) — імовірність, що дані не загубляться; S3 — 11 дев'яток.

Висока доступність (HA) — «запасне колесо»: працює далі після короткої паузи.

Відмовостійкість (fault tolerance) — «двомоторний літак»: без жодної перерви.

Аварійне відновлення (DR, «ді-ар») — план на випадок катастрофи регіону.

RPO («ар-пі-оу») — скільки даних можна втратити. **RTO** («ар-ті-оу») — скільки часу можна простояти.

Throughput / IOPS («сру-пут / ай-опс») — обсяг за секунду / операцій диска за секунду.

CIA — конфіденційність, цілісність, доступність.

Least privilege («ліст привілідж») — мінімально потрібні права.

Defense in depth — захист на всіх рівнях.

Shared Responsibility Model — AWS відповідає за безпеку OF the cloud, клієнт — IN the cloud.

Well-Architected — шість стовпів: вартість, надійність, операції, продуктивність, безпека, сталість.

🧩 Міні-тест: перевір себе

? Доступність 99,9% — це скільки простою приблизно за рік?

- **А** 5 хвилин
- **Б** 53 хвилини
- **В** Близько 8,8 години
- **Г** 3,65 дня

Відповідь: В. 0,1% від 8 760 годин року ≈ 8,8 години. Кожна наступна дев'ятка — удесятеро менше.

? S3 обіцяє «11 дев'яток». Про що саме ця обіцянка?

А Про швидкість завантаження

Б Про те, що S3 ніколи не буває недоступний

В Про надійність зберігання: дані майже ніколи не губляться

Г Про ціну

Відповідь: В. 11 дев'яток — durability. Доступність S3 — окрема, менша цифра.

? Двомоторний літак, що продовжує політ без перерви, коли відмовив мотор, — це образ...

А високої доступності

Б відмовостійкості

В аварійного відновлення

Г еластичності

Відповідь: Б. Відмовостійкість — без жодної паузи. Висока доступність — з короткою паузою, як заміна колеса.

? «Можна втратити не більше 5 хвилин замовлень». Про яке число мова?

А RPO

Б RTO

В SLA

Г IOPS

Відповідь: А. RPO — Point: точка в минулому, до якої відкотяться дані. RTO — час простою.

? Хто ставить патчі операційної системи на віртуальному сервері EC2?

- **А** AWS
- **Б** Клієнт
- **В** Ніхто, вони не потрібні
- **Г** Виробник процесора

Відповідь: Б. На EC2 операційна система — твоя «квартира». На керованих базах ОС латає AWS.

? Хто відповідає за фізичну охорону дата-центрів AWS?

- **А** AWS
- **Б** Клієнт
- **В** Інтернет-провайдер клієнта
- **Г** Ніхто

Відповідь: А. Це безпека OF the cloud — зона AWS.

? Що з переліченого — один із шести стовпів Well-Architected?

А Popularity (популярність)

Б Sustainability (сталість)

В Marketing (маркетинг)

Г Design (дизайн)

Відповідь: Б. CROPS + S: Cost, Reliability, Operational Excellence, Performance, Security + Sustainability.

✅ **Головне**

Доступність — «двері відчинені» (дев'ятки: 99,9% ≈ 8,8 год простою на рік); **durability** — «речі цілі» (S3 — 11 дев'яток).

HA — запасне колесо, **fault tolerance** — двомоторний літак, **DR** — план на пожежу регіону; **RPO** — скільки даних втратити, **RTO** — скільки стояти.

Безпека — **CIA**, **мінімальні права**, **захист на всіх рівнях**; «найдешевше» = найдешевше з тих, що виконують **усі** вимоги.

Спільна відповідальність: AWS — безпека **OF** the cloud, клієнт — **IN** the cloud; дані й доступи завжди твої.

Well-Architected — CROPS + S; домени іспиту: безпека 30%, відмовостійкість 26%, продуктивність 24%, вартість 20%.

🪪 Хто ти і що тобі можна: IAM з нуля

👴 Простими словами

IAM — це офісний центр із перепустками. Кожна людина чи програма отримує свою перепустку, а на ній написано, у які двері їй можна. За замовчуванням — нікуди.

🎯 Що ти дізнаєшся

- чим автентифікація відрізняється від авторизації
- навіщо MFA
- хто такі користувачі, групи, ролі й політики в IAM
- як прочитати політику в JSON рядок за рядком
- як AWS вирішує «дозволити чи заборонити» — три кроки напам'ять
- що таке принцип мінімальних прав

🥐 У «Хмаринці» з'явилися працівники: Марко приймає онлайн-замовлення, Соня веде бухгалтерію. Обом потрібен доступ до AWS. Досі Тарас робив усе під root — майстер-ключем. «Дати їм мій пароль?» — питає Оля. «У жодному разі, — лякається Тарас. — Кожному — власна перепустка з потрібними правами. Для цього в AWS є IAM».

Два питання охоронця

На вході в будь-який офісний центр охорона ставить два питання:

1. **«Хто ти?»** — показуєш паспорт чи перепустку. Це **автентифікація** (authentication, «автентикейшн»): перевірка, що ти — це ти.
2. **«Куди тобі можна?»** — охоронець дивиться, які поверхи й кабінети вписано у твою перепустку. Це **авторизація** (authorization, «авторизейшн»): перевірка прав.

Людина може бути автентифікована (охорона знає, хто вона), але не авторизована зайти в серверну. Ці два слова часто плутають — а на іспиті вони мають різні відповіді.

Пароль і MFA: два замки замість одного

Пароль — це «те, що ти **знаєш**». Його можуть підглянути, вгадати, викрасти через фальшивий сайт.

MFA («ем-еф-ей», multi-factor authentication — багатофакторна автентифікація) додає «те, що ти **маєш**»: телефон із застосунком-автентифікатором, що щопівхвилини показує новий код, або фізичний ключ-брелок. Тепер, щоб увійти, треба **і** знати пароль, **і** тримати в руці телефон. Викрадений пароль сам по собі вже нічого не відчинить.

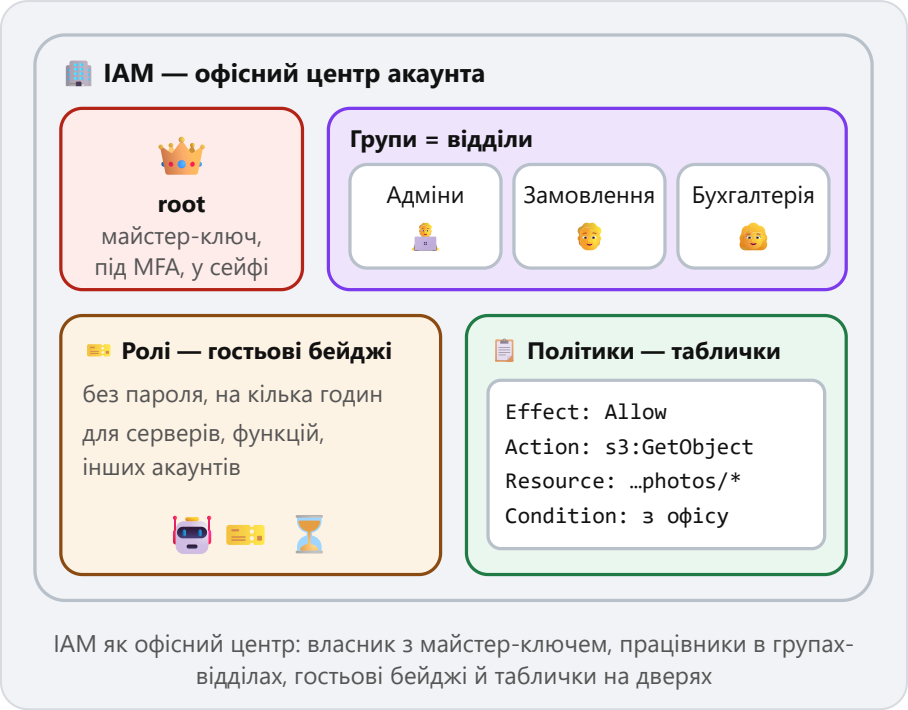
> 💡 Правило, яке варто застосувати з першого дня: MFA — для root і для кожної людини з доступом до консолі.

IAM — офісний центр із перепустками

IAM (читається «ай-ем», Identity and Access Management — керування ідентичністю й доступом) — сервіс AWS, який вирішує, **хто** і **що** може робити в акаунті. Він **глобальний** (один на весь акаунт, а не на регіон) і **безкоштовний**.

Образ з простого курсу — **великий офісний центр**:

- **Root user** — власник будівлі з майстер-ключем. Вмикає MFA, ховає ключ і майже ніколи ним не користується.
- **IAM users** (користувачі) — працівники з особистими перепустками. У кожного — свій логін і пароль для консолі, а для програм — ключі доступу.
- **Groups** (групи) — відділи. Права дають відділу, а працівники отримують їх, бо в ньому числяться. Групи не вкладаються одна в одну: «відділу у відділі» немає.
- **Roles** (ролі) — тимчасові гостьові бейджі, які охорона видає на кілька годин. Власного пароля в ролі немає. Детально — у наступному розділі.
- **Policies** (політики) — **таблички на дверях**: хто, у які двері й за яких умов може заходити.



IAM як офісний центр: власник з майстер-ключем, працівники в групах-відділах, гостьові бейджі й таблички на дверях

Політика — табличка на дверях

Політика — це невеликий документ у форматі JSON (пам'ятаєш його з розділу про вебсайти?). Ось політика, яка дозволяє читати й завантажувати фото в сховище пекарні:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:PutObject"],
      "Resource": "arn:aws:s3:::khmarynka-photos/*"
    }
  ]
}
```

Розберемо рядок за рядком:

- `"Version": "2012-10-17"` — версія «мови» політик. Завжди саме цей рядок, просто запам'ятай.
- `"Statement"` — список правил. Їх може бути кілька.
- `"Effect"` — **що робити**: `Allow` (дозволити) або `Deny` (заборонити).
- `"Action"` — **які дії**: назва сервісу, двокрапка, назва дії. `s3:GetObject` — «взяти файл із S3», `s3:PutObject` — «покласти файл».
- `"Resource"` — **над чим**: ARN ресурсу. Зірочка `*` означає «будь-що»: `khmarynka-photos/*` — усі файли в сховищі.
- `"Condition"` (необов'язково) — **за яких умов**. Наприклад, лише з адреси офісу або лише якщо людина увійшла з MFA:

```
"Condition": { "IpAddress": { "aws:SourceIp": "203.0.113.0/24" } }
```

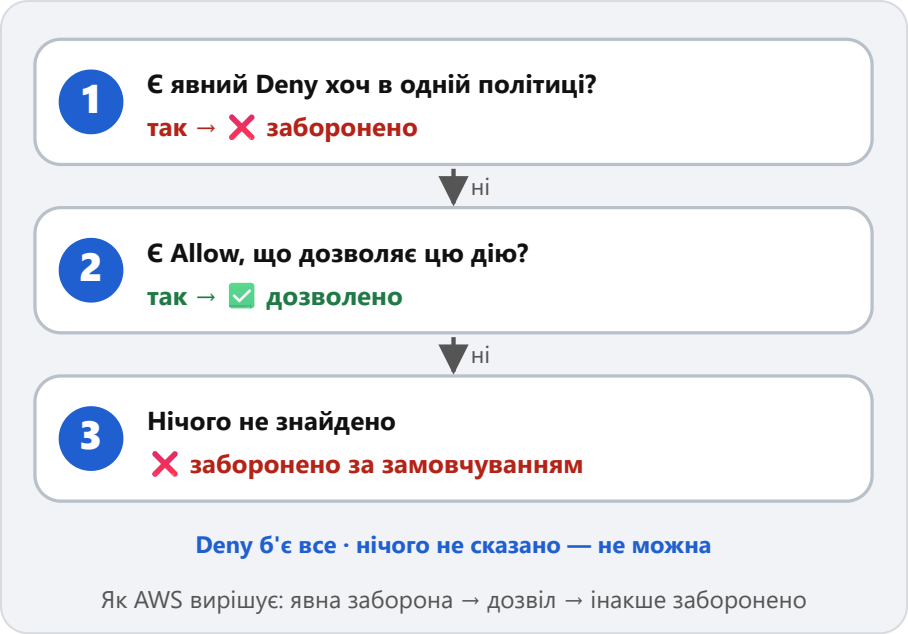
Бачиш знайомий запис `/24`? Це «квартал» адрес офісу з розділу про IP. Усе вивчене раніше тут складається докупи.

Ще одне поле — `"Principal"` (**хто**) — з'являється в політиках, які чіпляють не до людини, а до ресурсу (наприклад, до сховища). Про це — у наступному розділі.

Як AWS вирішує: три кроки напам'ять

Коли будь-хто робить будь-яку дію, AWS збирає всі політики, що стосуються цього запиту, і перевіряє їх так:

1. **Є явна заборона (`Deny`)?** Якщо хоч в одній політиці написано «заборонити» — **заборонено**. Крапка. Deny перемагає будь-які дозволи.
2. **Є дозвіл (`Allow`)?** Якщо заборони немає, AWS шукає дозвіл. Є — **дозволено**.
3. **Немає нічого?** Тоді **заборонено**. Це «неявна заборона»: за замовчуванням не можна **нічого**.



Звідси два правила, які розв'язують десятки питань іспиту:

- **Deny б'є все.**
- **Нічого не сказано — значить, не можна.** Новий користувач без політик не може зробити в AWS абсолютно нічого.

(У великих компаніях до цієї перевірки додаються ще «стелі» — правила організації й межі прав. Ми додамо їх у розділі про багато акаунтів, але три кроки лишаються тими самими.)

Принцип мінімальних прав

Least privilege («ліст привілідж») — кожен отримує **рівно ті права, що потрібні для роботи**, і не більше. Марко бачить замовлення, але не видаляє сервери. Соня бачить рахунки, але не чіпає сайт.

Практично це виглядає так:

- починати з нуля прав і додавати потрібне, а не «дати все, потім заберемо»;
- давати права групам, а не кожній людині окремо;
- регулярно прибирати зайве.

Допомагають два інструменти:

- **IAM Access Analyzer** — знаходить ресурси, до яких мають доступ **чужі** акаунти, і вміє згенерувати політику з мінімальними правами на основі того, що людина чи програма справді робила.
- **Access Advisor** («last accessed») — показує, якими сервісами користувач чи роль **реально** користувалися. Рік не заходив в S3 — право на S3 можна забрати.

Готові й власні політики

- **AWS managed** — готові політики від AWS: `ReadOnlyAccess` (лише дивитися), `AdministratorAccess` (усе), `AmazonS3ReadOnlyAccess` тощо. Зручно, але часто ширше, ніж треба.

- **Customer managed** — власні політики, які ти пишеш і чіпляєш до багатьох груп чи ролей.
- **Inline** — політика, «вшита» в одного конкретного користувача чи роль. Використовують рідко.

За тим, **куди** чіпляють політику, розрізняють:

- **identity-based** — до людини, групи чи ролі: «що **цей хтось** може робити»;
- **resource-based** — до самого ресурсу (сховища, черги, ключа): «**хто** може працювати **зі мною**».

Ключі доступу: перепустка, що не закінчується

Щоб програма чи командний рядок працювали від імені користувача, йому можна видати **access keys** — пару «Access Key ID + Secret Access Key». Це **довгострокові** облікові дані: вони діють, доки їх не видалять.

Звідси ризик: ключ, що потрапив у код, у спільну папку чи на форум, — як **перепустка, приклеєна скотчем до дверей**. Її знайдуть. Тому для програм, що працюють в AWS, ключі не використовують узагалі — для них є **ролі**. Це тема наступного розділу.

🥐 Тарас навів лад: root тепер під MFA і лежить «у сейфі». Створено групи «Адміни» (Тарас), «Замовлення» (Марко) і «Бухгалтерія» (Соня), у кожної — своя політика. Кожен входить у консоль під власним іменем, з MFA. А якщо Марко звільниться, його просто видалять з групи — і ні з ким не доведеться міняти паролі.

📖 Словничок

Автентифікація — перевірка «хто ти». **Авторизація** — перевірка «що тобі можна».

MFA («ем-еф-ей») — другий «замок»: пароль + код з телефона чи ключ-брелок.

IAM («ай-ем») — глобальний безкоштовний сервіс керування доступом.

IAM user / group / role — працівник з перепусткою / відділ / тимчасовий бейдж.

Політика (policy) — JSON-документ: Effect, Action, Resource, Condition (і Principal у політиках ресурсів).

Allow / Deny — дозволити / заборонити. Явний Deny перемагає все.

Неявна заборона — усе, що не дозволено явно, заборонено.

Least privilege («ліст привілідж») — мінімально потрібні права.

AWS managed / customer managed / inline policy — готова від AWS / власна багаторазова / вшита в одного.

Identity-based / resource-based policy — прив'язана до «хто» / до ресурсу.

Access keys — довгострокові ключі для програм і CLI; ризиковані, краще ролі.

IAM Access Analyzer / Access Advisor — пошук зайвого доступу / які сервіси реально використовуються.

🧩 Міні-тест: перевір себе

? Що таке автентифікація?

- **А** Перевірка, хто ти, — наприклад, пароль і код MFA
- **Б** Перевірка, до яких сервісів тобі можна
- **В** Шифрування даних
- **Г** Створення нового акаунта

Відповідь: А. Автентифікація — «покажи паспорт». Авторизація — «куди тобі можна за цією перепусткою».

? Двоє нових працівників бухгалтерії мають отримати однакові права. Як зробити правильно?

А Дати обом пароль від root

Б Написати кожному окрему однакову політику

В Створити групу «Бухгалтерія» з потрібною політикою і додати їх у групу

Г Створити їм один спільний обліковий запис

Відповідь: В. Права дають групі-відділу. Прийшов новий бухгалтер — додали в групу; пішов — прибрали.

? В одній політиці користувача написано Allow на видалення файлів, в іншій — явний Deny на те саме. Що буде?

А Дозволено, бо Allow записано першим

Б Заборонено: явний Deny завжди перемагає

В AWS запитає адміністратора

Г Буде помилка політики

Відповідь: Б. Крок 1 перевірки — пошук явного Deny. Знайшли — заборонено, хоч би скільки було дозволів.

? Користувач має лише політику на читання S3. Чи може він запустити сервер EC2?

А Так, бо заборони на EC2 немає

Б Ні: усе, що не дозволено явно, заборонено за замовчуванням

В Так, якщо має MFA

Г Лише в регіоні us-east-1

Відповідь: Б. Неявна заборона: без Allow дії немає.

? Що означає "Resource": "arn:aws:s3:::khmarynka-photos/*"?

А Лише саме сховище, без файлів

Б Усі об'єкти (файли) у сховищі khmarynka-photos

В Усі сховища в акаунті

Г Файли в усіх акаунтах світу

Відповідь: Б. Зірочка після слеша — «будь-який об'єкт усередині цього сховища».

? Чим root відрізняється від звичайного IAM-користувача?

А Нічим

Б Root має повну владу над акаунтом; його захищають MFA і не використовують щодня

В Root може лише дивитися рахунки

Г Root створюється для кожного працівника

Відповідь: Б. Root — майстер-ключ. Для щоденної роботи — окремі користувачі з мінімальними правами.

✅ **Головне**

Автентифікація — хто ти; **авторизація** — що тобі можна. **MFA** — другий замок.

IAM — глобальний безкоштовний «офісний центр»: root (майстер-ключ під MFA), users (працівники), groups (відділи), roles (тимчасові бейджі), policies (таблички на дверях).

Політика: **Effect, Action, Resource, Condition** (+ Principal у політиках ресурсів); `*` — будь-що.

Перевірка: **явний Deny → Allow → інакше заборонено**. Deny б'є все; нічого не сказано — не можна.

Least privilege: права групам, мінімально потрібні; Access Analyzer і Access Advisor допомагають прибрати зайве. Довгі **access keys** — ризик.

🎫 Ролі й тимчасові ключі: безпечніше за паролі

👴 Простими словами

Роль — це тимчасова перепустка, яку видають лише на час роботи. Вона сама «згорає», тож її не страшно загубити, на відміну від постійного пароля чи ключа.

🎯 Що ти дізнаєшся

- чому довгі ключі доступу небезпечні
- що таке роль і тимчасові облікові дані від STS
- як сервери, функції й контейнери отримують права без ключів
- що таке trust policy і як дати доступ іншому акаунту
- що таке resource-based policy, ExternalId, permission boundary і ABAC

🥐 Сайт «Хмаринки» має сам завантажувати фото тортів у сховище. Тарас по-швидкому вписав ключ доступу просто в код на сервері — і випадково виклав цей код у відкритий репозиторій. За годину на пошту прийшов лист від AWS: «Ваш ключ знайдено у відкритому доступі». Тарас зблід. «Отже, ключі в коді — погана ідея, — каже він Олі. — Зараз зробимо правильно: ролями».

Чим погані довгі ключі

Ключ доступу (access key) — це перепустка, яка **ніколи не закінчується**. Вона працює, доки хтось не помітить і не видалить її. А помічають зазвичай тоді, коли на рахунку вже з'явилися чужі сервери для майнингу криптовалют.

Потрібна перепустка, яка:

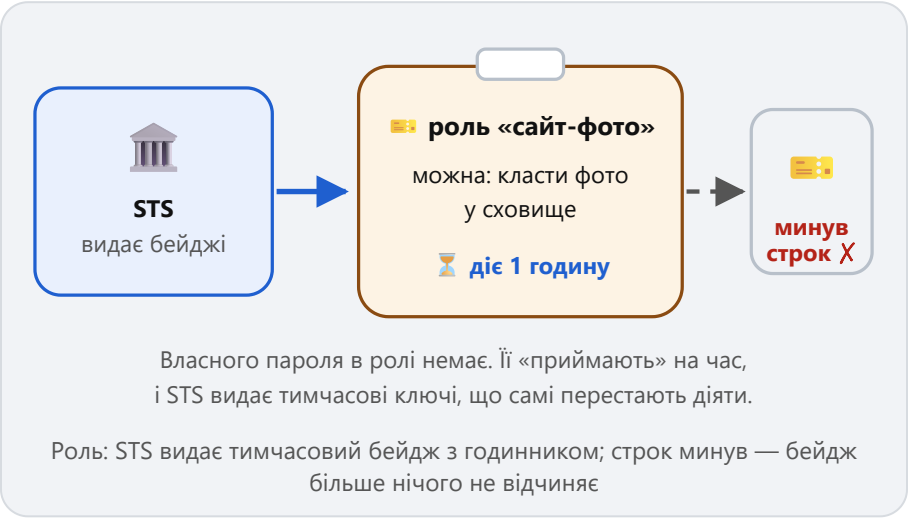
- видається лише тому, кому можна;
- **сама перестає діяти** через короткий час;
- оновлюється автоматично, без людини.

Саме це і є роль.

Роль — тимчасовий гостьовий бейдж

Роль (IAM role) — це набір прав **без власного пароля чи ключів**. Роль ніхто не «має» постійно — її **приймають** (assume, «ассьюм») на певний час.

Коли хтось приймає роль, спеціальна служба AWS — **STS** («ес-ті-ес», Security Token Service — служба токенів безпеки) — видає йому **тимчасові облікові дані**: ключ, секрет і токен, що діють від кількох хвилин до кількох годин. Минув строк — бейдж перетворюється на шматок пластику.



Навіть якщо тимчасовий ключ хтось підгляне, користі від нього мало: скоро він стане недійсним.

Ролі для серверів, функцій і контейнерів

Найчастіше роль приймають не люди, а **сервіси AWS**:

- **EC2** — роль чіпляють до сервера через **instance profile** («інстанс профайл»). Програма на сервері автоматично отримує тимчасові ключі й сама їх оновлює. Образ з простого курсу — **бейдж, прикріплений до робота**: він сам продовжується кожні кілька годин, і нікому не треба ховати ключ.
- **Lambda** — у кожної функції є **execution role** («роль виконання»): що функції дозволено робити.
- **ECS** (контейнери) — **task role**: права саме для конкретного контейнера.

🎯 На іспиті: «застосунку на EC2 потрібен доступ до S3» → **IAM role через instance profile**. Варіанти «зберегти ключі в коді», «покласти ключі у файл на сервері» чи «у змінні середовища» — неправильні.

🔍 **Для допитливих: Як програма на сервері бачить тимчасові ключі?**

Кожен сервер EC2 може звернутися до спеціальної внутрішньої адреси — **сервісу метаданих інстансу** (instance metadata). Там лежать відомості про сам сервер і, якщо до нього прикріплено роль, свіжі

тимчасові ключі. Бібліотеки AWS (SDK) самі їх звідти беруть і оновлюють. Безпечна версія цього сервісу — **IMDSv2**: вона вимагає спершу отримати токен сесії, що захищає від деяких хитрих атак. На іспиті іноді питають: «як безпечніше звертатися до метаданих» → IMDSv2.

Дві політики ролі: хто може взяти і що відчиняє

У кожної ролі є дві різні політики:

- **Trust policy** («траст полісі», політика довіри) — **хто може взяти бейдж**: «сервіс EC2», «сервіс Lambda», «користувачі акаунта 111122223333».
- **Permissions policy** (політика дозволів) — **що бейдж відчиняє**: «читати й писати в сховище фото».



У ролі дві політики: довіри (хто може прийняти роль) і дозволів (що вона відчиняє)

Ось як виглядає trust policy, що дозволяє серверам EC2 приймати роль:

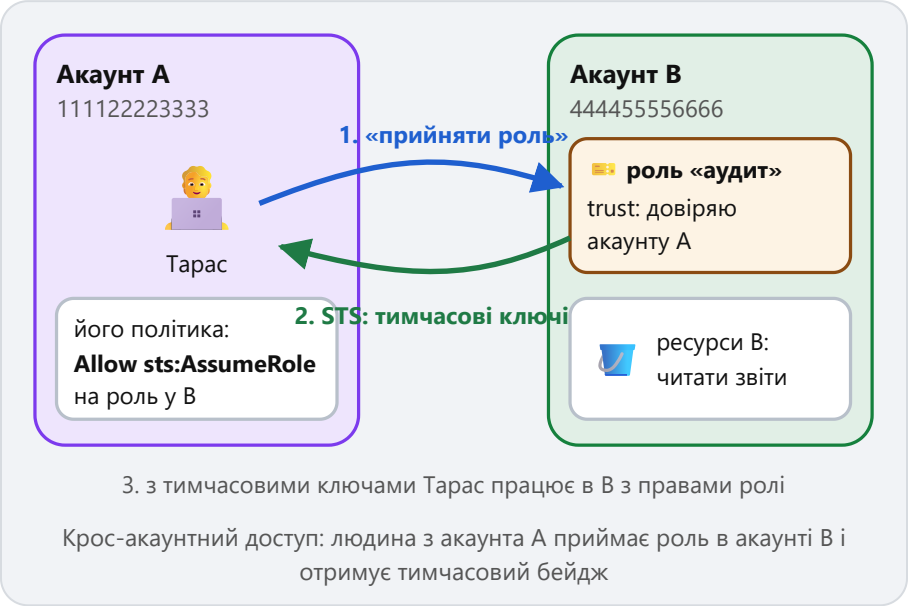
```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": { "Service": "ec2.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }]
}
```

`Principal` — **хто**; `sts:AssumeRole` — «прийняти роль».

Доступ до іншого акаунта

У компанії багато акаунтів, і людям з одного треба часом попрацювати в іншому. Ключі тут теж не потрібні:

1. В **акаунті B** створюють роль, у trust policy якої написано: «довіряю акаунту A».
2. Людина з **акаунта A** (якій її власна політика дозволяє `sts:AssumeRole` на цю роль) «перемикається» в роль.
3. STS видає тимчасові ключі, і людина працює в акаунті B з правами ролі.



Крос-акаунтний доступ: людина з акаунта A приймає роль в акаунті B і отримує тимчасовий бейдж

ExternalId: захист від «плутанини заступника»

Буває, що доступ до твого акаунта потрібен **сторонній компанії** — наприклад, сервісу, який аналізує рахунки AWS. Ти створюєш для неї роль.

Але уяви: цей сервіс обслуговує тисячі клієнтів. Хитрий клієнт X може вказати сервісу **чужу** роль — твою — і

сервіс, сам того не бажаючи, полізе у твій акаунт від імені X. Це називається **confused deputy** («плутанина заступника»).

Захист простий: у trust policy твоєї ролі додають умову `sts:ExternalId` — секретну «кодову фразу», яку сервіс видав саме тобі. Без правильної фрази роль не прийняти, а чужу фразу клієнт X не знає.

> 🎯 «Сторонній SaaS-вендор отримує доступ через роль, треба захист від confused deputy» → **trust policy з умовою sts:ExternalId**.

Resource-based policy: табличка на самому ресурсі

Політику можна причепити не лише до людини чи ролі, а й **до ресурсу**. Найвідоміший приклад — **bucket policy** (політика сховища S3). Інші — політики черг SQS, тем SNS, ключів KMS, функцій Lambda. У таких політиках обов'язково є поле `Principal` — **кому** дозволено.

Наприклад, дозволити акаунту партнера `111122223333` читати файли:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::111122223333:root" },
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::khmarynka-photos/*"
  }]
}
```

⚠️ Для доступу **з іншого акаунта** через політику ресурсу потрібен Allow **з обох боків**: у політиці ресурсу («пускаю акаунт A») **і** в IAM-політиці людини з акаунта A («мені можна туди ходити»). Бракує одного — AccessDenied.

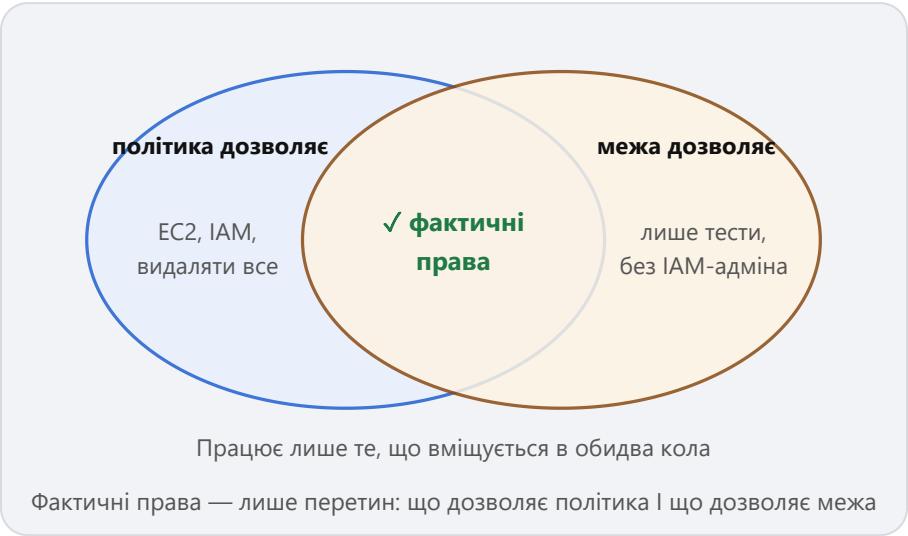
Корисна умова для політик ресурсів — `aws:PrincipalOrgID`: «пускати лише акаунти моєї організації», не перелічуючи їх усіх.

Permission boundary: стеля для конкретної людини

Уяви: розробникам треба дозволити **самим створювати ролі** для своїх програм. Небезпечно: розробник може створити роль з правами адміністратора і взяти її собі.

Permission boundary («пермішн баундері», межа прав) — це **стеля** для користувача чи ролі. Навіть якщо політика дає більше, фактичні права — лише те, що вміщується **і** в політику, **і** в межу. Розробникам

дозволяють створювати ролі **лише з цією межею** — і вище стелі ніхто не стрибне.



Фактичні права — лише перетин: що дозволяє політика І що дозволяє межа

ABAC: права за ярликами

Замість сотні політик «Марку — ці сервери, Соні — ті» можна написати одну: «людина може керувати ресурсами, у яких тег `project` такий самий, як у неї». Це **ABAC** («ей-бі-ей-сі», attribute-based access control) — доступ за атрибутами (тегами). Новий проєкт — новий тег, а політика та сама.

Ще кілька інструментів

- **Федерація** — вхід через зовнішню «систему паспортів» (корпоративний каталог, Google) без окремих IAM-користувачів: STS видає тимчасові ключі за підтвердженням від неї (протоколи SAML 2.0 чи OIDC). Детальніше — у наступному розділі.
- **IAM Roles Anywhere** — тимчасові ключі AWS для серверів **поза AWS** (в офісі) за цифровим сертифікатом, без довгих ключів.
- **Credential report** — звіт про всіх користувачів: у кого є пароль, MFA, коли востаннє змінювали ключі.

🥐 Тарас видалив злощасний ключ, створив роль «сайт-фото» з правом писати лише в сховище фото й прикріпив її до сервера. Код тепер узагалі не знає жодних ключів. «А якщо сервер зламають?» — питає Оля. «Тоді зловмисник отримає лише право класти фото, і то на годину», — відповідає Тарас.

📖 **Словничок**

Роль (IAM role) — набір прав без пароля; її тимчасово приймають (assume).

STS («ес-ті-ес») — служба, що видає тимчасові облікові дані.

Instance profile — «кріплення» ролі до сервера EC2.

Execution role / task role — роль функції Lambda / контейнера ECS.

IMDSv2 — безпечна версія сервісу метаданих сервера.

Trust policy — хто може прийняти роль. **Permissions policy** — що роль дозволяє.

sts:AssumeRole — дія «прийняти роль».

Confused deputy — «плутанина заступника»; захист — умова **sts:ExternalId**.

Resource-based policy — політика на ресурсі (bucket policy тощо) з полем Principal.

aws:PrincipalOrgID — умова «лише акаунти моєї організації».

Permission boundary — стеля прав для користувача чи ролі.

ABAC («ей-бі-ей-сі») — доступ за тегами.

Федерація — вхід через зовнішню систему ідентичності (SAML, OIDC).

IAM Roles Anywhere — тимчасові ключі для серверів поза AWS за сертифікатом.

🧩 Міні-тест: перевір себе

? Застосунок на EC2 має завантажувати фото в S3. Як дати доступ правильно?

А. Вписати ключ доступу в код

Б. Покласти ключі у файл на сервері

В. Прикріпити до сервера IAM-роль через instance profile

Г. Дати серверу пароль root

Відповідь: В. Роль видає серверу тимчасові ключі, які оновлюються самі. Жодних ключів у коді.

? Скільки діють облікові дані, які STS видає для ролі?

- **А** Вічно
- **Б** Обмежений час — від хвилин до годин, потім стають недійсними
- **В** Рівно рік
- **Г** Доки хтось не видалить роль

Відповідь: Б. Тимчасовість — головна перевага. Навіть підглянутий ключ скоро нічого не відчинить.

? Що описує trust policy ролі?

- **А** Хто може прийняти цю роль
- **Б** Які файли роль може читати
- **В** Скільки коштує роль
- **Г** У якому регіоні роль працює

Відповідь: А. Trust — «хто може взяти бейдж». Що бейдж відчиняє — у permissions policy.

? Сторонній сервіс отримує доступ до твого акаунта через роль. Як захиститися від «плутанини заступника»?

А Дати сервісу пароль root

Б Додати в trust policy умову sts:ExternalId

В Вимкнути MFA

Г Створити для сервісу IAM-користувача з ключами

Відповідь: Б. ExternalId — кодова фраза, яку знає лише цей сервіс для твого акаунта. Чужий клієнт роль не прийме.

? Розробникам дозволили створювати ролі, але так, щоб вони не могли дати собі більше прав, ніж визначено. Що використати?

- **А** SCP
- **Б** Permission boundary
- **В** Resource-based policy
- **Г** MFA

Відповідь: Б. Межа прав — стеля: фактичні права = перетин політики й межі.

? Користувач з акаунта A читає файли зі сховища в акаунті B через bucket policy. Що потрібно?

- **А** Лише bucket policy в акаунті B
- **Б** Лише IAM-політика в акаунті A
- **В** Allow і в bucket policy, і в IAM-політиці користувача
- **Г** Пароль від root акаунта B

Відповідь: В. Для крос-акаунтного доступу через політику ресурсу потрібні дозволи з обох боків.

✅ Головне

Довгі **access keys** — перепустка, що не закінчується; для програм в AWS їх замінюють **ролі**.

Роль приймають на час: **STS** видає **тимчасові** ключі. EC2 — **instance profile**, Lambda — **execution role**, ECS — **task role**.

У ролі дві політики: **trust** (хто може взяти) і **permissions** (що відчиняє).

Інший акаунт — роль із довірою; стороння компанія — ще й **sts:ExternalId**. Політика ресурсу (bucket policy) з іншого акаунта — Allow **з обох боків**; «лише моя організація» — **aws:PrincipalOrgID**.

Permission boundary — стеля прав; **ABAC** — доступ за тегами; **Roles Anywhere** — для серверів поза AWS.

🏢 Багато акаунтів і багато людей: Organizations, SCP, вхід для працівників і клієнтів

👴 Простими словами

Коли акаунтів багато, їх об'єднують в Organizations — як філії однієї компанії. SCP — правила від головного офісу, які забороняють філіям певні дії, хоч би що там дозволили на місці.

🎯 Що ти дізнаєшся

- навіщо компанії десятки акаунтів
- як AWS Organizations об'єднує їх і що дає спільний рахунок
- як працюють SCP — «стеля» для цілих акаунтів
- що робить Control Tower
- як працівники входять одним паролем в усі акаунти (Identity Center), а клієнти застосунку — через Cognito
- що таке Directory Service і RAM

🥐 «Хмаринка» росте. Тарас хоче окремий акаунт для робочого сайту, окремий — для тестів, окремий — для журналів безпеки, щоб помилка в тестах ніколи не зачепила сайт. Оля ставить дві умови: «Один рахунок за все і правило, яке ніхто не зможе обійти: журнали безпеки не вимикати». А ще клієнти мобільного застосунку мають реєструватися й входити через Google. «Для всього цього в AWS є окремі сервіси», — каже Тарас.

Навіщо багато акаунтів

Один акаунт на все — як одна кімната для всієї компанії: тісно, і будь-яка помилка зачіпає всіх. Окремі акаунти дають:

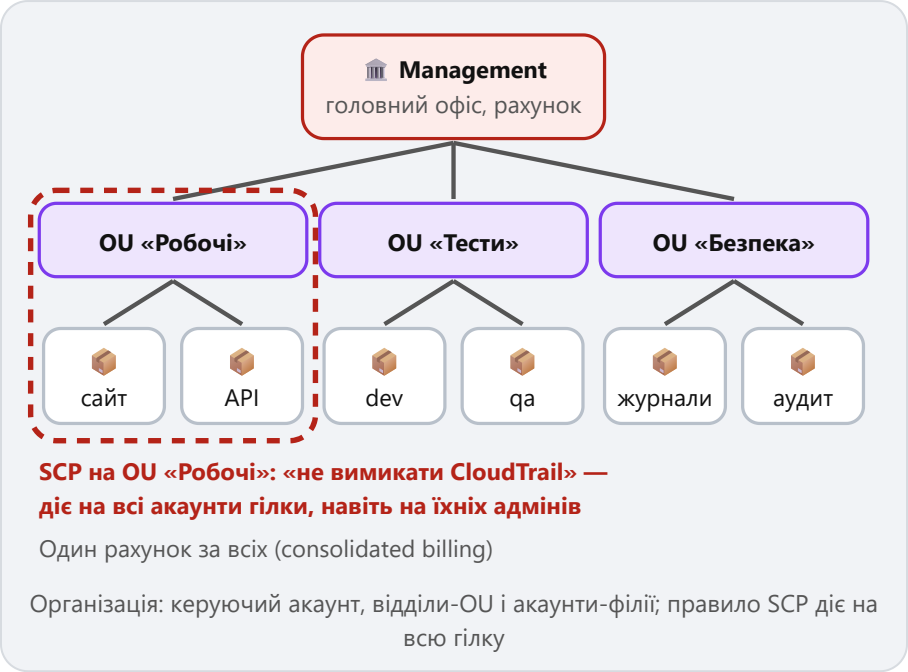
- **ізоляцію** — помилка чи злам у тестовому акаунті не зачепить робочий сайт (фахівці кажуть «менший радіус ураження», blast radius);
- **окремі права** — розробникам можна все в тестах і майже нічого в робочому акаунті;
- **зрозумілі витрати** — видно, скільки коштує кожен проєкт;

- **окремі ліміти** — квоти одного акаунта не заважають іншому.

AWS Organizations: головний офіс мережі

AWS Organizations — сервіс, що об'єднує багато акаунтів під одним «головним офісом»:

- **Management account** («менеджмент акаунт», керуючий) — головний офіс: створює організацію, запрошує чи створює акаунти, встановлює правила.
- **Member accounts** — філії.
- **OU** («оу-ю», organizational units — організаційні одиниці) — **відділи**, у які групують акаунти: «Робочі», «Тести», «Безпека». OU можна вкладати одна в одну.



Організація: керуючий акаунт, відділи-OU і акаунти-філії; правило SCP діє на всю гілку

Приємний бонус — **consolidated billing** (об'єднаний рахунок): один рахунок за всі акаунти, об'ємні знижки (що більше спожито разом, то дешевше) і спільне використання знижок за довгострокові зобов'язання (до них дійдемо в розділі про оплату серверів).

SCP: стеля для цілих філій

SCP («ес-сі-пі», service control policy — політика контролю сервісів) — правила головного офісу для філій. Їх чіпляють до OU чи акаунта, і вони діють на **всіх** у ньому — навіть на адміністраторів і навіть на root філії. Головне про SCP — три речі:

1. **SCP — це стеля, а не ключ.** Вона **нічого не дозволяє сама**, лише обмежує максимум. Щоб людина щось могла, їй усе одно потрібен Allow у звичайній IAM-політиці. Але вище стелі SCP не стрибне ніхто.
2. SCP **не діє на management account** — головний офіс правилами для філій себе не обмежує.
3. Типові приклади: «заборонити вимикати журнали CloudTrail», «заборонити працювати поза дозволеними регіонами» (для вимог про дані в ЄС), «заборонити виходити з організації».

Тепер повна перевірка доступу виглядає так — фільтри один за одним:



- Є явний **Deny** будь-де — заборонено.
- **SCP** організації не дозволяє дію — заборонено.
- **Permission boundary** (якщо є) не дозволяє — заборонено.
- Політика людини чи ресурсу дає **Allow** — дозволено.
- Інакше — заборонено.

🆕 З 2024 року є ще **RCP** (resource control policies) — «стеля» для ресурсів в акаунтах організації, наприклад: «до наших сховищ S3 не може звертатися ніхто поза організацією». Є також політики тегів і політики бекапів.

Control Tower: готова «правильна» організація

Налаштувати організацію за всіма найкращими практиками вручну — довго. **AWS Control Tower** робить це за тебе й будує **landing zone** («лендінг зоун» — «посадкову зону»):

- окремі акаунти для журналів і аудиту;
- централізовані журнали з усіх акаунтів;
- **guardrails** («гардрейлз», огорожі; нині їх звуть controls): **превентивні** (забороняють, через SCP), **детективні** (помічають порушення, через правила AWS Config), **проактивні** (перевіряють шаблони до створення ресурсів);
- **Account Factory** — «фабрика» нових акаунтів, однаково налаштованих з першої хвилини.

🎯 «Швидко налаштувати багато акаунтів за найкращими практиками, з огорожами й централізованими журналами» → **Control Tower**.

Працівники: один вхід в усе — IAM Identity Center

Створювати кожному працівнику окремого IAM-користувача в кожному з двадцяти акаунтів — жах. **IAM Identity Center** (колишня назва — AWS SSO) дає **одну перепустку працівника на всі будівлі компанії**:

- людина входить один раз (single sign-on, «сингл сайн-он») і бачить усі акаунти й бізнес-застосунки, куди їй можна;
- права задають **permission sets** («набори прав»): «адмін у тестах», «лише перегляд у робочому»;
- користувачі можуть жити у вбудованому каталозі Identity Center, у корпоративному **Active Directory** або в зовнішній системі — **Okta**, **Microsoft Entra ID** — через стандарти **SAML 2.0** і **SCIM**.

Active Directory в AWS: Directory Service

Багато компаній тримають працівників і комп'ютери в **Active Directory** (AD) — корпоративному «довіднику» від Microsoft. **AWS Directory Service** пропонує три варіанти:

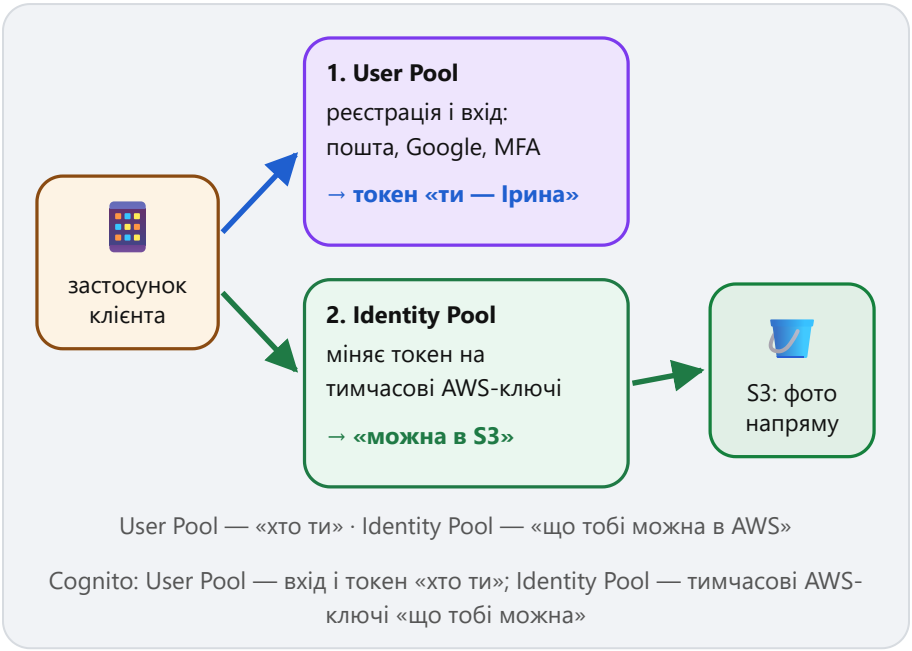
- **AWS Managed Microsoft AD** — справжній AD у хмарі; можна налаштувати **довіру (trust)** з AD в офісі.
- **AD Connector** — «перехідник»: запити до офісного AD, нічого не зберігаючи в AWS.

- **Simple AD** — дешевий спрощений сумісний каталог для невеликих середовищ, без довіри з офісним AD.

Клієнти застосунку: Amazon Cognito

Працівники — це одне, а **клієнти** твого застосунку — зовсім інше: їх тисячі, вони реєструються самі, входять через Google чи Facebook. Для них є **Amazon Cognito** («когніто»). Він має дві частини — і їх постійно плутають:

- **User Pool** («юзер пул», пул користувачів) — **реєстрація і вхід**: email і пароль, вхід через соцмережі чи корпоративні системи, підтвердження пошти, MFA. Після входу видає застосунку **токени** (JWT) — «ти — це пані Ірина». Вміє працювати разом з балансувальником ALB та API Gateway. Це про **хто ти**.
- **Identity Pool** («айдентіті пул») — **тимчасові AWS-ключі** для користувачів застосунку (навіть неавторизованих гостей), щоб застосунок на телефоні міг **напряму** працювати з сервісами AWS — наприклад, завантажити фото в S3. Це про **що тобі можна в AWS**.



⚠️ Не плутай: **Identity Center** — для **працівників** (вхід в акаунти AWS). **Cognito** — для **клієнтів** твого застосунку. **User Pool** — вхід, **Identity Pool** — ключі до AWS.

Ділитися ресурсами між акаунтами: RAM

Іноді ресурс зручно створити один раз і дати ним користуватися кільком акаунтам — наприклад, спільну мережу. Для цього є **AWS RAM** («рем», Resource Access Manager): він «ділиться» підмережами (так звана VPC sharing), мережевими хабами, правилами DNS та іншим між акаунтами організації.

Хто де входить — одна таблиця

Хто	Куди	Сервіс
Працівники	в акаунти AWS і бізнес-застосунки	IAM Identity Center
Працівники з корпоративним AD	те саме через AD	Identity Center + Directory Service
Клієнти застосунку	реєстрація і вхід	Cognito User Pool
Клієнти застосунку	напряму до S3, DynamoDB	Cognito Identity Pool
Програми в AWS	до інших сервісів	IAM roles
Сервери поза AWS	до сервісів AWS	IAM Roles Anywhere

📖 Словничок

AWS Organizations — об'єднання акаунтів під «головним офісом».

Management account / member account — керуючий акаунт / акаунт-філія.

OU («оу-ю») — організаційна одиниця, «відділ» з акаунтів.

Consolidated billing — один рахунок за всі акаунти й спільні знижки.

SCP («ес-сі-пі») — стеля прав для OU чи акаунта; сама нічого не дозволяє; не діє на management account.

RCP — стеля для ресурсів організації.

Control Tower — готова landing zone з guardrails, журналами й Account Factory.

IAM Identity Center (колишній AWS SSO) — єдиний вхід працівників; permission sets.

SAML 2.0 / SCIM — стандарти входу / синхронізації користувачів із зовнішніх систем.

Directory Service — Managed Microsoft AD, AD Connector, Simple AD.

Cognito User Pool — реєстрація й вхід клієнтів застосунку (токени JWT).

Cognito Identity Pool — тимчасові AWS-ключі для клієнтів застосунку.

AWS RAM («рем») — спільне використання ресурсів між акаунтами.

🧩 **Міні-тест: перевір себе**

? Що дає consolidated billing в AWS Organizations?

А Один рахунок за всі акаунти організації й спільні об'ємні знижки

Б Безкоштовні сервери

В Окремий пароль для кожного акаунта

Г Автоматичне шифрування

Відповідь: А. Об'єднаний рахунок: зручно платити, а спільне споживання дає більші знижки.

? SCP забороняє вимикати CloudTrail в OU «Робочі». Адміністратор акаунта в цій OU має політику AdministratorAccess. Чи зможе він вимкнути CloudTrail?

А Так, бо він адміністратор

Б Ні: SCP — стеля, вище якої не стрибне ніхто в цьому акаунті

В Так, якщо увімкне MFA

Г Лише вночі

Відповідь: Б. SCP обмежує максимум прав для всіх в акаунті, навіть для адміністраторів і root філії.

? На який акаунт SCP не діє?

А. На тестові акаунти

Б. На management account

В. На акаунти в ЄС

Г. На всі, SCP — лише рекомендація

Відповідь: Б. Головний офіс правилами для філій себе не обмежує. Тому в management account не запускають робочі навантаження.

? Працівники мають одним входом заходити у 20 акаунтів AWS через корпоративний Microsoft Entra ID. Що обрати?

А Cognito User Pool

Б Створити IAM-користувачів у кожному акаунті

В IAM Identity Center

Г Simple AD

Відповідь: В. Identity Center — єдиний вхід працівників в усі акаунти, з підключенням зовнішньої системи через SAML.

? Клієнти мобільного застосунку мають реєструватися й входити через Google. Що обрати?

- **А** Cognito User Pool
- **Б** IAM Identity Center
- **В** IAM-користувача для кожного клієнта
- **Г** AD Connector

Відповідь: А. User Pool — реєстрація і вхід клієнтів застосунку, зокрема через соцмережі.

? Застосунок на телефоні має завантажувати фото користувачів напряму в S3. Що дасть йому тимчасові AWS-ключі?

- **А** Cognito User Pool
- **Б** Cognito Identity Pool
- **В** Root-ключі, вбудовані в застосунок
- **Г** SCP

Відповідь: Б. Identity Pool видає користувачам застосунку тимчасові облікові дані AWS з обмеженими правами.

✅ **Головне**

Багато акаунтів = ізоляція, окремі права, зрозумілі витрати. **Organizations**: management account, member accounts, **OU**, **consolidated billing**.

SCP — стеля для OU/акаунтів: **нічого не дозволяє**, лише обмежує; **не діє на management account**. Повна перевірка: Deny → SCP → boundary → Allow → інакше заборонено.

Control Tower — готова landing zone з guardrails і журналами.

Працівники → **IAM Identity Center** (+ Directory Service для AD). Клієнти → **Cognito**: **User Pool** — вхід, **Identity Pool** — тимчасові ключі до AWS.

RAM — ділитися ресурсами (підмережами, хабами) між акаунтами.

🔐 Шифрування з нуля: замки, ключі й конверти

👴 Простими словами

Шифрування — замок на даних. Без ключа замість тексту — каша з символів. KMS — банк, який зберігає ключі за тебе, щоб ти їх не загубив і не віддав чужим.

🎯 Що ти дізнаєшся

- що таке шифрування і чим симетричне відрізняється від асиметричного
- як HTTPS домовляється про секретний ключ і навіщо сертифікат
- чим захист «у спокої» відрізняється від захисту «в дорозі»
- як працює банк ключів KMS і «конверти» envelope encryption
- як шифрують S3, диски й бази — і як зашифрувати вже існуючі
- CloudHSM, Secrets Manager, Parameter Store і сертифікати ACM

🥐 У базі «Хмаринки» вже тисячі телефонів і адрес клієнтів. Юрист попереджає Олю: «Персональні дані треба захищати — інакше штрафи». Оля хвилюється: «А якщо хтось украде диск просто з дата-центру?» — «Навіть тоді він побачить лише кашу з символів, — каже Тарас. — Якщо все зашифрувати. Покажу, як це працює».

Що таке шифрування

Шифрування (encryption, «енкрипшн») — перетворення зрозумілих даних на «кашу», яку можна повернути назад лише за допомогою **ключа**:

`Круасан з мигдалем, вул. Сихівська 5` → `k8#Qz!v2Lp0…`

Важливо розрізняти дві речі:

- **алгоритм** — «конструкція замка». Її не приховують: найкращі алгоритми (наприклад, AES) відомі всім і перевірені тисячами фахівців;
- **ключ** — те, що тримають у секреті. Без ключа навіть знаючи алгоритм, дані не прочитати за розумний час.

Симетричне шифрування: один ключ

Симетричне шифрування — як ключ від квартири: **той самий ключ** і замикає, і відмикає. Це дуже швидко, тому ним шифрують великі обсяги даних. Найвідоміший алгоритм — **AES-256**.

Слабке місце одне: як передати ключ іншій стороні? Якщо передати його відкрито, дорогою його можуть підглянути.

Асиметричне шифрування: відкритий замок і ключ

Асиметричне шифрування використовує **пару** ключів:

- **публічний** (public key) — можна роздавати всім;
- **приватний** (private key) — тримаєш лише в себе.

Що зашифровано публічним, розшифрує **лише** приватний. Образ — **поштова скринька з щілиною**: будь-хто може вкинути лист (зашифрувати публічним ключем), але відчинити скриньку може лише власник ключа.

Асиметричне шифрування повільніше, тому ним зазвичай шифрують не самі дані, а **передачу ключа** і **цифрові підписи** (доказ, що документ підписав саме власник приватного ключа).

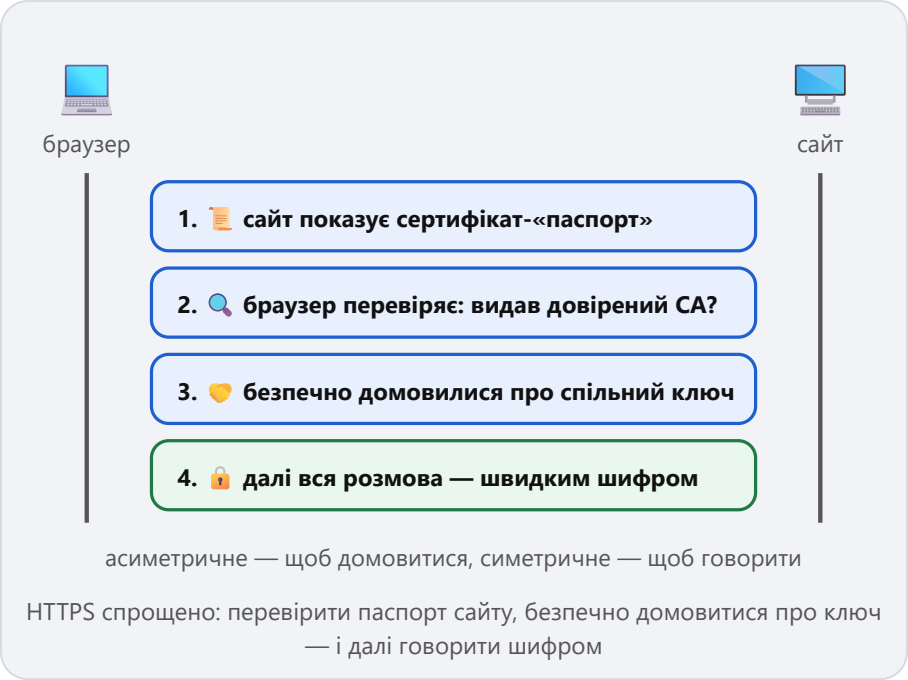


Симетричне шифрування — один ключ на двох; асиметричне — скринька: вкинути може будь-хто, відчинити — лише власник

HTTPS: як браузер і сайт домовляються

Тепер зрозуміло, як працює замочок у браузері. Спрощено:

1. Браузер звертається до сайту, сайт показує свій **сертифікат** — «паспорт» з публічним ключем.
2. Браузер перевіряє паспорт: чи видав його **довірений центр сертифікації** (CA — «сі-ей»), чи не прострочений, чи збігається ім'я сайту.
3. Користуючись асиметричним шифруванням, сторони безпечно домовляються про **спільний симетричний ключ**.
4. Далі вся розмова йде швидким симетричним шифруванням.



Протокол, який це робить, називається **TLS** («ті-ел-ес»; старша назва — SSL). Тому кажуть «TLS-сертифікат» або «SSL-сертифікат» — це одне й те саме.

Дані в спокої і дані в дорозі

Дані бувають у двох станах — і захищати треба обидва:

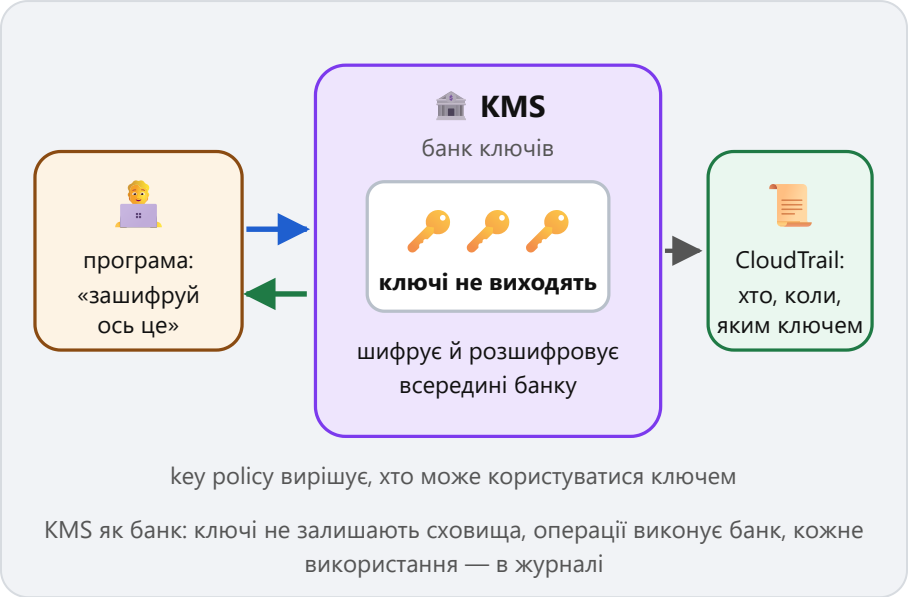
- **At rest** («ет рест», у спокої) — дані лежать: на диску, у сховищі, в базі. Образ — **документи в сейфі**. Захист — шифрування ключем, яким керує, наприклад, KMS.

- **In transit** («ін транзит», у дорозі) — дані їдуть мережею. Образ — **броньований інкасаторський фургон**. Захист — TLS, тобто HTTPS.

> 🎯 На іспиті: «encrypt data at rest» → KMS (шифрування сховищ, дисків, баз); «encrypt data in transit» → TLS/HTTPS (сертифікати ACM на балансувальнику чи CloudFront).

KMS — банк ключів

AWS KMS («кей-ем-ес», Key Management Service — сервіс керування ключами) — **банк ключів**. Головне правило банку: **ключ тобі на руки не видають**. Ти приносиш дані й кажеш «зашифруй» або «розшифруй» — банк робить це всередині, у спеціальному захищеному обладнанні, і **записує кожну операцію в журнал** (CloudTrail): хто, коли, яким ключем.



Ключі KMS бувають трьох видів:

- **AWS owned** — ключі, якими AWS шифрує дані «під капотом»; ти їх не бачиш і нічого не платиш.
- **AWS managed** — ключі з назвами на кшталт `aws/s3`, `aws/ebs`, створені сервісами для тебе. Ротація (заміна ключового матеріалу) — **автоматично щороку**; керувати їхньою політикою не можна.
- **Customer managed** — **твої** ключі. Ти сам пишеш **key policy** (хто може користуватися ключем), налаштовуєш ротацію (раз на 90–2560 днів, за замовчуванням щороку, або вручну на вимогу) і можеш дати доступ іншим акаунтам.

Ще кілька важливих фактів:

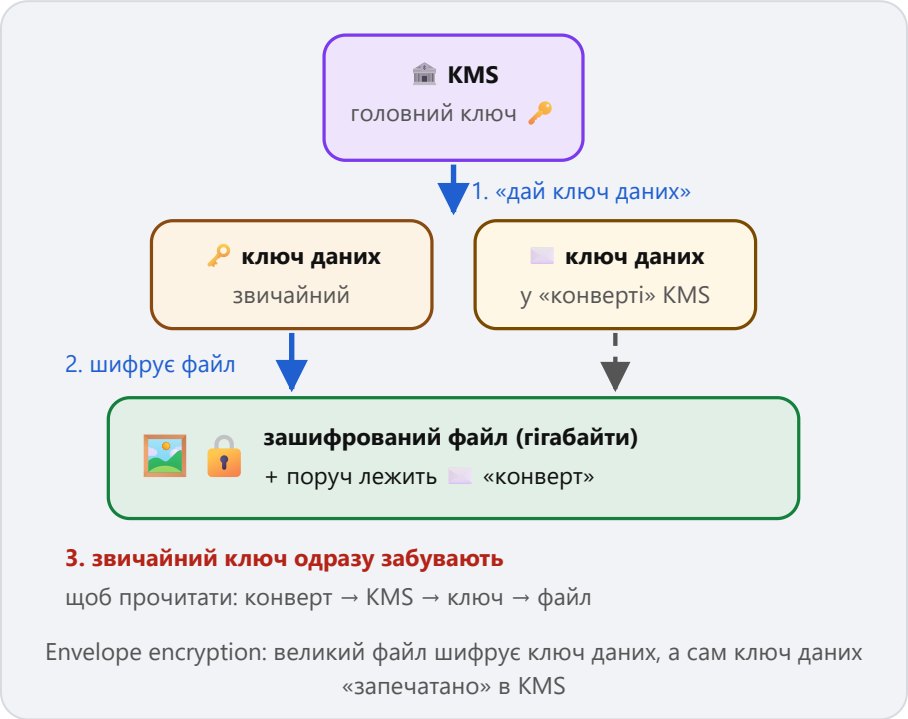
- **Key policy обов'язкова** — це головний «замок» ключа. Якщо вона не дозволяє, жодна IAM-політика доступу до ключа не дасть.
- KMS — **регіональний**. Для шифрування в одному регіоні й розшифрування в іншому тим самим ключем є **Multi-Region keys**.
- Ключ не видаляють миттєво: лише з **очікуванням 7–30 днів** (за замовчуванням 30). Поки строк не минув — видалення можна скасувати. Безпечніша альтернатива — **вимкнути** ключ. Адже дані, зашифровані видаленим ключем, повернути неможливо.

Envelope encryption: лист у конверті, конверт у сейфі

KMS шифрує напряму лише **невеликі** дані — **до 4 КВ**. А як шифрувати фото на мегабайти чи бази на гігабайти? Для цього є **envelope encryption** («енвелоуп енкрипшн», шифрування конвертом):

1. Програма просить KMS: «дай **ключ даних**» (GenerateDataKey).
2. KMS повертає ключ даних у **двох** видах: звичайному й зашифрованому своїм головним ключем.

3. Програма шифрує великий файл **звичайним** ключем даних просто в себе — швидко — і одразу **забуває** звичайний ключ.
4. Поруч із зашифрованим файлом зберігається **зашифрований** ключ даних — «конверт».
5. Щоб прочитати файл, програма надсилає конверт у KMS, отримує звичайний ключ, розшифровує файл і знову його забуває.



Envelope encryption: великий файл шифрує ключ даних, а сам ключ даних «запечатано» в KMS

Образ з простого курсу: **лист у конверті, конверт у сейфі банку**. Сервіси AWS (S3, EBS, бази) роблять усе це

самі — тобі досить обрати ключ. Але на іспиті слова «4 KB» і «data key» — підказка саме на envelope encryption.

Як шифрують сховище S3

- **SSE-S3** — шифрування ключами, якими повністю керує S3. **Увімкнене за замовчуванням** для всіх нових об'єктів з 2023 року.
- **SSE-KMS** — шифрування ключем KMS: кожне використання ключа видно в журналі CloudTrail, а доступ до ключа — окремий від доступу до файлу. Обирають, коли потрібен **аудит** і контроль ключа.
- **DSSE-KMS** — подвійне шифрування двома шарами, для найсуворіших вимог.
- **SSE-C** — ключ щоразу приносить клієнт; AWS його не зберігає (лише через HTTPS).
- **Шифрування на клієнті** — дані шифрують ще до відправки в S3.

Два практичні нюанси:

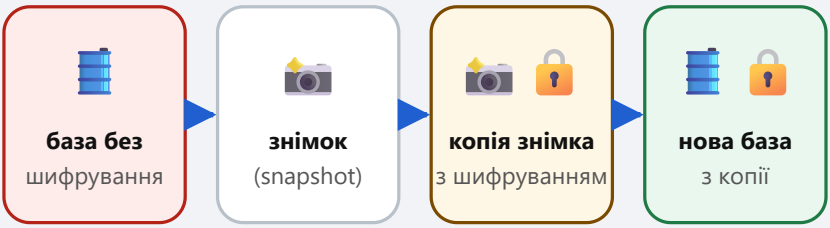
- Якщо мільйони файлів шифруються через SSE-KMS, KMS може почати відповідати помилкою **ThrottlingException** (забагато запитів). Ліки — **S3 Bucket Keys**: сховище отримує проміжний ключ і звертається до KMS у рази рідше. Ще й дешевше.

- Щоб **заборонити нешифроване з'єднання** зі сховищем, у bucket policy пишуть Deny за умови `aws:SecureTransport = false`.

Диски й бази: як зашифрувати вже існуючі

Нові диски EBS і бази RDS шифруються однією галочкою при створенні. Для дисків можна ввімкнути **шифрування за замовчуванням** на весь регіон.

А от **існуючий** нешифрований диск чи базу «зашифрувати галочкою» не можна. Робиться це через знімок — як **переїзд у зашифровану квартиру**:



«переїзд у зашифровану квартиру»: галочкою на існуючій базі не ввімкнути

після відновлення — нова адреса бази, застосунок переналаштовують

Зашифрувати наявну базу чи диск: знімок → копія знімка з шифруванням → нова база чи диск з копії

1. Зробити **знімок** (snapshot) бази чи диска.
2. **Скопіювати** знімок, увімкнувши шифрування.
3. **Відновити** нову базу чи диск із зашифрованої копії й переключити застосунок на неї.

Ще два факти: репліка бази для читання шифрується так само, як основна база; а поділитися зашифрованим знімком з іншим акаунтом можна лише тоді, коли він зашифрований **customer managed** ключем (і тому акаунту дано доступ до ключа). Знімок з AWS managed ключем спершу копіюють, перешифровуючи своїм ключем.

CloudHSM — власний сейф

AWS CloudHSM («клауд-ейч-ес-ем», Hardware Security Module) — **виділений апаратний модуль** лише для тебе (single-tenant, «сингл-тенант»): ключі повністю під твоїм контролем, **AWS не має до них доступу**. Відповідає суворому стандарту **FIPS 140 Level 3**. Працює через стандартні інтерфейси (PKCS#11, JCE, CNG), підходить для шифрування баз Oracle (TDE), для «розвантаження» TLS і як особисте сховище ключів для KMS. Для надійності — кластер щонайменше з **двох** HSM у різних AZ.

Образ з простого курсу: KMS — **банк ключів** (спільний, зручний, інтегрований з усім), CloudHSM — **власний сейф у сховищі банку**, код від якого знаєш тільки ти.

🎯 «Ключі лише під контролем компанії, виділене обладнання, FIPS 140 Level 3, AWS не має доступу» → **CloudHSM**.

Секрети: паролі до баз і налаштування

Застосунку потрібні паролі до бази, ключі сторонніх сервісів, налаштування. Писати їх у код — те саме, що ключ під килимком. Для цього є два сервіси:

- **AWS Secrets Manager — сейф з автоматичною заміною замків**: зберігає секрети й **сам змінює паролі** за розкладом (ротація) — вбудовано для RDS, Aurora, Redshift, DocumentDB, для інших — через функцію Lambda. Уміє копіювати секрети в інші регіони. Платний за кожен секрет.
- **SSM Parameter Store — шафка для налаштувань**: зберігає параметри й секрети (тип SecureString шифрується KMS), має ієрархію «папок», звичайний рівень — безкоштовний. **Вбудованої ротації немає.**

🎯 «Автоматично змінювати пароль до бази кожні 30 днів» → **Secrets Manager**. «Дешево зберігати налаштування й рядки підключення» → **Parameter Store**.

Сертифікати для HTTPS: ACM

ACM («ей-сі-ем», AWS Certificate Manager) — **паспортний стіл для сайтів**: видає **безкоштовні** публічні TLS-сертифікати й **сам їх поновлює** (якщо підтвердити домен через DNS). Сертифікат ставлять на **балансувальник, CloudFront, API Gateway**.

Три факти, які треба знати напам'ять:

- Для **CloudFront** сертифікат ACM має бути в регіоні **us-east-1** (для балансувальника — у регіоні балансувальника).
- Публічний сертифікат ACM **не можна** встановити просто на сервер EC2 — лише на інтегровані сервіси.
- **Імпортовані** (куплені деінде й завантажені в ACM) сертифікати ACM сам не поновлює — за строком стежать через сповіщення.

> 🔍 **Для допитливих: Хеш — відбиток пальця даних**
>
> Поруч із шифруванням живе **хешування**. Хеш-функція перетворює будь-які дані на короткий «відбиток» фіксованої довжини. Той самий файл завжди дає той самий відбиток, а зміна хоч однієї коми — зовсім інший. Повернути дані з відбитка неможливо — це дорога в один бік. Навіщо? Щоб перевіряти **цілісність**: наприклад, AWS CloudTrail

уміє доводити, що журнали ніхто не змінював (log file integrity validation), саме порівнюючи відбитки. А паролі користувачів нормальні системи зберігають не як є, а як хеші.

📖 Словничок

Шифрування («енкрипшн») — перетворення даних на «кашу», яку повертає лише ключ.

Симетричне — один ключ на обидві дії (AES-256). **Асиметричне** — пара: публічний і приватний ключі.

Сертифікат / CA («сі-ей») — «паспорт» сайту / центр, що видає паспорти.

TLS («ті-ел-ес», колишній SSL) — протокол шифрування в дорозі; основа HTTPS.

At rest / in transit — дані в спокої (сейф) / у дорозі (броньований фургон).

KMS («кей-ем-ес») — банк ключів; ключі не залишають сервіс, кожне використання в журналі.

AWS owned / AWS managed / customer managed key — невидимий / створений сервісом (`aws/s3`) / твій власний.

Key policy — обов'язкова політика доступу до ключа.

Envelope encryption — великі дані шифрує ключ даних, а ключ даних шифрує KMS (напряму KMS — до 4 KB).

SSE-S3 / SSE-KMS / SSE-C — способи шифрування S3: ключі S3 / ключ KMS з аудитом / ключ клієнта.

S3 Bucket Keys — менше викликів KMS, без ThrottlingException, дешевше.

CloudHSM — виділений апаратний модуль ключів, FIPS 140 Level 3.

Secrets Manager — секрети з автоматичною ротацією. **Parameter Store** — налаштування й секрети без ротації.

ACM («ей-сі-ем») — безкоштовні TLS-сертифікати з автопоновленням; для CloudFront — us-east-1.

🧩 **Міні-тест: перевір себе**

? Що таке симетричне шифрування?

- **А** Той самий ключ і шифрує, і розшифровує
- **Б** Шифрування двома різними ключами: публічним і приватним
- **В** Шифрування без ключа
- **Г** Однакове шифрування для всіх сайтів

Відповідь: А. Симетричне — як ключ від квартири. Швидко, тому ним шифрують великі обсяги даних.

? Від чого захищає HTTPS?

- **А** Від вірусів на диску
- **Б** Від підглядання й підміни даних у дорозі між браузером і сайтом
- **В** Від видалення бази даних
- **Г** Від DDoS-атак

Відповідь: Б. HTTPS (TLS) — шифрування in transit. Дані в спокої захищає окреме шифрування, наприклад KMS.

? Потрібно бачити в журналі, хто і коли використовував ключ для шифрування даних у S3. Що обрати?

А SSE-S3

Б SSE-KMS

В Жодного шифрування

Г SSE-C

Відповідь: Б. З SSE-KMS кожне використання ключа видно в CloudTrail, а доступ до ключа контролюється окремо.

? KMS напряму шифрує лише до 4 КВ. Як шифрують великі файли?

А Ділять файл на шматочки по 4 КВ і відправляють у KMS

Б Envelope encryption: файл шифрує ключ даних, а ключ даних шифрує KMS

В Великі файли не шифрують

Г Купують більший KMS

Відповідь: Б. «Лист у конверті, конверт у сейфі»: швидко і безпечно.

? Існуюча нешифрована база RDS має стати зашифрованою. Як?

А Поставити галочку «шифрувати» в налаштуваннях

Б Знімок → копія знімка з шифруванням → відновити нову базу з копії

В Перезавантажити базу

Г Це неможливо

Відповідь: Б. На існуючій базі чи диску шифрування «галочкою» не вмикається — лише через зашифровану копію знімка.

? Пароль до бази має автоматично змінюватися кожні 30 днів. Що обрати?

- **А** SSM Parameter Store
- **Б** AWS Secrets Manager
- **В** Записати пароль у код
- **Г** KMS

Відповідь: Б. Secrets Manager — «сейф з автозаміною замків». У Parameter Store вбудованої ротації немає.

? Ключі мають бути лише під контролем компанії, на виділеному обладнанні з FIPS 140 Level 3, і AWS не повинен мати до них доступу. Що обрати?

А AWS managed key

Б Customer managed key у KMS

В AWS CloudHSM

Г SSE-S3

Відповідь: В. CloudHSM — власний сейф: single-tenant, повний контроль над ключами.

? У якому регіоні треба створити сертифікат ACM для CloudFront?

- **А** us-east-1
- **Б** У будь-якому
- **В** У регіоні, найближчому до клієнтів
- **Г** eu-central-1

Відповідь: А. Для CloudFront — лише us-east-1. Для балансувальника — у регіоні самого балансувальника.

✅ **Головне**

Симетричне шифрування — один ключ (швидко, AES-256); **асиметричне** — «скринька з щілиною»: публічний і приватний ключі. **HTTPS/TLS** поєднує обидва; сертифікат — «паспорт» сайту.

Захищай дані **at rest** (KMS) і **in transit** (TLS). **KMS** — банк ключів: ключі не виходять, кожне використання в журналі; key policy обов'язкова; видалення — через 7–30 днів.

Envelope encryption: KMS напряму — до **4 KB**, великі дані шифрує ключ даних.

S3: **SSE-S3** за замовчуванням, **SSE-KMS** для аудиту, **Bucket Keys** проти throttling. Існуючі RDS/EBS — через **зашифровану копію знімка**.

CloudHSM — власний сейф (FIPS 140 Level 3); **Secrets Manager** — ротація паролів; **Parameter Store** — налаштування; **ACM** — безкоштовні сертифікати, для CloudFront — **us-east-1**.

🛡️ Атаки й охорона: від DDoS до розслідувань

👴 Простими словами

Атаки бувають різні: від натовпу фальшивих відвідувачів (DDoS) до злодія, що підбирає паролі. AWS дає охоронців (Shield, WAF), сигналізацію (GuardDuty) і детектива для розслідувань.

🎯 Що ти дізнаєшся

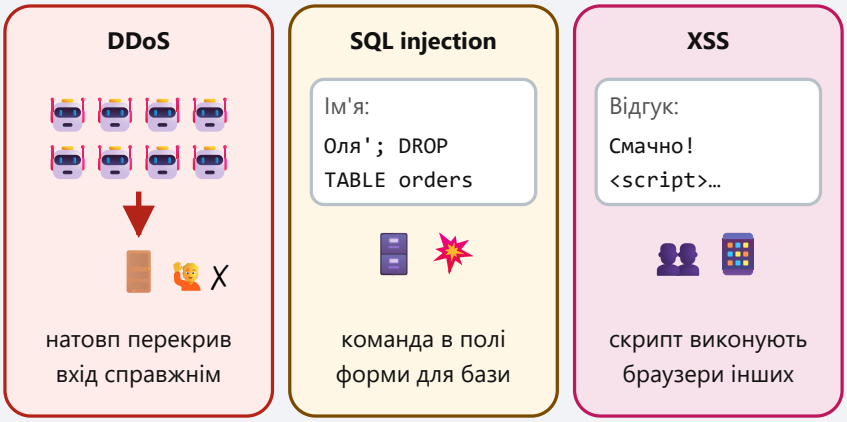
- як нападають на сайти — DDoS, SQL injection, XSS, підбір паролів, вразливості — простими словами
- що таке захист на всіх рівнях
- що роблять WAF, Shield, Network Firewall і Firewall Manager
- як GuardDuty, Inspector, Macie, Detective і Security Hub помічають загрози
- навіщо CloudTrail, Config і Artifact
- як налаштувати автоматичну реакцію

🥐 Одного вечора сайт «Хмаринки» почав гальмувати: тисячі запитів за секунду з незнайомих адрес. Уранці Тарас помітив у журналі сотні спроб угадати пароль адмінки, а ще — дивний вхід з іншої країни. «На нас напали?» — лякається Оля. «Нас перевіряли на міцність, — каже Тарас. — Це нормально для будь-якого сайту в інтернеті. Головне — мати охорону. У AWS її багато, і кожен охоронець робить своє».

Як нападають: словничок злодія

- **DDoS** («ді-дос», distributed denial of service — розподілена відмова в обслуговуванні) — **натовп фальшивих відвідувачів** перекриває вхід, і справжні клієнти не можуть зайти. «Розподілена» — бо натовп іде з тисяч заражених комп'ютерів одночасно. Буває на рівні мережі (засипати пакетами) і на рівні сайту (засипати HTTP-запитами).
- **SQL injection** («ес-к'ю-ел ін'єкшн») — зловмисник вписує в поле форми не ім'я, а **шматок команди для бази даних**. Якщо програма наївна, база цю команду виконає. Уяви, що в бланку замовлення написали «...і видайте ключ від каси», а кухар слухняно видав.

- **XSS** («екс-ес-ес», cross-site scripting) — у текст, який бачать інші користувачі (відгук, коментар), підкидають **шкідливий скрипт**. Браузери відвідувачів його виконують — і, наприклад, віддають зловмисникові їхні cookies.
- **Підбір паролів** (brute force) — мільйони спроб угадати пароль.
- **Боти** — програми, що автоматично обходять сайт: копіюють ціни, скуповують товар, шукають слабкі місця.
- **Вразливості** — відомі «тріщини» в програмах (їх нумерують як **CVE**). Зловмисники автоматично шукають сервери, де тріщини ще не залатані.
- **Викрадені ключі** — і на твоєму рахунку з'являються чужі сервери для майнингу криптовалют.
- **Витік даних** — наприклад, сховище з документами випадково стало публічним.

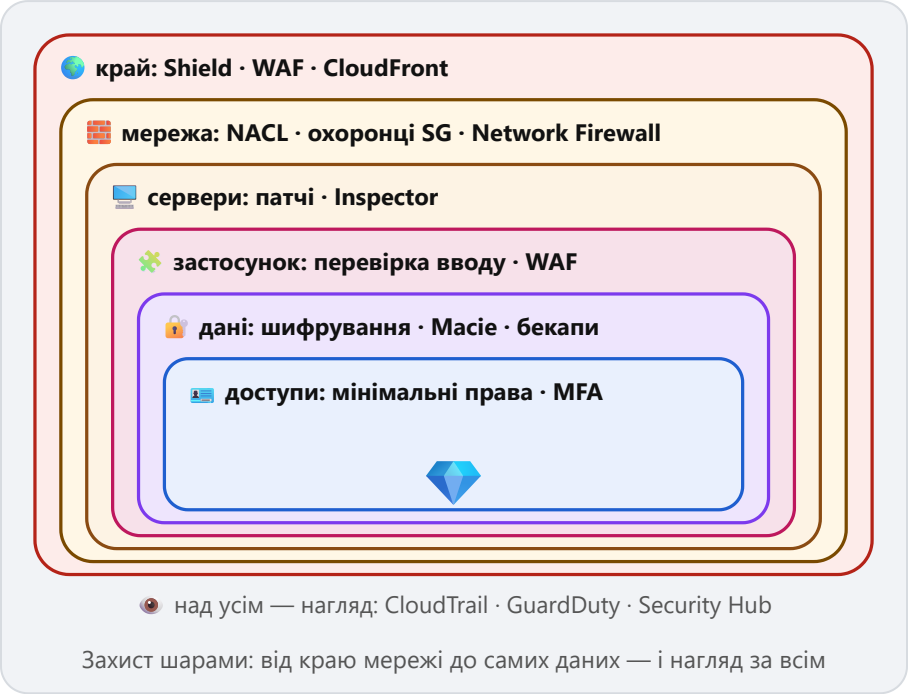


від усіх трьох на рівні сайту захищає AWS WAF (DDoS — ще й Shield)

Три класичні атаки на сайт: натовп (DDoS), «команда в бланку» (SQL injection) і «шкідлива записка» (XSS)

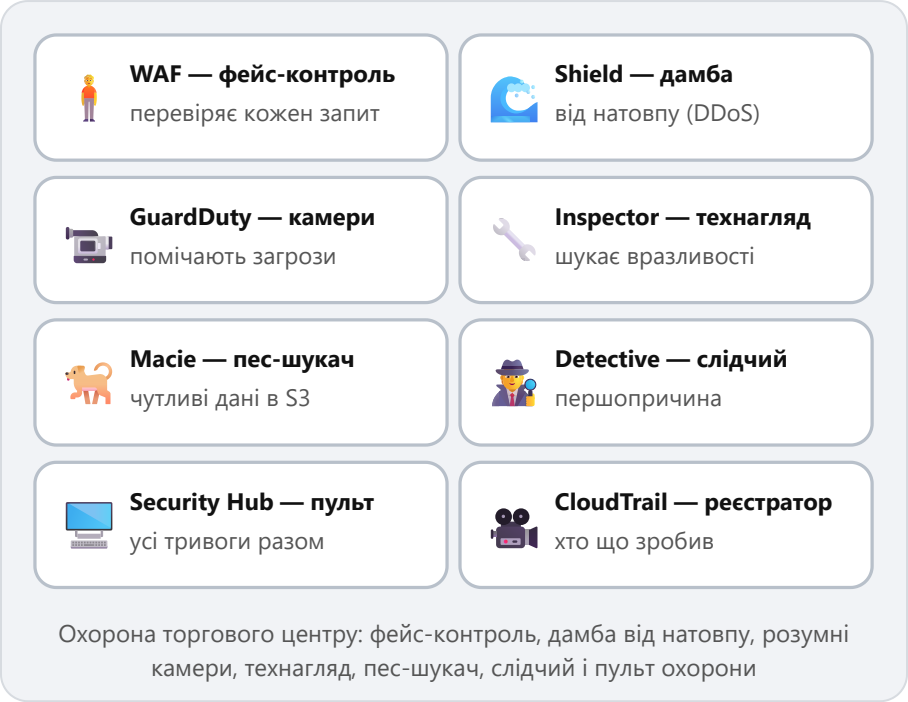
Захист на всіх рівнях

Жоден окремий замок не зупинить усіх. Тому захист будують **шарами**, як середньовічний замок: рів, стіна, ворота, варта, сейф у вежі. Навіть якщо один шар подолано, далі — наступний. Це і є **defense in depth** — захист на всіх рівнях.



- **Край** — захист від натовпу й фільтр запитів ще до того, як вони дійдуть до сайту.
- **Мережа** — хто взагалі може постукати в які двері.
- **Сервери** — патчі, пошук вразливостей.
- **Застосунок** — перевірка того, що вводять користувачі.
- **Дані** — шифрування, пошук чутливого, бекапи.
- **Доступи** — мінімальні права, MFA.
- **Нагляд** — журнали, тривоги, автоматична реакція.

Далі — охоронці AWS для кожного шару. Образ з простого курсу: **торговий центр та його охорона**.



Охорона торгового центру: фейс-контроль, дамба від натовпу, розумні камери, технагляд, пес-шукач, слідчий і пульт охорони

WAF — фейс-контроль на вході

AWS WAF («ваф», web application firewall — файрвол вебзастосунків) перевіряє **кожен HTTP-запит** (це «рівень 7» — рівень змісту запиту) за правилами:

- блокує типові атаки — **SQL injection** і **XSS**;
- блокує окремі IP-адреси та цілі **країни** (geo match);
- **rate-based rules** — «цей відвідувач заходить надто часто»: обмежує кількість запитів з однієї адреси;
- готові набори правил від AWS і партнерів (managed rule groups), захист від ботів (Bot Control).

WAF ставлять **перед** сайтом: на **CloudFront**, **балансувальник ALB**, **API Gateway**, AppSync, Cognito, App Runner.

⚠️ WAF **не ставиться на балансувальник NLB** і не ставиться прямо на сервер EC2. Це улюблена пастка іспиту.

Shield — дамба від натовпу

- **Shield Standard** — **безкоштовно й автоматично** для всіх клієнтів AWS: захист від поширених DDoS-атак на рівні мережі (рівні 3–4). Нічого вмикати не треба.
- **Shield Advanced** — платна «страховка» (близько 3 000 доларів на місяць на організацію): команда експертів **SRT** цілодобово, **компенсація рахунку**, якщо під час атаки ресурси масштабувалися й витрати зросли, розширене виявлення і захист на рівні 7 разом з WAF. Для CloudFront, Route 53, Global Accelerator, балансувальників і EC2 з Elastic IP.

🎯 «DDoS-захист + експерти 24/7 + компенсація витрат від атаки» → **Shield Advanced**. Звичайний базовий захист уже є в усіх — це Shield Standard.

Мережеві файрволи й централізоване керування

- **AWS Network Firewall** — **митниця на в'їзді в мережу** (VPC): фільтрує весь трафік, зокрема **вихідний за доменними іменами** («сервери можуть ходити лише на сайти оновлень»), вміє виявляти й зупиняти вторгнення (IPS/IDS).
- **Route 53 Resolver DNS Firewall** — не дає серверам навіть дізнатися адресу шкідливих доменів.
- **AWS Firewall Manager** — **єдиний пульт** правил WAF, Shield Advanced, мережевих охоронців і Network Firewall для **всіх акаунтів організації**. Нові акаунти й ресурси отримують правила автоматично.

Виявлення: хто помічає проблеми

П'ять сервісів, які постійно плутають. Запам'ятовуй за образами:

- **Amazon GuardDuty** («гард-д'юті») — **розумні камери**, що помічають **загрози**. Аналізує журнали дій (CloudTrail), мережеві потоки (VPC Flow Logs) і DNS-запити, а за бажанням — і S3, Kubernetes (систему керування контейнерами — про неї в частині про обчислення), бази, Lambda. Помічає криптомайнинг, викрадені ключі, зв'язок з відомими шкідливими адресами, дивні входи.

- **Amazon Inspector** («інспектор») — **технагляд**, що шукає **вразливості** (CVE) і відкриті мережеві шляхи в серверах EC2, образах контейнерів і функціях Lambda. Працює безперервно.
- **Amazon Macie** («мейсі») — **пес-шукач**, що нюхає **чутливі дані** (персональні дані, номери карток) у сховищах **S3**.
- **Amazon Detective** — **слідчий**: коли тривога вже спрацювала, будує граф подій і допомагає знайти **першопричину**.
- **AWS Security Hub** — **пульт охорони**: збирає знахідки всіх інших в одне місце й перевіряє відповідність стандартам (CIS, PCI DSS, найкращі практики AWS).

Мнемоніка з простого курсу: **Guard**Duty — охорона від загроз. **Inspect**or — інспектор дефектів. **Macie** — «мої персональні дані».

> 💡 Inspector **знаходить** вразливості, але не латає. Патчі ставить інший сервіс — Patch Manager з набору Systems Manager (про нього — у частині про експлуатацію).

Аудит і відповідність

- **AWS CloudTrail** — **відеореєстратор** усіх дій: **хто, що, коли і звідки** зробив (кожен виклик API). Історія за **90 днів** безкоштовно; для довгого зберігання — «trail» у S3; **organization trail** — журнал одразу всіх акаунтів. Уміє доводити, що журнали не змінювали (log file integrity validation).
- **AWS Config** — **фотоархів налаштувань**: як виглядав кожен ресурс учора й місяць тому і чи відповідає він правилам («усі диски зашифровані», «жодного публічного сховища»). Порушення можна **виправляти автоматично**.
- **AWS Artifact** — «шафа з документами» AWS для твоїх аудиторів: звіти відповідності самого AWS (SOC, PCI, ISO) і угоди (наприклад, для медичних даних).
- **AWS Audit Manager** — автоматично збирає докази для аудиту.

⚠️ Не плутай трійцю: **CloudTrail** — **хто** зробив (дії). **Config** — **яким був** ресурс і чи за правилами (стан). **CloudWatch** — **метрики, логи й тривоги** (здоров'я і продуктивність, про нього — у частині про експлуатацію).

Автоматична реакція

Помітити загрозу мало — треба швидко відреагувати, навіть о третій ночі. Типовий ланцюжок:



Автоматична реакція: знахідка GuardDuty → правило EventBridge → функція, що ізолює сервер і сповіщає людей

1. GuardDuty знаходить загрозу (наприклад, сервер спілкується з відомою шкідливою адресою).
2. Знахідка стає **подією** в сервісі **EventBridge** (що таке подія, пам'ятаєш з розділу про застосунки).
3. Правило EventBridge запускає **функцію Lambda** чи автоматизацію Systems Manager: ізолювати сервер, відкликати ключі, написати в чат чергових.

Так само працює AWS Config: знайшов публічне сховище — автоматично закрив.

А щоб увімкнути GuardDuty, Security Hub, Macie й Inspector в **усіх** акаунтах організації (і в нових теж), призначають один **delegated administrator**

(«делегований адміністратор») — окремий акаунт безпеки, який керує ними централізовано.

> 🥐 Тарас поставив сайт за CloudFront із WAF: правило «не більше 1 000 запитів за 5 хвилин з однієї адреси» і готові набори правил проти SQL injection і XSS. Shield Standard уже працював сам. Увімкнув GuardDuty і Security Hub, а CloudTrail пише журнали всіх акаунтів у захищене сховище. Наступного разу натовп ботів навіть не дійде до сервера.

📖 **Словничок**

DDoS («ді-дос») — атака натовпом запитів з тисяч пристроїв.

SQL injection / XSS — «команда в бланку» для бази / «шкідлива записка» у вмісті сайту.

Brute force — підбір паролів. **CVE** — номер відомої вразливості.

Defense in depth — захист шарами.

AWS WAF («ваф») — фільтр HTTP-запитів (рівень 7): SQLi, XSS, IP, країни, rate-based; не на NLB.

Shield Standard / Advanced — безкоштовний базовий DDoS-захист / платна «страховка» з експертами SRT і компенсацією.

Network Firewall — файрвол на рівні VPC, зокрема фільтр вихідного трафіку за доменами.

Firewall Manager — єдиний пульт правил для всіх акаунтів організації.

GuardDuty — загрози. **Inspector** — вразливості. **Macie** — чутливі дані в S3. **Detective** — розслідування. **Security Hub** — пульт знахідок і стандартів.

CloudTrail — хто що зробив (API). **Config** — стан ресурсів і правила. **Artifact** — звіти відповідності AWS.

Delegated administrator — акаунт, що централізовано керує сервісами безпеки для всієї організації.

🧩 **Міні-тест: перевір себе**

? Що таке DDoS-атака?

А Натовп фальшивих запитів з тисяч пристроїв, щоб справжні клієнти не могли зайти

Б Крадіжка пароля через фальшивий сайт

В Помилка в програмі

Г Видалення бази даних

Відповідь: А. «Розподілена відмова в обслуговуванні»: натовп перекриває вхід.

? Сайт за балансувальником ALB треба захистити від SQL injection і XSS. Що обрати?

- **А** AWS WAF
- **Б** Shield Standard
- **В** Amazon Macie
- **Г** Amazon Inspector

Відповідь: А. WAF перевіряє зміст HTTP-запитів і блокує типові атаки.

? Потрібно обмежити кількість запитів з однієї IP-адреси, щоб боти не завалювали сайт. Що використати?

- **А** Security Hub
- **Б** Правило WAF типу rate-based
- **В** CloudTrail
- **Г** Шифрування KMS

Відповідь: Б. Rate-based rule — «цей відвідувач заходить надто часто».

? Компанія хоче команду експертів з DDoS цілодобово і компенсацію витрат, спричинених атакою. Що обрати?

- **А** Shield Standard
- **Б** Shield Advanced
- **В** GuardDuty
- **Г** WAF без Shield

Відповідь: Б. Shield Standard — безкоштовний базовий захист у всіх. Експерти SRT і компенсація — лише в Shield Advanced.

? Треба знайти персональні дані клієнтів у тисячах сховищ S3. Що обрати?

- **А** GuardDuty
- **Б** Inspector
- **В** Macie
- **Г** Detective

Відповідь: В. Macie — «пес-шукач» чутливих даних у S3.

? Хто видалив сервер учора о 23:10? Де шукати відповідь?

- **А** CloudTrail
- **Б** AWS Config
- **В** CloudWatch
- **Г** Macie

Відповідь: А. CloudTrail — журнал усіх дій: хто, що, коли й звідки.

? Сервер раптом почав спілкуватися з відомими шкідливими адресами, а з акаунта йдуть незвичні виклики API. Який сервіс це помітить?

- **А** Inspector
- **Б** GuardDuty
- **В** Artifact
- **Г** ACM

Відповідь: Б. GuardDuty — «розумні камери» загроз: аналізує журнали, мережеві потоки й DNS.

? Треба знайти відомі вразливості (CVE) в образах контейнерів і на серверах. Що обрати?

- **А** Macie
- **Б** GuardDuty
- **В** Inspector
- **Г** Security Hub

Відповідь: В. Inspector — «технагляд»: шукає вразливості. Латає їх уже Patch Manager.

✅ **Головне**

Атаки: **DDoS** (натовп), **SQL injection** (команда в бланку), **XSS** (шкідлива записка), підбір паролів, боти, вразливості **CVE**, викрадені ключі, витоки.

Захист шарами: край → мережа → сервери → застосунок → дані → доступи → нагляд.

WAF — фейс-контроль HTTP (SQLi, XSS, країни, rate-based), **не на NLB**; **Shield Standard** — безкоштовно всім, **Advanced** — експерти й компенсація; **Network Firewall** — митниця VPC; **Firewall Manager** — пульт для всієї організації.

GuardDuty — загрози, **Inspector** — вразливості, **Macie** — чутливі дані в S3, **Detective** — розслідування, **Security Hub** — пульт.

CloudTrail — хто зробив, **Config** — стан і правила, **Artifact** — звіти AWS. Реакція: знахідка → **EventBridge** → **Lambda**; для всієї організації — **delegated administrator**.

🏠 VPC: будуємо власне містечко в хмарі

👴 Простими словами

VPC — твоє власне огороджене містечко в хмарі. Ти сам вирішуєш, які в ньому квартали (підмережі), де ворота в інтернет і куди ведуть дороги.

🎯 Що ти дізнаєшся

- що таке VPC і чому це «огороджене містечко»
- як крок за кроком збудувати мережу — діапазон адрес, підмережі в різних AZ, ворота в інтернет, дорожні вказівники
- чим публічна підмережа відрізняється від приватної — одним рядком
- як приватні сервери виходять в інтернет через NAT Gateway і чому NAT ставлять у кожній AZ
- що таке Elastic IP і egress-only ворота

🥐 Настав час будувати «дім» для сайту «Хмаринки» в AWS. «Спершу огородимо ділянку, — каже Тарас, розстеляючи на столі аркуш. — Намалюємо вулиці, квартали, ворота й дорожні знаки. У хмарі це називається VPC. Без неї жоден сервер не має де жити».

Що таке VPC

VPC («ві-пі-сі», Virtual Private Cloud — віртуальна приватна хмара) — це **твоя власна ізольована мережа** в регіоні AWS. Образ з простого курсу — **огороджене містечко** всередині великого міста AWS. Ніхто ззовні не зайде в нього, доки ти власноруч не збудуєш ворота. Сервери, бази й балансувальники «живуть» саме у VPC.

Кілька властивостей:

- VPC — **регіональна**: вона охоплює всі зони доступності регіону;
- в акаунті може бути багато VPC, і за замовчуванням вони **не бачать одна одну**;
- у кожному регіоні вже є **default VPC** — «містечко за замовчуванням» для швидкого старту. Для серйозних проєктів будують власну.

Побудуємо VPC «Хмаринки» за чотири кроки.

Крок 1. Діапазон адрес — номери будинків

Спершу обирають **CIDR** — діапазон приватних адрес для всього містечка. Ти вже вмієш його читати: `10.0.0.0/16` — це 65 536 адрес.

- Розмір VPC — від `/16` (найбільша) до `/28` (найменша).
- Діапазон варто обирати так, щоб він **не перетинався** з офісом та іншими VPC — інакше їх потім не з'єднаєш.

Крок 2. Підмережі — квартали в районах

Містечко ділять на **підмережі** (subnets) — квартали. Головне правило: **кожна підмережа живе рівно в одній AZ**. Не може одна підмережа розтягнутися на два райони.

Для надійності кожен вид кварталу роблять **у двох (або трьох) AZ**. Типова схема — шість підмереж:

Підмережа	AZ a	AZ b	Хто там живе
Публічна	`10.0.1.0/24`	`10.0.2.0/24`	балансувальник, NAT Gateway
Приватна — застосунок	`10.0.11.0/24`	`10.0.12.0/24`	сервери з програмою сайту
Приватна — бази	`10.0.21.0/24`	`10.0.22.0/24`	бази даних

У **кожній** підмережі AWS забирає **5 адрес** собі: першу (адреса самої мережі), другу (вбудований роутер VPC),

третю (DNS), четверту (запас на майбутнє) і останню. Тому в `/24` для твоїх ресурсів лишається 251 адреса, а не 256.

Крок 3. Internet Gateway — головні ворота

Щоб містечко спілкувалося з інтернетом, будують **Internet Gateway** («інтернет гейтвей», IGW) — **головні ворота на трасу**. Одна VPC — одні такі ворота. Ними керує AWS: вони самі масштабуються і не бувають «вузьким місцем».

Але самі ворота нічого не вирішують. Щоб машини поїхали до воріт, потрібні вказівники.

Крок 4. Route tables — дорожні вказівники

Таблиця маршрутів (route table) — набір **дорожніх вказівників**: «пакети для таких-то адрес — туди». Кожна підмережа прив'язана до однієї таблиці.

У кожній таблиці завжди є рядок **local**: «усе, що в межах VPC (`10.0.0.0/16`), — всередину». Тому всі квартали одного містечка бачать одне одного.

А тепер найголовніший рядок усього розділу:

Куди (Destination)	Через (Target)
`10.0.0.0/16`	local
`0.0.0.0/0`	igw — Internet Gateway

Рядок `0.0.0.0/0 → igw` означає: «усе, що не наше, — до головних воріт». **Саме він робить підмережу публічною.**



Таблиця маршрутів: local — усе всередині VPC; 0.0.0.0/0 → IGW робить підмережу публічною. Перемагає найточніший маршрут

💡 Якщо для адреси підходять кілька рядків, перемагає **найточніший** (longest prefix match). Пакет на `10.0.2.5` підходить і під `10.0.0.0/16`, і під `0.0.0.0/0`, але `/16` точніший — тож він піде local, а не в інтернет.

Публічна чи приватна — вирішує один рядок

- **Публічна підмережа** — у її таблиці є маршрут `0.0.0.0/0 → Internet Gateway`. Ресурси з публічною IP-адресою тут досяжні з інтернету (якщо дозволить охоронець — про охоронців наступний розділ).
- **Приватна підмережа** — такого маршруту **немає**. Ззовні до неї не дістатися в принципі.

У публічних підмережах ставлять лише те, що мусить бути видно назовні: балансувальники, NAT Gateway. Сервери з програмою й бази — у **приватних**.

Публічні IP і Elastic IP

Щоб ресурс у публічній підмережі був досяжним з інтернету, йому потрібна **публічна IP-адреса**:

- звичайна публічна адреса видається автоматично, але **змінюється**, якщо сервер зупинити й запустити знову;
- **Elastic IP** («еластик ай-пі») — **постійна** публічна адреса, яка належить твоєму акаунту, доки ти її не віддаси. Її можна перечепити на інший сервер.

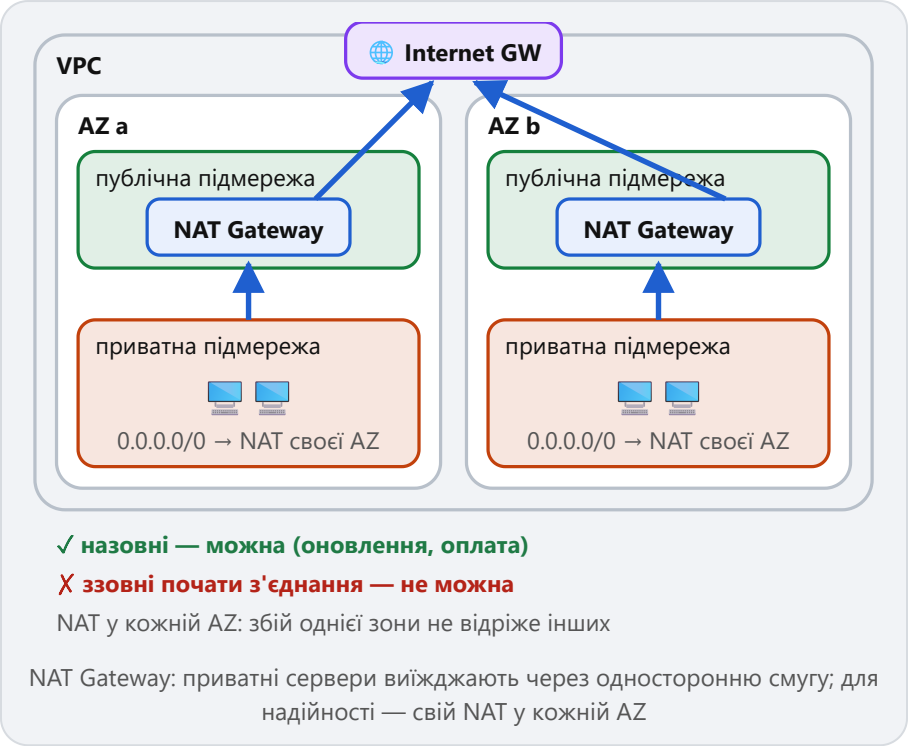
🆕 З 2024 року **всі** публічні IPv4-адреси в AWS платні (погодинно), навіть «вільні» Elastic IP. Тому зайвих публічних адрес не тримають.

NAT Gateway — виїзд з односторонньою смугою

Сервери в приватних підмережах мусять іноді виходити в інтернет — завантажити оновлення чи звернутися до стороннього сервісу оплати. Але ззовні до них ніхто не повинен дістатися. Пам'ятаєш домашній роутер з розділу про IP-адреси? Тут те саме.

NAT Gateway («нат гейтвей») — **виїзд з односторонньою смугою**:

1. NAT Gateway ставлять у **публічну** підмережу (з Elastic IP).
2. У таблиці маршрутів **приватної** підмережі пишуть: `0.0.0.0/0 → NAT Gateway`.
3. Приватні сервери виходять назовні через NAT, а відповіді повертаються тим самим шляхом. Почати з'єднання **ззовні** — неможливо.



Що важливо знати для іспиту:

- NAT Gateway — **керований** сервіс: масштабується сам, латати не треба.
- Платиш **за годину роботи і за кожен гігабайт**, що через нього проходить. (Запам'ятай — це повернеться в розділі про гроші.)

- **Для високої доступності — окремий NAT Gateway у кожній AZ**, і приватні підмережі кожної зони ходять через «свій» NAT. Якщо NAT один на всю VPC, то збій його зони відріже інтернет серверам в усіх зонах, а трафік з інших зон до нього ще й коштуватиме грошей (переїзд між районами платний).
- У NAT Gateway **немає** мережевих охоронців (Security Groups).
- Старий самокерований варіант — **NAT instance** (звичайний сервер з вимкненою перевіркою «source/destination check»): дешевший, але без автоматичної надійності й масштабування. На іспиті зазвичай програє NAT Gateway.

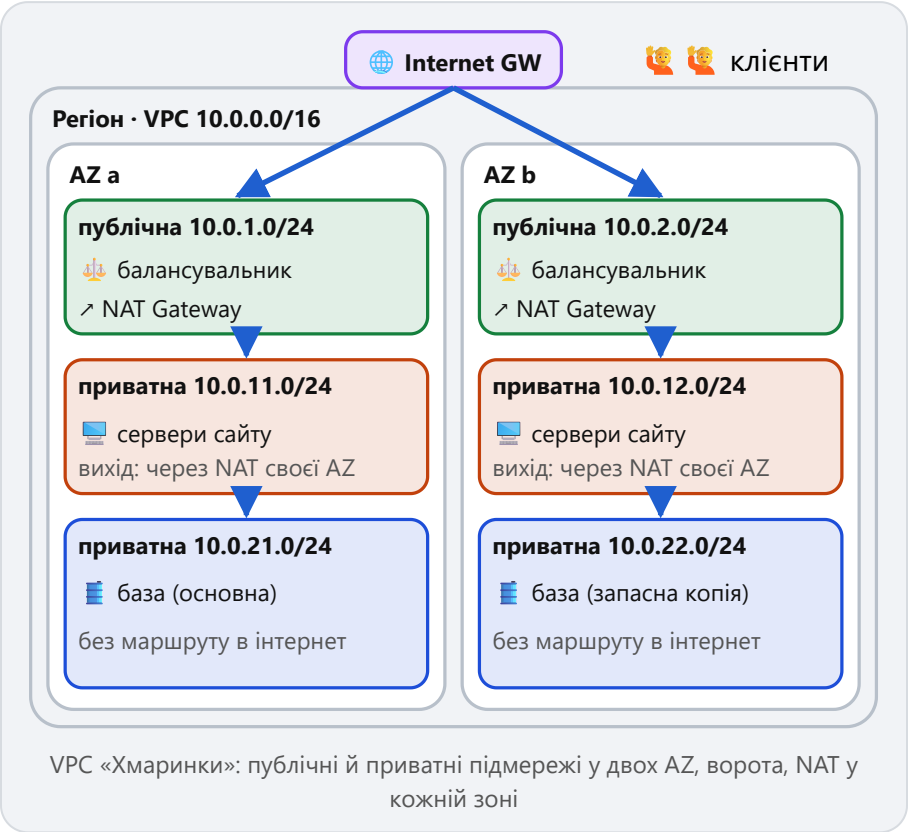
🆕 3 листопада 2025 року є **Regional NAT Gateway** — один NAT на всю VPC, який сам розширюється на потрібні AZ і не потребує публічної підмережі. Але на іспиті частіше питають класичну схему: **NAT у публічній підмережі кожної AZ**.

IPv6: ворота лише на вихід

В IPv6 кожна адреса публічна, і NAT як такий не потрібен. Але що, як треба дозволити серверам з IPv6 **виходити** в інтернет, не дозволяючи **заходити** до них? Для цього є

Egress-only Internet Gateway («ігрес-онлі») — **ворота лише на виїзд** для IPv6.

Увесь малюнок разом



VPC «Хмаринки»: публічні й приватні підмережі у двох AZ, ворота, NAT у кожній зоні

Простежимо шлях замовлення:

1. Клієнт з інтернету приходить через **Internet Gateway** до **балансувальника** в публічних підмережах.

2. Балансувальник передає запит **серверу застосунку** в приватній підмережі своєї зони.
3. Сервер звертається до **бази** в приватній підмережі баз.
4. Коли серверу треба оновитися, він виходить в інтернет через **NAT Gateway** своєї зони.

Жодна частина не стоїть в одній зоні сама: зламається район A — усе працює в районі B.

🧪 **Спробуй у справжньому AWS: Подивися на VPC у консолі й збудуй тренувальну (15 хвилин, безкоштовно)**

Сама VPC, підмережі, таблиці маршрутів і Internet Gateway нічого не коштують. Платні лише NAT Gateway, тож у цій вправі ми їх **не створюємо**.

1. У консолі AWS відкрий сервіс **VPC**. Переконайся, що вгорі обрано твій регіон.
2. Розділ **Your VPCs**: знайди default VPC — у неї діапазон `172.31.0.0/16`.
3. Розділ **Subnets**: по одній default-підмережі в кожній AZ. Подивися на колонку Availability Zone.
4. Розділ **Route tables**: відкрий таблицю default VPC — побачиш рядок `0.0.0.0/0` → `igw-…`. Ось чому всі default-підмережі публічні.

5. Тепер натисни **Create VPC** і обери **VPC and more**. Постав: 2 AZ, 2 публічні і 2 приватні підмережі, **NAT gateways: None**, VPC endpoints: None.
6. Праворуч з'явиться **схема-прев'ю**: підмережі, таблиці маршрутів, Internet Gateway — рівно те, що на ілюстрації вище. Натисни **Create VPC**.
7. Порівняй таблиці маршрутів публічних і приватних підмереж: у приватних немає рядка на `igw`.
8. Прибери за собою: **Actions** → **Delete VPC** для тренувальної VPC.

📖 **Словничок**

VPC («ві-пі-сі») — власна ізольована мережа в регіоні, «огороджене містечко».

Default VPC — готова VPC у кожному регіоні для швидкого старту (`172.31.0.0/16`).

Підмережа (subnet) — «квартал»; живе рівно в одній AZ; AWS резервує 5 адрес.

Internet Gateway, IGW — «головні ворота» VPC в інтернет.

Route table — «дорожні вказівники»; рядок local — усе всередині VPC.

Longest prefix match — перемагає найточніший маршрут.

Публічна / приватна підмережа — є / немає маршруту `0.0.0.0/0 → IGW`.

Elastic IP — постійна публічна IPv4-адреса акаунта (платна).

NAT Gateway — керований «виїзд з односторонньою смугою» для приватних підмереж; платний за годину й гігабайт; для HA — по одному на AZ.

NAT instance — старий самокерований варіант NAT на звичайному сервері.

Egress-only Internet Gateway — ворота «лише на вихід» для IPv6.

🧩 **Міні-тест: перевір себе**

? Що робить підмережу публічною?

А Слово «public» у назві

Б Маршрут 0.0.0.0/0 на Internet Gateway у її таблиці маршрутів

В Те, що в ній стоїть балансувальник

Г Розмір /24

Відповідь: Б. Публічність вирішує один рядок у таблиці маршрутів.

? У скількох зонах доступності може жити одна підмережа?

- **А** Лише в одній
- **Б** У двох
- **В** У всіх зонах регіону
- **Г** У жодній, підмережі глобальні

Відповідь: А. Підмережа — «квартал» одного району. Для надійності роблять однакові підмережі в кількох AZ.

? Скільки адрес AWS резервує в кожній підмережі?

- **А** 0
- **Б** 1
- **В** 5
- **Г** 256

Відповідь: В. Перші чотири й остання. У /24 вільних лишається 251.

? Сервери в приватній підмережі мають завантажувати оновлення з інтернету, лишаючись недосяжними ззовні. Що потрібно?

А Перенести їх у публічну підмережу

Б NAT Gateway у публічній підмережі й маршрут 0.0.0.0/0 на нього з приватної

В Elastic IP на кожен сервер

Г Egress-only Internet Gateway для IPv4

Відповідь: Б. NAT — виїзд з односторонньою смугою: назовні можна, ззовні — ні. Egress-only — аналог для IPv6.

? Чому для високої доступності ставлять NAT Gateway у кожній AZ?

А. Бо один NAT не вміє працювати з більше ніж одним сервером

Б. Щоб збій однієї зони не відрізав інтернет серверам в інших і не було платного трафіку між зонами

В. Щоб було дешевше

Г. Так вимагає закон

Відповідь: Б. Один NAT на всю VPC — єдина точка відмови плюс платні «переїзди» між районами.

? Пакет іде на 10.0.2.5. У таблиці: 10.0.0.0/16 → local і 0.0.0.0/0 → IGW. Куди він піде?

- **А** local — перемагає найточніший маршрут
- **Б** В інтернет, бо 0.0.0.0/0 охоплює все
- **В** Нікуди, маршрути конфліктують
- **Г** Випадково в один з двох

Відповідь: А. Longest prefix match: /16 точніший за /0.

✅ **Головне**

VPC — власне «огороджене містечко» в регіоні: CIDR від `/16` до `/28`, без перетинів з іншими мережами.

Підмережа живе в **одній AZ**; кожен вид підмереж — у 2+ AZ; **5 адрес** у кожній зарезервовано.

Internet Gateway — ворота; **route table** — вказівники; `0.0.0.0/0 → IGW` робить підмережу **публічною**; найточніший маршрут перемагає.

Балансувальники й NAT — у публічних підмережах; сервери й бази — у **приватних**.

NAT Gateway — вихід назовні без входу ззовні; платний за годину й ГБ; **по одному на AZ**; для IPv6 — **egress-only IGW**. Усі публічні IPv4 (зокрема Elastic IP) платні.

🚧 Охоронці мережі: Security Groups і Network ACL

👴 Простими словами

Є два охоронці. Security Group стоїть біля дверей кожного будинку і пам'ятає, кого впустив — тому випускає його назад. NACL стоїть на шлагбаумі кварталу і нікого не пам'ятає — перевіряє кожного в обидва боки.

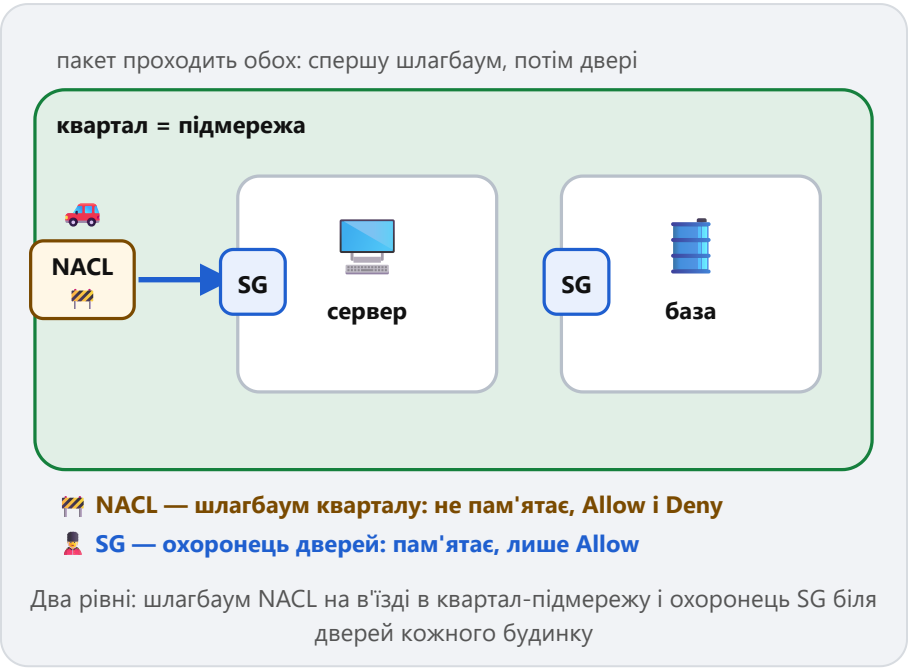
🎯 Що ти дізнаєшся

- чим Security Group відрізняється від Network ACL — і чому саме це питання обов'язково буде на іспиті
- що означає stateful і stateless на прикладі одного пакета
- як збудувати «ланцюжок довіри» балансувальник → сервер → база
- що таке ephemeral ports і чому про них забувають
- як заблокувати конкретну IP-адресу

🥐 Містечко збудоване, але поки в нього можна постукати звідусіль. Тарас формулює правила: «Клієнти — лише на сайт. До бази — тільки наші сервери. А той набридливий бот з однієї адреси, що сотнями разів на хвилину пробує паролі, — геть назавжди». Для цього в AWS є два охоронці. Вони працюють по-різному, і плутати їх не можна.

Два охоронці на різних рівнях

- **Security Group** («секьюріті груп», SG) — **охоронець біля дверей конкретного будинку** (сервера, бази, балансувальника). Точніше — біля мережевого інтерфейсу ресурсу.
- **Network ACL** («нетворк ей-сі-ел», NACL, network access control list) — **шлагбаум на в'їзді в квартал** (підмережу). Перевіряє всіх, хто в'їжджає в квартал і виїжджає з нього.



Два рівні: шлагбаум NACL на в'їзді в квартал-підмережу і охоронець SG біля дверей кожного будинку

Пакет, що йде ззовні до сервера, проходить **обох**: спершу шлагбаум кварталу, потім охоронця будинку.

Security Group: охоронець, що пам'ятає

Правила Security Group:

- **лише дозволи** (Allow). **Заборон (Deny) у SG не буває** — усе, що не дозволено, і так заборонено;
- окремо правила на **вхід** (inbound) і на **вихід** (outbound). За замовчуванням вхід — нікому, вихід — усім;

- у правилі вказують **порт** і **джерело**: діапазон адрес (`0.0.0.0/0`, `203.0.113.10/32`) або **іншу Security Group**.

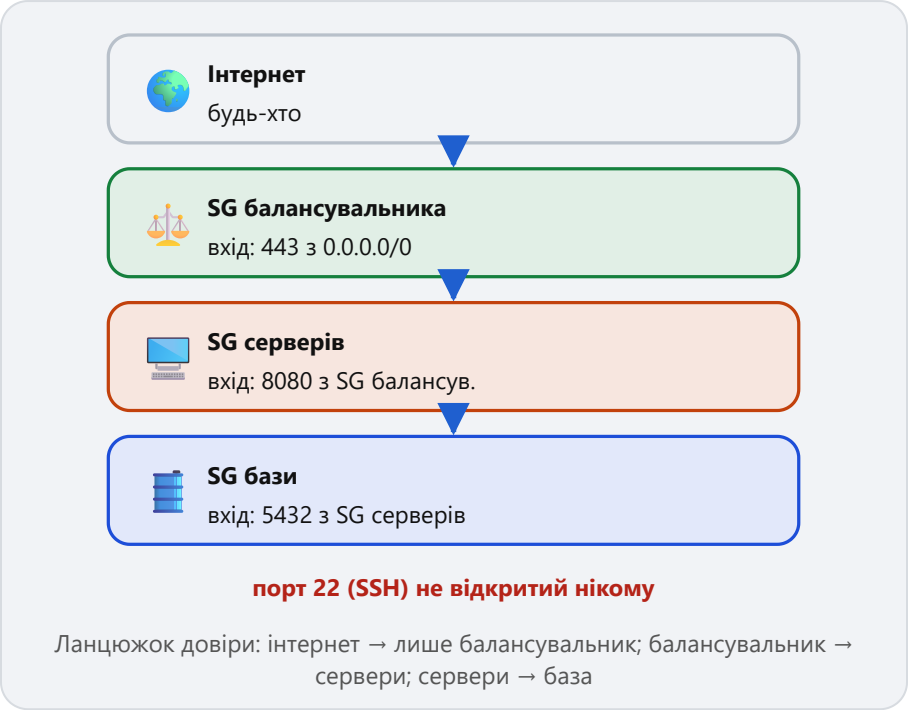
А головне — SG **stateful** («стейтфул», пам'ятає стан). Охоронець **пам'ятає**, хто вийшов з будинку, і сам пускає відповідь назад. Якщо з'єднання дозволене в один бік, відповідь пропускається автоматично — окремих правил для неї не треба.

Ланцюжок довіри: «пускати всіх з бейджем»

Найкорисніша можливість SG — вказати джерелом **іншу SG**: «пускати всіх, хто носить бейдж охоронної групи вебсерверів». Не треба знати їхні IP-адреси: сервери додаються й зникають, а правило працює.

Для «Хмаринки» це виглядає так:

Security Group	Вхід дозволено	Звідки
SG балансувальника	порт 443 (HTTPS)	`0.0.0.0/0` — увесь інтернет
SG серверів застосунку	порт 8080	SG балансувальника
SG бази PostgreSQL	порт 5432	SG серверів застосунку



Зверни увагу: порт 22 (SSH) не відкрито **нікому**. Адміни заходитимуть без нього — як саме, дізнаєшся в наступному розділі.

🎯 «База має приймати з'єднання лише від вебсерверів» → **SG бази з джерелом = SG вебсерверів**.

Network ACL: шлагбаум з пронумерованими правилами

Правила NACL влаштовані інакше:

- є і **дозволи (Allow), і заборони (Deny)**;

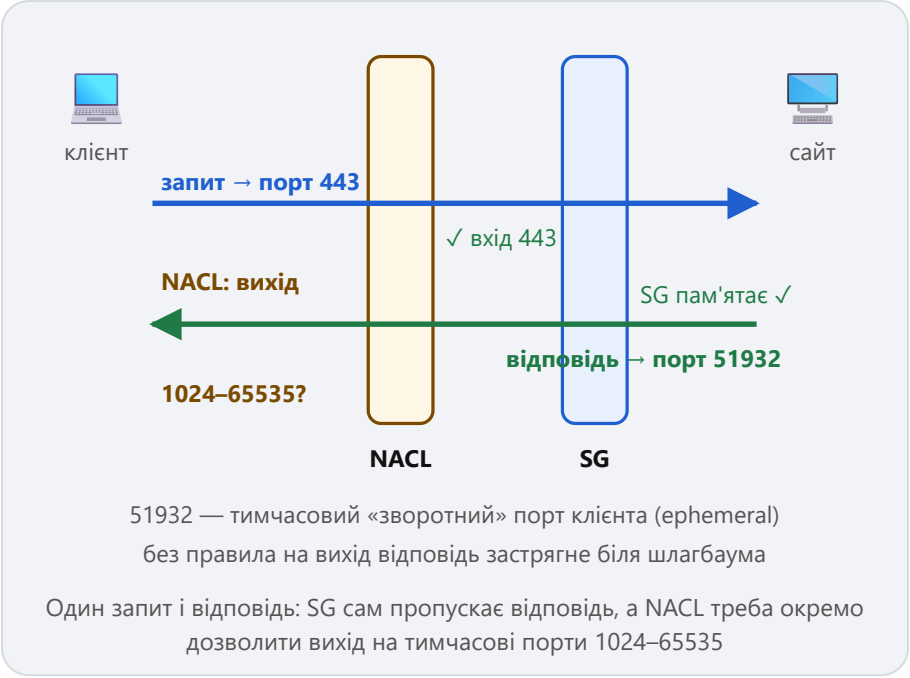
- правила **пронумеровані** й перевіряються **від меншого номера до більшого**; спрацьовує **перше збігле**, решта ігнорується;
- наприкінці завжди стоїть правило-зірочка `*`: «усе інше — заборонити»;
- один NACL може обслуговувати кілька підмереж, але в підмережі NACL рівно один. Стандартний (default) NACL пропускає все, а новостворений — нічого, доки не додаси правила.

А головне — NACL **stateless** («стейтлес», не пам'ятає). Шлагбаум перевіряє **кожну машину в обидва боки окремо** і не пам'ятає, що ця ж машина щойно в'їхала. Тому дозволяти треба **і вхід, і відповідь**.

Ephemeral ports: зворотна адреса відповіді

Щоб зрозуміти, про яку відповідь мова, простежимо один запит.

Клієнт стукає на сайт — на **порт 443**. Але відповідь має повернутися клієнтові не на 443, а на тимчасовий номер «дверей», який його комп'ютер вибрав випадково для цієї розмови, — **ephemeral port** («іфемерал порт», тимчасовий порт), десь у діапазоні **1024–65535**.



51932 — тимчасовий «зворотний» порт клієнта (ephemeral)
без правила на вихід відповідь застрягне біля шлагбаума

Один запит і відповідь: SG сам пропускає відповідь, а NACL треба окремо дозволити вихід на тимчасові порти 1024–65535

- **Security Group** запам'ятала з'єднання і сама випустить відповідь.
- **NACL** не пам'ятає нічого: якщо в нього немає правила «вихід на порти 1024–65535 дозволено», відповідь залишиться за шлагбаумом. Сайт «не відповідає», хоча вхід на 443 дозволено.

⚠️ Класична пастка: після налаштування NACL сайт «зависає». Найімовірніше, не дозволено **вихід на ephemeral ports 1024–65535**.

Як заблокувати конкретну адресу

У SG заборон немає, тому заблокувати одну набридливу адресу SG **не може**. Варіанти:

- **NACL з правилом Deny** з номером **меншим**, ніж дозвільні правила (бо спрацьовує перше збігле);
- **AWS WAF** — якщо мова про запити до сайту (рівень HTTP).

Приклад NACL «Хмаринки»:

№	Тип	Порт	Джерело	Дія
90	HTTPS	443	`198.51.100.23/32`	**DENY**
100	HTTPS	443	`0.0.0.0/0`	ALLOW
*	усе	усі	`0.0.0.0/0`	DENY

Бот з адреси `198.51.100.23` упреться в правило 90. А якби заборона мала номер 200, першим спрацювало б правило 100 — і бот пройшов би.

Порівняння в одній таблиці

	Security Group	Network ACL
Рівень	ресурс (двері будинку)	підмережа (в'їзд у квартал)
Пам'ять	**stateful** — відповідь пускає сам	**stateless** — дозволяй обидва боки
Правила	лише **Allow**	**Allow і Deny**
Порядок	усі правила разом	за **номером**, перше збігле
Джерело	адреси або **інша SG**	лише адреси
Для чого	основний захист завжди	додатковий шар, блокування адрес

Мнемоніка з простого курсу: **SG — Stateful, лише Allow. NACL — No memory, Numbered, може Deny.**

📖 **Словничок**

Security Group, SG («секьюріті груп») — охоронець ресурсу: лише Allow, stateful, джерелом може бути інша SG.

Network ACL, NACL («нейсл», «ей-сі-ел») — шлагбаум підмережі: Allow і Deny, stateless, правила за номером.

Stateful / stateless — пам'ятає з'єднання й сам пускає відповідь / перевіряє кожен напрямок окремо.

Inbound / outbound — правила на вхід / на вихід.

Ephemeral ports («іфемерал портс») — тимчасові порти відповіді, 1024–65535.

Правило-зірочка `*` — останнє правило NACL: «решту заборонити».

🧩 **Міні-тест: перевір себе**

? Потрібно заблокувати одну конкретну IP-адресу, що атакує сервери підмережі. Що використати?

А Security Group з правилом Deny

Б Network ACL з правилом Deny (з меншим номером, ніж дозвільні)

В Internet Gateway

Г Route table

Відповідь: Б. У SG заборон немає. Блокують NACL (або WAF для HTTP).

? База даних має приймати з'єднання лише від серверів застосунку. Як налаштувати?

А NACL з переліком IP-адрес усіх серверів

Б SG бази з джерелом = SG серверів застосунку

В Відкрити порт бази для 0.0.0.0/0

Г Покласти базу в публічну підмережу

Відповідь: Б. Посилання на SG — «пускати всіх з бейджем». Сервери можуть з'являтися й зникати, а правило працює.

? Що означає, що Security Group — stateful?

А Вона пам'ятає дозволене з'єднання й сама пропускає відповідь

Б Вона зберігає файли

В Вона працює лише вдень

Г Вона перевіряє кожен напрямок окремо

Відповідь: А. Stateful — «охоронець з пам'яттю». Stateless (NACL) — без пам'яті.

? Після налаштування NACL сайт перестав відповідати, хоча вхід на 443 дозволено. Що найімовірніше?

А Зламався Internet Gateway

Б Не дозволено вихід на тимчасові порти 1024–65535

В Security Group заблокувала відповідь

Г Закінчився домен

Відповідь: Б. NACL не пам'ятає з'єднань: відповідь іде на ephemeral port, і її треба дозволити окремо.

? Правила NACL: 100 — Allow 443 з 0.0.0.0/0; 200 — Deny 443 з 198.51.100.23. Чи пройде 198.51.100.23 на 443?

- **А** Так: першим спрацює правило 100
- **Б** Ні: Deny завжди перемагає
- **В** Ні, бо 200 більше
- **Г** Залежить від Security Group

Відповідь: А. У NACL важливий порядок: перевіряють від меншого номера, спрацьовує перше збігле. Deny треба ставити раніше.

? На якому рівні працює Security Group?

- **А** На рівні регіону
- **Б** На рівні підмережі
- **В** На рівні ресурсу (його мережевого інтерфейсу)
- **Г** На рівні акаунта

Відповідь: В. SG — «двері будинку»; NACL — «в'їзд у квартал».

✅ **Головне**

SG — охоронець ресурсу: **stateful**, ***лише Allow***, джерелом може бути **інша SG** («ланцюжок довіри»).

NACL — шлагбаум підмережі: **stateless**, **Allow і Deny**, правила **за номером**, перше збігле перемагає, у кінці `*` — заборонити все.

NACL вимагає дозволити **відповідь на ephemeral ports 1024–65535**.

Заблокувати адресу → **NACL Deny** (або WAF); у SG заборон немає.

Мнемоніка: **SG — Stateful, *лише Allow*. NACL — No memory, Numbered, *може* Deny.**

🛣️ Приватні дороги: VPC endpoints, PrivateLink, Flow Logs і вхід адміна

👴 Простими словами

Іноді не хочеться їхати до сервісу AWS через інтернет. VPC endpoint — це приватна дорога прямо з твого містечка до сервісу, без виходу на загальну трасу.

🎯 Що ти дізнаєшся

- чому сервери у VPC ходять до S3 та інших сервісів «через інтернет» і чим це погано
- що таке gateway endpoint і interface endpoint і коли який
- як надати свій сервіс іншим мережам через PrivateLink
- навіщо VPC Flow Logs
- як адміністратор безпечно заходить на приватні сервери без відкритих портів

🥐 Сервери «Хмаринки» в приватних підмережах щодня вивантажують сотні фото тортів у сховище S3. Наприкінці місяця Оля бачить у рахунку дивний рядок: «NAT Gateway — обробка даних». «Чому ми платимо за кожен гігабайт, якщо і сервери, і сховище — в AWS?» — «Бо зараз фото їдуть до S3 через NAT, ніби через інтернет, — каже Тарас. — Прокладемо приватну дорогу».

Проблема: сервіси AWS живуть «за парканом»

Твої сервери живуть у VPC, а багато сервісів AWS — **S3, DynamoDB, SQS, KMS** та інші — живуть **поза нею**, на публічних адресах. Щоб приватний сервер дістався до S3, він за замовчуванням їде через **NAT Gateway** на «загальну трасу».

Це погано з двох причин:

- **гроші** — NAT бере плату за кожен гігабайт;
- **безпека** — трафік виходить з твого огородженого містечка.

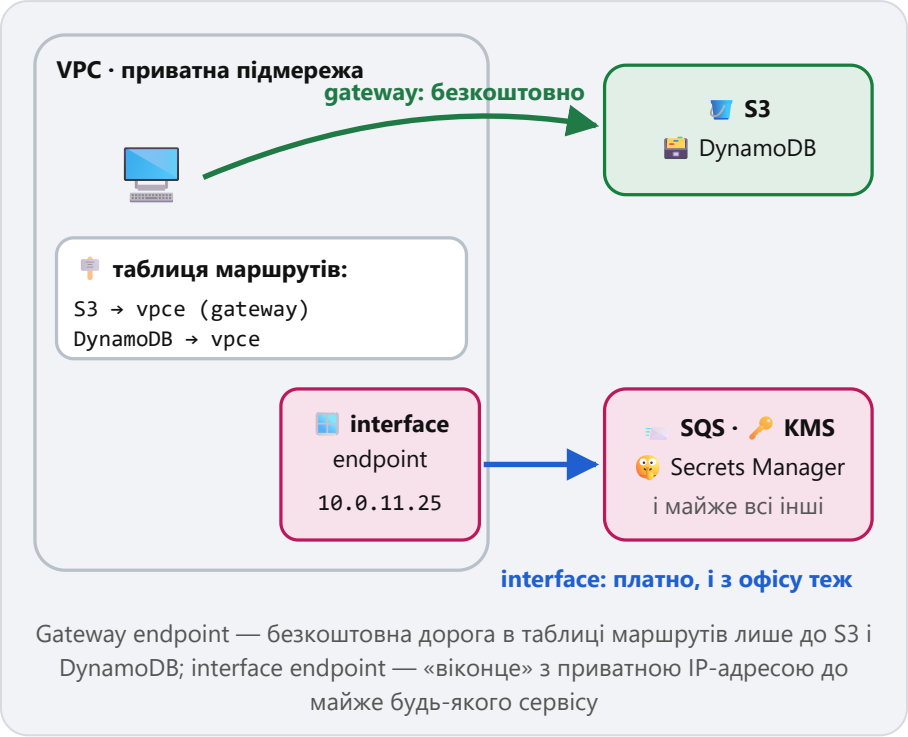
Рішення — **VPC endpoints** («ві-пі-сі ендпойнтс»): **приватні дороги** від VPC до сервісів AWS, без інтернету і без NAT.

Gateway endpoint: безкоштовна дорога до двох складів

Gateway endpoint («гейтвей ендпойнт») — найпростіший вид:

- існує лише для **двох** сервісів: **S3 і DynamoDB**;
- **безкоштовний**;
- працює як **новий рядок у таблиці маршрутів**: «до S3 — ось цією приватною дорогою»;
- працює лише для ресурсів **усередині VPC** — з офісу через VPN чи Direct Connect ним скористатися не можна.

Мнемоніка з простого курсу: gateway endpoint — лише «**SD**»: **S**3 і **D**ynamoDB.



Gateway endpoint — безкоштовна дорога в таблиці маршрутів лише до S3 і DynamoDB; interface endpoint — «віконце» з приватною IP-адресою до майже будь-якого сервісу

Interface endpoint: віконце в стіні містечка

Interface endpoint («інтерфейс ендпойнт», технологія **PrivateLink**) — це **мережевий інтерфейс з приватною IP-адресою у твоїй підмережі**, за яким «сидить» потрібний сервіс. Образ — **віконце видачі** просто у твоєму кварталі.

- Існує для **майже всіх** сервісів: SQS, SNS, KMS, Secrets Manager, CloudWatch, ECR, STS і навіть S3.
- **Платний**: за годину в кожній AZ і за гігабайт.
- Доступний і **з офісу** через VPN чи Direct Connect.

- Завдяки «приватному DNS» звичне ім'я сервісу автоматично веде на приватну адресу віконця — код програми міняти не треба.

	Gateway endpoint	Interface endpoint
Сервіси	лише S3 і DynamoDB	майже всі
Як працює	рядок у таблиці маршрутів	мережевий інтерфейс з приватною IP
Ціна	безкоштовно	погодинно + за ГБ
З офісу (VPN/DX)	ні	так

🎯 Приватний сервер → S3 або DynamoDB, дешево, без інтернету → **gateway endpoint**. Приватний сервер → SQS, KMS, Secrets Manager → **interface endpoint**. Великий рахунок за NAT через трафік до S3 → **gateway endpoint для S3**.

Політики endpoint і «лише через мою дорогу»

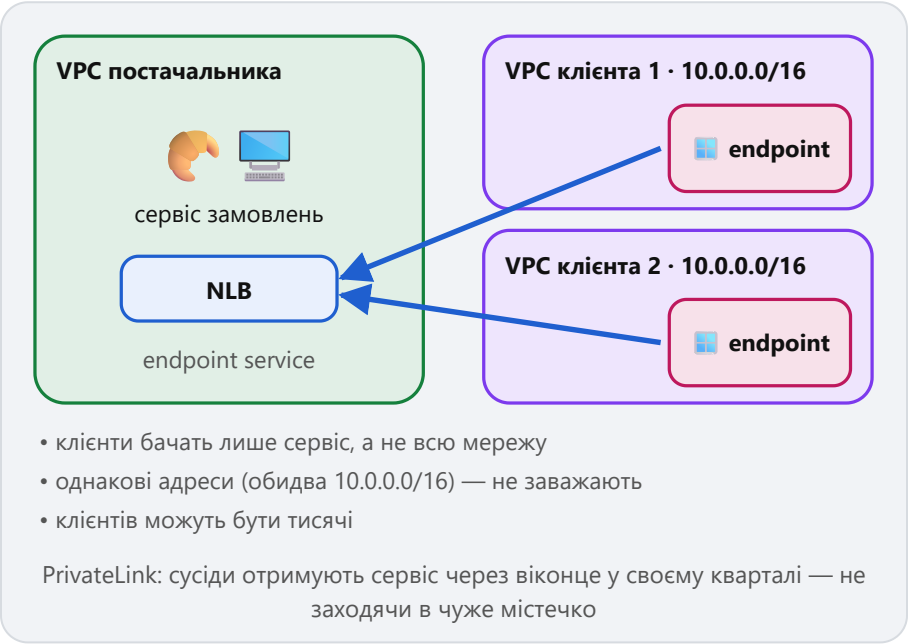
На endpoint можна почепити **endpoint policy** — обмеження, що можна робити через цю дорогу: наприклад, «лише наші сховища, а не будь-які сховища світу». Це захищає від витоку даних у чуже сховище.

А з іншого боку, у **bucket policy** сховища можна написати: «пускати **лише** через endpoint `vpce-…`» (умова `aws:SourceVpce`) чи «лише з нашої VPC» (`aws:SourceVpc`).

PrivateLink для власних сервісів

Та сама технологія PrivateLink дозволяє **надати власний сервіс** іншим VPC — навіть в інших акаунтах:

1. **Постачальник** ставить свій сервіс за **балансувальником NLB** і створює **endpoint service**.
2. **Споживач** створює в своїй VPC **interface endpoint** до цього сервісу.
3. Споживач звертається до сервісу через своє «віконце», не заходячи в містечко постачальника.



Переваги: доступ **односторонній** (споживач бачить лише сервіс, а не всю мережу), **адреси мереж можуть**

перетинатися, і споживачів можуть бути **тисячі**.

> 🎯 «Надати свій сервіс сотням клієнтських VPC приватно, діапазони адрес перетинаються» → **PrivateLink (NLB + endpoint service)**.

VPC Flow Logs: журнал шлагбаума

VPC Flow Logs («флоу логз») — **журнал камер біля шлагбаума**: він записує номери машин і час, але не заглядає в кузов. Технічно — метадані кожного мережевого потоку:

- звідки й куди (адреси й порти), протокол;
- скільки байтів;
- і головне — **ACCEPT** чи **REJECT**: пропустили чи відхилили.

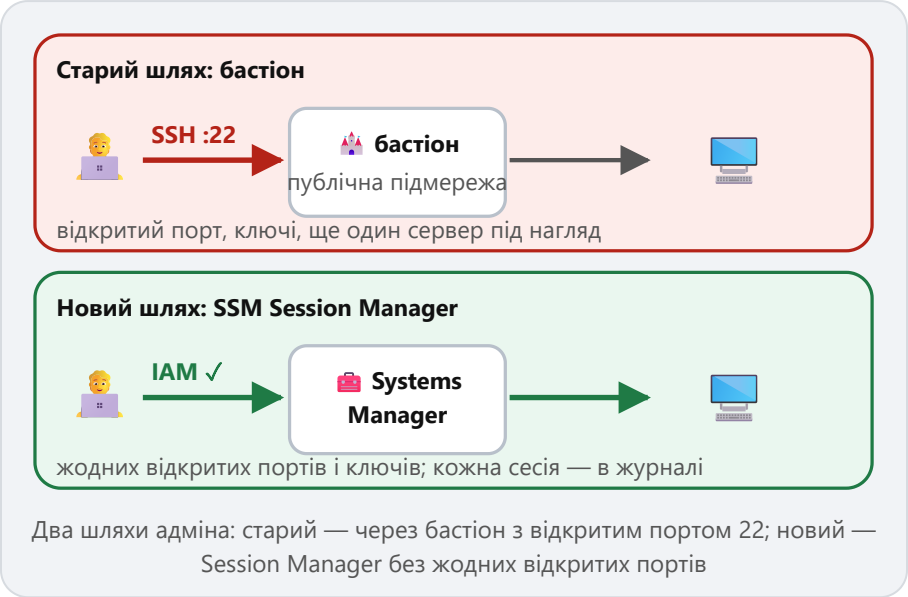
Журнал можна вести для всієї VPC, підмережі чи окремого інтерфейсу й складати в **CloudWatch Logs**, **S3** або **Firehose**. Вміст пакетів Flow Logs **не пишуть** (для цього є окремий сервіс Traffic Mirroring).

Навіщо? Шукати, **чому** не проходить з'єднання (його відхиляє SG чи NACL?), і аналізувати, **хто стукає** в сервери. На Flow Logs, до речі, спирається й GuardDuty.

Як адміну зайти на приватний сервер

Сервери в приватних підмережах недосяжні ззовні — це добре. Але як тоді адміністраторові полагодити щось усередині?

- **Bastion host** («бастіон», jump box) — класичний спосіб: окремий маленький сервер у **публічній** підмережі з SSH, відкритим лише з адреси офісу. Адмін заходить на бастіон, а з нього — на приватні сервери. Працює, але бастіон треба латати й охороняти, і порт 22 все одно відкритий.
- **SSM Session Manager** («сешн менеджер», з набору Systems Manager) — сучасний спосіб: вхід у консоль сервера **через браузер чи CLI**, **без відкритих портів**, **без бастіону** і **без SSH-ключів**. Хто може заходити — вирішує IAM, а **кожна сесія записується**.
- **EC2 Instance Connect Endpoint** — SSH до приватних серверів без публічних адрес через спеціальний endpoint.



Два шляхи адміна: старий — через бастіон з відкритим портом 22; новий — Session Manager без жодних відкритих портів

🎯 «Адмін-доступ до приватних серверів без відкритого порту 22, без бастіону, з записом сесій» → **SSM Session Manager**.

📖 Словничок

VPC endpoint — приватна дорога від VPC до сервісу AWS без інтернету й NAT.

Gateway endpoint — безкоштовний, лише S3 і DynamoDB, рядок у таблиці маршрутів, не з офісу.

Interface endpoint / PrivateLink — мережевий інтерфейс з приватною IP до майже будь-якого сервісу; платний; доступний з офісу.

Endpoint policy — що дозволено робити через endpoint.

aws:SourceVpce / aws:SourceVpc — умови bucket policy «лише через цей endpoint / з цієї VPC».

Endpoint service — власний сервіс за NLB, наданий іншим через PrivateLink.

VPC Flow Logs — журнал метаданих трафіку: хто, куди, ACCEPT чи REJECT.

Bastion host («бастіон») — проміжний сервер для входу адмінів.

SSM Session Manager — вхід на сервер без відкритих портів і ключів, із записом сесій.

🧩 **Міні-тест: перевір себе**

? Приватні сервери пишуть багато даних у S3 через NAT Gateway, і рахунок за NAT великий. Найдешевше рішення?

А Перенести сервери в публічну підмережу

Б Gateway endpoint для S3

В Interface endpoint для S3

Г Другий NAT Gateway

Відповідь: Б. Gateway endpoint до S3 безкоштовний і прибирає плату NAT за кожен гігабайт.

? Приватний сервер має звертатися до SQS, не виходячи в інтернет. Що обрати?

А Gateway endpoint

Б Interface endpoint (PrivateLink)

В Internet Gateway

Г Elastic IP

Відповідь: Б. Gateway endpoint є лише для S3 і DynamoDB. Для SQS — interface endpoint.

? Для яких двох сервісів існує gateway endpoint?

А SQS і SNS

Б EC2 і RDS

В S3 і DynamoDB

Г KMS і Secrets Manager

Відповідь: В. Мнемоніка «SD»: S3 і DynamoDB.

? Компанія хоче надати свій сервіс сотням клієнтських VPC приватно, а діапазони адрес у клієнтів перетинаються. Що обрати?

- **А** VPC peering з кожним клієнтом
- **Б** PrivateLink: NLB + endpoint service
- **В** Публічний Elastic IP
- **Г** Спільну підмережу

Відповідь: Б. PrivateLink дає односторонній доступ до одного сервісу, і перетин адрес йому не заважає.

? Треба з'ясувати, хто стукає в сервери і чиї з'єднання відхиляються. Що увімкнути?

- **А** VPC Flow Logs
- **Б** Macie
- **В** CloudFront
- **Г** NAT Gateway

Відповідь: А. Flow Logs записують метадані трафіку й результат ACCEPT/REJECT.

? Адмінам потрібен доступ до приватних серверів без відкритого порту 22 і з записом усіх сесій. Що обрати?

- **А** Bastion host з SSH
- **Б** Публічну IP-адресу на кожен сервер
- **В** SSM Session Manager
- **Г** Правило NACL на порт 22

Відповідь: В. Session Manager — без відкритих портів, бастіону й ключів; доступ через IAM, сесії записуються.

✅ **Головне**

Сервіси AWS живуть поза VPC; щоб не їхати до них через NAT та інтернет, будують **VPC endpoints**.

Gateway endpoint — безкоштовний, лише **S3 і DynamoDB**, рядок у таблиці маршрутів, не з офісу. **Interface endpoint (PrivateLink)** — майже всі сервіси, приватна IP у підмережі, платний, доступний з офісу.

Endpoint policy обмежує дорогу; bucket policy з **aws:SourceVpce** пускає лише через неї.

PrivateLink надає власний сервіс (за **NLB**) тисячам VPC, навіть з перетином адрес.

VPC Flow Logs — хто, куди, ACCEPT/REJECT (без вмісту). Адміни — через **SSM Session Manager** без відкритих портів.

🌉 Як з'єднати мережі: peering, Transit Gateway, VPN і Direct Connect

👴 Простими словами

Мережі можна з'єднати по-різному: місток між двома містечками (peering), великий вокзал для багатьох (Transit Gateway), захищений тунель через інтернет (VPN) або власна виділена лінія (Direct Connect).

🎯 Що ти дізнаєшся

- як з'єднати дві VPC мостом peering і чому він «не транзитивний»
- коли потрібен Transit Gateway — транспортна розв'язка на сотні мереж
- як з'єднати офіс з AWS — через VPN за хвилини чи через Direct Connect за тижні
- чому Direct Connect сам не шифрує і як його зарезервувати
- як офіс і хмара «бачать» імена одне одного — гібридний DNS

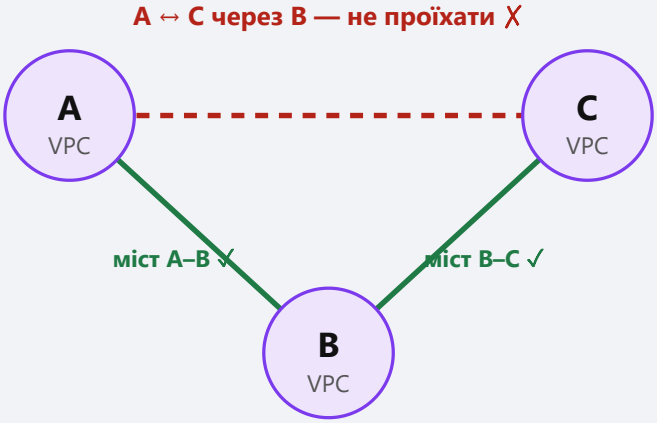
🥐 У «Хмаринки» вже три VPC: сайт, аналітика й тести. А ще — мережа самої пекарні з касами і друга точка на іншому кінці міста. Каси мають бачити сервер замовлень, аналітика — брати дані з сайту, а тести — ні в якому разі не чіпати робочу базу. «Будуватимемо мости й розв'язки», — каже Тарас.

VPC Peering — міст між двома містечками

VPC Peering («ві-пі-сі пірінг») — **міст** між двома VPC. Після нього сервери обох мереж спілкуються приватними адресами, ніби вони в одній мережі.

- Працює між VPC в **одному** акаунті, у **різних** акаунтах і навіть у **різних регіонах**.
- Потрібно оновити **таблиці маршрутів з обох боків**: «до 10.1.0.0/16 — через міст».
- Діапазони адрес **не можуть перетинатися** — пам'ятаєш про два міста з однаковими вулицями.
- Дешево: платиш лише за трафік між зонами чи регіонами.

А головна особливість — **peering не транзитивний**:



Кожній парі — окремий міст. Адреси мереж не повинні перетинатися.

Peering не транзитивний: мости A–B і B–C не дають проїзду A–C — потрібен окремий міст

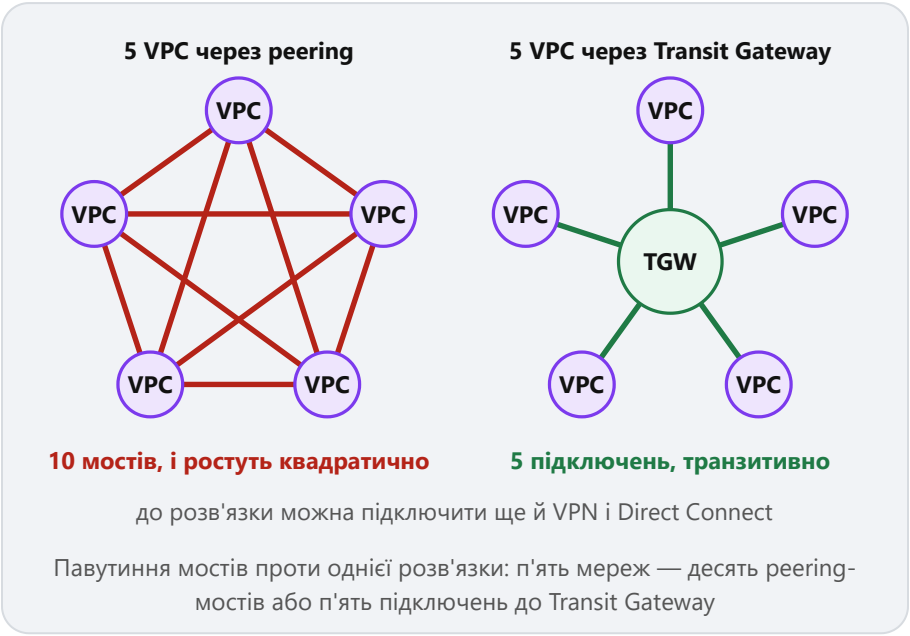
Якщо є міст A–B і міст B–C, машини з A **не можуть** проїхати в C через B. Кожній парі — свій міст. Для трьох мереж це три мости, для десяти — вже 45, і керувати цим павутинням стає неможливо.

Transit Gateway — транспортна розв'язка

AWS Transit Gateway («транзит гейтвей», TGW) — **велика транспортна розв'язка**, де сходяться дороги всіх містечок і офісів. Кожну мережу підключають **один раз** — до розв'язки, а далі вона везе куди треба.

- **Транзитивна** маршрутизація: A бачить C через розв'язку.

- До однієї розв'язки — **сотні й тисячі** VPC, а ще VPN і Direct Connect.
- Власні **таблиці маршрутів** розв'язки дозволяють ділити мережі на сегменти: «тести не бачать робочу базу».
- Між регіонами розв'язки з'єднують власним peering; ділитися розв'язкою з іншими акаунтами можна через **AWS RAM**.
- Уміє сумувати пропускну здатність кількох VPN-тунелів (ECMP) і розсилати трафік багатьом одержувачам (multicast).
- Платиш за **кожне підключення** щогодини і за **кожен гігабайт**, що проходить розв'язкою.



Павутиння мостів проти однієї розв'язки: п'ять мереж — десять peering-мостів або п'ять підключень до Transit Gateway

🎯 З'єднати 2–3 VPC просто й дешево → **VPC Peering**. Десятки чи сотні VPC плюс офіс, з транзитивною маршрутизацією → **Transit Gateway**.

З'єднати офіс з AWS: VPN

Мережа пекарні з касами — це **on-premises**. Найшвидший спосіб з'єднати її з AWS — **AWS Site-to-Site VPN** («сайт-ту-сайт ві-пі-ен»):

- зашифровані тунелі **IPsec** через звичайний **інтернет** — образ з простого курсу: **броньований фургон по загальній трасі**;

- з боку AWS — **Virtual Private Gateway** (VGW, приєднаний до однієї VPC) або Transit Gateway; з боку офісу — **Customer Gateway** (твій роутер);
- AWS завжди дає **два тунелі** — для резерву;
- до **1,25 Gbps** на тунель (з листопада 2025 на Transit Gateway можна «великі» тунелі до 5 Gbps);
- налаштовується **за хвилини**, коштує недорого.

Мінус один: тунель іде через інтернет, тож швидкість і затримка **нестабільні** — як дорога в годину пік.

Ще кілька варіантів VPN:

- **Client VPN** — для **ноутбуків окремих працівників**, що працюють з дому.
- **VPN CloudHub** — кілька філій з'єднуються з одним VGW і через нього спілкуються **між собою**.
- **Accelerated VPN** — тунель заходить у найближчу точку мережі AWS і далі їде швидкою приватною мережею (лише з Transit Gateway).

Direct Connect — власна залізниця до AWS

AWS Direct Connect («директ коннект», DX) — **виділена фізична лінія** від твого дата-центру до AWS через спеціальну точку обміну (Direct Connect location). Образ — **власна приватна залізниця**:

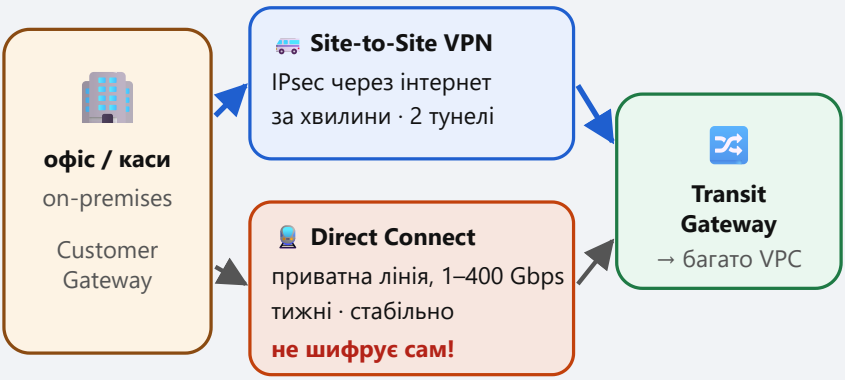
- швидкість **стабільна**, затримка передбачувана;

- **Dedicated** (виділене підключення): 1, 10, 100 або 400 Gbps; **Hosted** (через партнера): від 50 Mbps до 25 Gbps;
- вихідний трафік з AWS через DX **дешевший**, ніж через інтернет, — вигідно для великих обсягів;
- але прокладання займає **тижні або місяці**.

⚠️ Direct Connect **сам по собі не шифрує** трафік — це приватна лінія, але не сейф. Якщо потрібне шифрування: **IPsec VPN поверх Direct Connect** або **MACsec** (шифрування на рівні самої лінії).

Ще три речі про DX:

- **Direct Connect Gateway** — один DX до **багатьох VPC у різних регіонах**.
- Віртуальні інтерфейси (VIF): **private** — до VPC; **public** — до публічних сервісів AWS (наприклад, S3); **transit** — до Transit Gateway.
- **Резервування**: максимальна стійкість — **два DX у різних DX-локаціях**; дешевший резерв — **DX + Site-to-Site VPN**.



шифрування на DX → VPN поверх DX або MACsec

резерв DX → VPN (дешево) або другий DX в іншій локації

Офіс ↔ AWS: VPN через інтернет (швидко, дешево) або Direct Connect (стабільно, тижні); далі — Transit Gateway до багатьох VPC

Гібридний DNS

Коли офіс і хмара з'єднані, вони мають розуміти **імена** одне одного: касі треба знати адресу `orders.khmarynka.internal`, а серверу в AWS — `kasa1.office.local`. Це вміють **Route 53 Resolver endpoints**:

- **Inbound endpoint** («вхідний») — офісні сервери питають у нього імена з AWS;
- **Outbound endpoint** + правила пересилання — сервери у VPC питають імена з офісу.

Що обрати — одна таблиця

Задача	Рішення
2–3 VPC, просто й дешево	VPC Peering
Багато VPC + офіс, транзитивно	Transit Gateway
Один сервіс для тисяч чужих VPC, адреси перетинаються	PrivateLink
Канал до офісу вже сьогодні, недорого, з шифруванням	Site-to-Site VPN
Стабільна швидкість, великі обсяги, приватна лінія	Direct Connect
Direct Connect + шифрування	VPN поверх DX або MACsec
Дешевий резерв для Direct Connect	Site-to-Site VPN
Максимальна стійкість гібридного з'єднання	два DX у різних локаціях
Ноутбуки працівників	Client VPN
Філії спілкуються між собою через AWS	VPN CloudHub

🥐 Тарас з'єднав сайт і аналітику через Transit Gateway, а тести поклав в окрему таблицю маршрутів — до робочої бази їм не дотягнутися. Каси обох пекарень підключено через Site-to-Site VPN — за один вечір. «А коли будемо великими, прокладемо Direct Connect», — мріє Оля.

📖 Словничок

VPC Peering («пірінг») — міст між двома VPC; нетранзитивний; без перетину адрес.

Транзитивність — можливість проїхати з A в C через B.

Transit Gateway, TGW — «розв'язка» для сотень VPC, VPN і DX; транзитивна; платна за підключення й ГБ.

Site-to-Site VPN — IPsec-тунелі через інтернет; 2 тунелі; до 1,25 Gbps на тунель; за хвилини.

Virtual Private Gateway / Customer Gateway — кінець тунелю з боку AWS / з боку офісу.

Client VPN / VPN CloudHub / Accelerated VPN — для ноутбуків / філії між собою / через мережу AWS.

Direct Connect, DX — виділена приватна лінія до AWS: 1–400 Gbps, тижні на прокладання, без шифрування.

MACsec — шифрування на рівні лінії Direct Connect.

Direct Connect Gateway — один DX до VPC у багатьох регіонах.

VIF — віртуальний інтерфейс DX: private, public, transit.

Route 53 Resolver endpoints — inbound (офіс → імена AWS) і outbound (VPC → імена офісу).

🧩 **Міні-тест: перевір себе**

? VPC A з'єднана peering з B, а B — з C. Чи зможе A напряму спілкуватися з C через B?

- **А** Так, завжди
- **Б** Ні: peering не транзитивний, потрібен окремий міст A–C
- **В** Так, якщо всі VPC в одному регіоні
- **Г** Лише через інтернет

Відповідь: Б. Кожній парі — свій міст. Або одна розв'язка Transit Gateway для всіх.

? Треба з'єднати 40 VPC і офіс так, щоб маршрутизація була транзитивною. Що обрати?

А 780 peering-з'єднань

Б Transit Gateway

В Internet Gateway у кожній VPC

Г NAT Gateway

Відповідь: Б. Transit Gateway — розв'язка на сотні мереж: кожна підключається один раз.

? Зашифрований канал до офісу потрібен уже сьогодні й недорого. Що обрати?

- **А** Site-to-Site VPN
- **Б** Direct Connect
- **В** VPC Peering
- **Г** CloudFront

Відповідь: А. VPN налаштовується за хвилини й шифрує трафік IPsec. Direct Connect прокладають тижнями.

? Компанії потрібна стабільна швидкість 10 Gbps і щоденна передача великих обсягів приватною лінією. Що обрати?

- **А** Site-to-Site VPN
- **Б** AWS Direct Connect
- **В** Client VPN
- **Г** Transit Gateway без зовнішнього з'єднання

Відповідь: Б. Direct Connect — «власна залізниця»: стабільно, приватно, дешевший вихідний трафік.

? Є Direct Connect, але вимога — шифрувати весь трафік. Що зробити?

- **А** Нічого, DX шифрує сам
- **Б** Пустити IPsec VPN поверх Direct Connect (або увімкнути MACsec)
- **В** Перейти на peering
- **Г** Використати NAT Gateway

Відповідь: Б. Direct Connect приватний, але сам по собі не шифрує.

? Потрібен недорогий резерв для Direct Connect. Що обрати?

- **А** Другий Direct Connect у тій самій локації
- **Б** Site-to-Site VPN як резервний канал
- **В** CloudFront
- **Г** Нічого, DX не ламається

Відповідь: Б. VPN — дешевий запасний шлях. Максимальна стійкість — два DX у різних локаціях.

? Сервери в офісі мають знаходити за іменами приватні ресурси у VPC. Що налаштувати?

- **А** Route 53 Resolver inbound endpoint
- **Б** Route 53 Resolver outbound endpoint
- **В** Internet Gateway
- **Г** VPC Flow Logs

Відповідь: А. Inbound — «вхідні питання» з офісу до AWS. Outbound — навпаки, з VPC до офісу.

✅ **Головне**

VPC Peering — міст між двома VPC: **нетранзитивний**, адреси **не перетинаються**, дешевий; для 2–3 мереж.

Transit Gateway — розв'язка на сотні VPC, VPN і DX: **транзитивно**, сегментація таблицями, платно за підключення й ГБ.

Site-to-Site VPN — IPsec через інтернет, **за хвилини**, 2 тунелі, до 1,25 Gbps на тунель; **Client VPN** — для ноутбуків; **CloudHub** — філії між собою.

Direct Connect — приватна лінія 1–400 Gbps, **тижні** на прокладання, **не шифрує** (VPN поверх DX або MACsec); резерв — VPN або другий DX в іншій локації; **DX Gateway** — багато регіонів.

Гібридний DNS — **Route 53 Resolver**: inbound (офіс → AWS), outbound (AWS → офіс).

🧭 Route 53, CloudFront і Global Accelerator: швидко для всього світу

👴 Простими словами

Route 53 — телефонна книга, що перетворює назву сайту на адресу. CloudFront — мережа складів по світу з копіями твого сайту, щоб кожен отримував їх зі складу поруч, а не з-за океану.

🎯 Що ти дізнаєшся

- які бувають записи DNS і що таке Alias
- як Route 53 обирає, яку адресу дати саме тобі, — сім політик маршрутизації на прикладах
- навіщо перевірки здоров'я
- як CloudFront роздає копії контенту з кіосків біля людей і як закрити сховище від прямого доступу
- чим Global Accelerator відрізняється від CloudFront

🥐 Сайт «Хмаринки» живе у Франкфурті, і львів'яни задоволені. Але сестра Олі з Торонто скаржиться: сторінки з фото тортів відкриваються повільно. А ще Оля хоче, щоб адреса khmarynka.ua вела на новий сайт, і щоб нову версію спершу побачили лише 10% відвідувачів — «про всяк випадок». «Це три різні інструменти, — каже Тарас. — Довідкова, мережа магазинів і VIP-коридор».

Route 53 — довідкова інтернету

Amazon Route 53 («рут фіфті-срі»; 53 — номер порту DNS) — керований **DNS**-сервіс AWS. Він:

- **реєструє домени** (можна купити `khmarynka.ua` просто тут або керувати доменом, купленим деінде);
- тримає **записи** «ім'я → адреса» у **hosted zone** («хостед зоун»): **публічній** (для всього інтернету) чи **приватній** (лише всередині твоїх VPC);
- **перевіряє здоров'я** серверів і вміє не давати адресу хворого;
- має угоду про доступність **100%** — «довідкова» не має права мовчати.

Записи DNS

- **A** — ім'я → IPv4-адреса. **AAAA** — ім'я → IPv6.
- **CNAME** («сі-нейм») — ім'я → **інше ім'я** («shop.khmarynka.ua — це те саме, що www.khmarynka.ua»). ⚠️ CNAME **не можна** ставити на корінь домену — на сам `khmarynka.ua` (його називають **zone apex**, «апекс»).
- **Alias** («ейліас») — особливий запис AWS: ім'я → **ресурс AWS** (балансувальник, CloudFront, сховище-сайт S3, API Gateway). Працює **і на корені домену**, а запити до ресурсів AWS через Alias **безкоштовні**.
- **MX** — куди доставляти пошту; **TXT** — текстові позначки (наприклад, підтвердження власності на домен); **NS** — хто відповідає за зону.

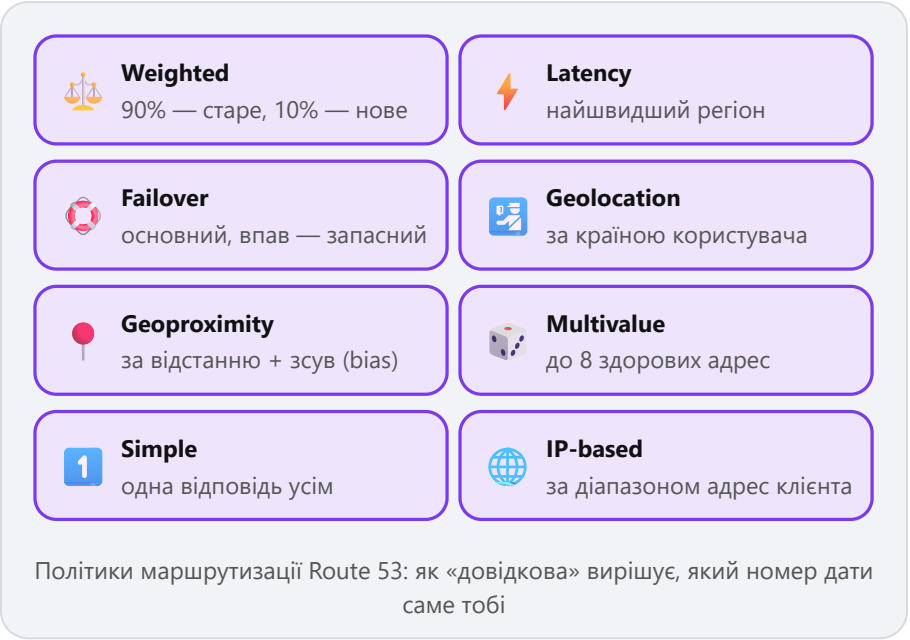
У кожного запису є **TTL** — скільки секунд відповідь можна пам'ятати (ти знаєш це з розділу про вебсайти).

> 🎯 «Направити корінь домену example.com на балансувальник ALB» → **Route 53 Alias record**. CNAME тут не підійде.

Сім способів обрати відповідь

Найцікавіше в Route 53 — **routing policies** («рутинг полісіз», політики маршрутизації): він може давати

різним людям **різні** відповіді на одне й те саме ім'я.



Політики маршрутизації Route 53: як «довідкова» вирішує, який номер дати саме тобі

- **Simple** — одна відповідь для всіх. Без перевірки здоров'я.
- **Weighted** («вейтед», зважена) — «**90% дзвінків — у старий офіс, 10% — у новий**». Для поступового випуску нової версії, A/B-тестів, blue/green.
- **Latency** («лейтенсі») — до регіону з **найменшою затримкою** для конкретного користувача: з Торонто — у Канаду, зі Львова — у Франкфурт.

- **Failover** («фейловер») — **основний** і **запасний**: поки перевірка здоров'я основного успішна, всім дають його; провалилася — запасний. Класика для аварійного відновлення «active-passive».
- **Geolocation** («джеолокейшн») — **за країною чи континентом** користувача, «за паспортом»: поляки бачать польську версію сайту, а з певних країн контент недоступний через ліцензії. Потрібен запис «за замовчуванням» для решти світу.
- **Geoproximity** («джеопроксиміті») — **за відстанню** до ресурсів, з можливістю **зсунути** межу (bias): «віддай Франкфурту більшу частку Європи».
- **Multivalue** («мульті-вел'ю») — до **8** здорових адрес у відповіді: клієнт обере сам. Простий розподіл, але **не заміна** балансувальнику.
- (Є ще **IP-based** — за діапазоном адрес клієнта, наприклад за провайдером.)

⚠️ Не плутай: **geolocation** — країна користувача. **Latency** — найшвидший регіон. **Geoproximity** — відстань плюс зсув.

Перевірки здоров'я

Health checks Route 53 — «чи живий сервер?». Перевіряльники стукають до адреси **з інтернету**, тож:

- перевіряти можна публічні адреси, інші перевірки (складена перевірка) чи **тривогу CloudWatch**;
- для **приватних** ресурсів — лише через тривогу CloudWatch, бо з інтернету до них не дістатися.

Популярний трюк: основний сайт упав → Route 53 failover автоматично віддає **запасну сторінку-заглушку**, яка лежить у сховищі S3.

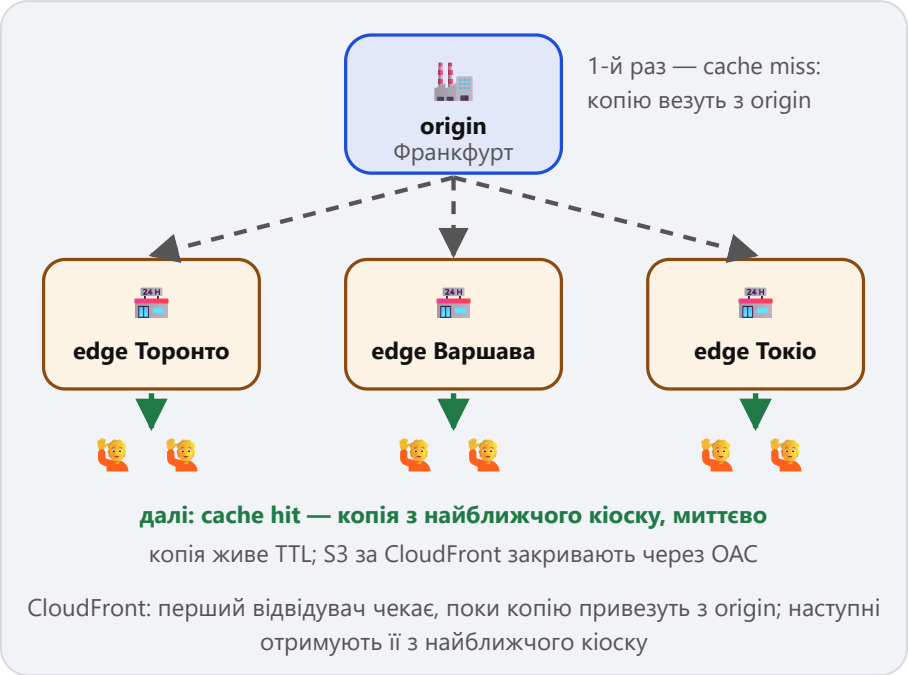
> ⚠️ Перемикання через DNS залежить від **TTL**: браузери й резолвери ще якийсь час пам'ятають стару адресу. Швидше перемикає Global Accelerator — про нього нижче.

CloudFront — мережа місцевих магазинів

Чому сестрі в Торонто повільно? Кожне фото летить з Франкфурта через океан. А що, як тримати **копії** фото поруч із нею?

Amazon CloudFront («клауд-фронт») — **CDN** (content delivery network — мережа доставки контенту). Образ з простого курсу — **мережа місцевих магазинів з**

копіями товару: береш у найближчому, а не їдеш на центральний склад.



Як це працює:

1. Твій оригінальний сайт чи сховище — це **origin** («оріджин», джерело): S3, балансувальник, сервер EC2, API Gateway чи навіть сервер у твоєму офісі.
2. Користувач звертається до **найближчої edge location** CloudFront.
3. Якщо копії там ще немає (**cache miss**) — edge бере її з origin і зберігає.

4. Наступні користувачі поруч отримують копію миттєво (**cache hit**).
5. Копія живе стільки, скільки дозволяє **TTL**. Щоб примусово оновити — **invalidation** (інвалідація), а ще краще — давати новим версіям файлів нові імена (`style.v2.css`).

CloudFront прискорює не лише статику, а й **динамічні** запити: вони їдуть від кіоску до origin швидкою приватною мережею AWS, а не загальною трасою. А ще CloudFront **знімає навантаження** з origin і **здешевлює** вихідний трафік.

CloudFront: що ще варто знати

- **Cache behaviors** — різні правила для різних шляхів: `/static/*` → зі сховища S3 з довгим кешем, `/api/*` → на балансувальник без кешу. Статика й API на одному домені.
- **OAC** («оу-ей-сі», Origin Access Control) — **склад S3 відчиняється лише кур'єрам CloudFront**: сховище лишається приватним, прямі посилання на нього не працюють. OAC — сучасна заміна старого **OAI** (підтримує шифрування KMS).

- **Signed URL** — «квиток» на **один** файл (наприклад, придбану книжку). **Signed cookies** — «абонемент» на **багато** файлів (увесь преміум-розділ з відео), без зміни посилань. Не плутай з **S3 presigned URL** — тимчасовим посиланням напряму на об'єкт S3.
- **Geo restriction** — дозволити чи заборонити цілі країни.
- **Origin failover** — група з основного й запасного origin: основний відповідає помилками — CloudFront бере із запасного.
- **CloudFront Functions** — дуже легкий код на edge (перенаправлення, заголовки, прості перевірки) за мікросекунди. **Lambda@Edge** — важча логіка з мережевими викликами.
- **HTTPS** — сертифікат ACM у регіоні **us-east-1**; інтеграція з **WAF** і **Shield**.
- **Price classes** — обмежити набір edge-локацій, щоб платити менше.

🎯 «Сховище S3 має бути доступне лише через CloudFront» → **OAC**. «Статичний сайт з HTTPS і власним доменом» → **S3 + CloudFront (OAC) + ACM + Route 53**.

Global Accelerator — VIP-коридор в аеропорту

AWS Global Accelerator («глобал акселерейтор») дає застосунку **дві статичні anycast IP-адреси** на весь світ. «Anycast» означає: ті самі дві адреси «живуть» у сотнях точок, і користувач автоматично потрапляє в **найближчу**. Далі трафік іде не загальною трасою інтернету, а **швидкою приватною мережею AWS** до твоїх серверів в одному чи кількох регіонах.

Образ з простого курсу — **VIP-коридор в аеропорту з двома постійними входами**: заходиш у найближчі двері, далі — швидка приватна доріжка.

- Працює з **TCP і UDP** — тобто з іграми, IoT, голосовим зв'язком, а не лише з вебсайтами.
- **Статичні IP** — зручно, коли клієнти додають твої адреси в «білий список».
- **Миттєве перемикання** між регіонами за перевірками здоров'я — без очікування DNS-кешу.
- **Нічого не кешує** — це не CDN.



• TCP і UDP · • нічого не кешує · • failover без DNS-кешу

образ: VIP-коридор в аеропорту з двома постійними входами

Global Accelerator: дві постійні адреси, вхід у найближчу точку і швидка приватна доріжка AWS до здорового регіону

CloudFront чи Global Accelerator?

	CloudFront	Global Accelerator
Що робить	кешує й роздає вміст	прискорює шлях до серверів
Кеш	так	**ні**
Протоколи	HTTP/HTTPS (+ WebSocket)	**TCP і UDP**
Адреси	змінні, за доменом	**2 статичні anycast IP**
Перемикання регіонів	через origin failover	миттєве за здоров'ям
Типові задачі	сайти, фото, відео, API	ігри, IoT, VoIP, статичні IP

Мнемоніка з простого курсу: **CloudFront — Кешує Контент (HTTP). Global Accelerator — Гоночна доріжка + статичні IP (TCP/UDP).**

🥐 Тарас поставив CloudFront перед сайтом: фото тортів тепер лежать у кіоску в Торонто, і сестра Олі бачить їх миттєво. Сховище з фото закрите через OAC. Домен khmarynka.ua веде на CloudFront через Alias, а нова версія сайту спершу отримала 10% відвідувачів через weighted routing — і лише після тижня без скарг — усіх.

📖 **Словничок**

Route 53 («рут фіфті-срі») — DNS AWS: домени, записи, перевірки здоров'я, SLA 100%.

Hosted zone — набір записів домену: публічна чи приватна (лише для VPC).

A / AAAA / CNAME / MX / TXT — IPv4 / IPv6 / інше ім'я / пошта / текстові позначки.

Zone apex («апекс») — корінь домену (`khmarynka.ua`); CNAME на ньому заборонений.

Alias («ейліас») — запис на ресурс AWS; працює на корені; безкоштовні запити.

Routing policies — simple, weighted, latency, failover, geolocation, geoproximity, multivalue, IP-based.

Health check — перевірка здоров'я; для приватних ресурсів — через тривогу CloudWatch.

CDN / CloudFront — мережа доставки контенту / сервіс AWS, що кешує копії на edge.

Origin («оріджин») — першоджерело вмісту для CloudFront.

Cache hit / miss, TTL, invalidation — знайшлося в кеші / ні, строк життя копії, примусове оновлення.

Cache behaviors — різні правила кешу й origin для різних шляхів.

OAC («оу-ей-сі») — сховище S3 доступне лише через CloudFront (заміна OAI).

Signed URL / signed cookies — доступ до одного файлу / до багатьох файлів через CloudFront.

CloudFront Functions / Lambda@Edge — легкий / важчий код на edge.

Global Accelerator — 2 статичні anycast IP, приватна мережа AWS, TCP/UDP, без кешу, миттєвий failover.

🧩 Міні-тест: перевір себе

? Корінь домену khmarynka.ua має вказувати на балансувальник ALB. Який запис використати?

- **А** CNAME
- **Б** Route 53 Alias
- **В** MX
- **Г** TXT

Відповідь: Б. CNAME на корені домену заборонений. Alias працює на корені й безкоштовний для ресурсів AWS.

? Нову версію сайту треба спершу показати 10% відвідувачів. Яка політика Route 53?

- **А** Weighted
- **Б** Geolocation
- **В** Latency
- **Г** Simple

Відповідь: А. Weighted — «90% в один бік, 10% — в інший».

? Користувачі з Польщі мають бачити польську версію сайту. Яка політика?

- **А** Latency
- **Б** Geolocation
- **В** Multivalue
- **Г** Failover

Відповідь: Б. Geolocation — за країною користувача. Latency — за швидкістю, а не за паспортом.

? Потрібно прискорити статичний і динамічний контент для клієнтів в Азії. Що обрати?

- **А** CloudFront
- **Б** Direct Connect
- **В** NAT Gateway
- **Г** Route 53 simple routing

Відповідь: А. CloudFront кешує статику на edge біля людей і прискорює динаміку через мережу AWS.

? Сховище S3 з фото має бути доступне лише через CloudFront, а не за прямими посиланнями. Що використати?

А Зробити сховище публічним

Б OAC (Origin Access Control)

В Signed cookies

Г Geo restriction

Відповідь: Б. OAC — «склад відчиняється лише кур'єрам CloudFront».

? Онлайн-гра працює на UDP, потрібні статичні IP-адреси для всього світу і миттєве перемикання між регіонами. Що обрати?

А CloudFront

Б AWS Global Accelerator

В Route 53 failover

Г Application Load Balancer

Відповідь: Б. Global Accelerator: TCP і UDP, дві статичні anycast IP, швидкий failover, без кешу.

? Підписники мають дивитися багато відео з преміум-розділу, не змінюючи посилань. Що обрати?

- **А** Signed URL для кожного відео
- **Б** CloudFront signed cookies
- **В** Публічне сховище
- **Г** S3 presigned URL

Відповідь: Б. Signed cookies — «абонемент» на багато файлів. Signed URL — «квиток» на один.

✅ **Головне**

Route 53 — DNS із реєстрацією доменів і перевірками здоров'я; **Alias** — на ресурси AWS, працює на **корені домену**, безкоштовно; **CNAME на корені не можна**.

Політики: **weighted** — частки (поступовий випуск), **latency** — найшвидший регіон, **failover** — основний/запасний, **geolocation** — країна, **geoproximity** — відстань + bias, **multivalue** — до 8 адрес.

Перемикання через DNS залежить від **TTL**; приватні ресурси перевіряють через **тривогу CloudWatch**.

CloudFront — CDN: кеш на edge, origin, cache behaviors, **OAC**, signed URL (один файл) / signed cookies (багато), сертифікат у **us-east-1**.

Global Accelerator — **2 статичні anycast IP**, **TCP/UDP**, **без кешу**, миттєвий failover регіонів.

🖥️ EC2: орендуємо віртуальний комп'ютер

👴 Простими словами

EC2 — це автопрокат комп'ютерів. Обираєш «модель» (скільки процесорів і пам'яті), вмикаєш на скільки треба і платиш, поки він працює.

🎯 Що ти дізнаєшся

- що таке EC2 і як крок за кроком запустити сервер
- що таке AMI — «зліпок» комп'ютера
- як читати назви типів на кшталт m7g.2xlarge і яке сімейство для чого
- життєвий цикл сервера і за що платиш у кожному стані
- user data, метадані, перевірки стану
- як розставити сервери «разом» чи «окремо» — placement groups

🥐 Мережа готова — час поселити в неї перший сервер для сайту «Хмаринки». «Як орендувати авто, — пояснює Тарас. — Обираєш модель, комплектацію, де паркувати — і платиш, поки їздиш». Оля сміється: «Тільки не кажи, що є ще й страховка». — «Є. Але про це згодом».

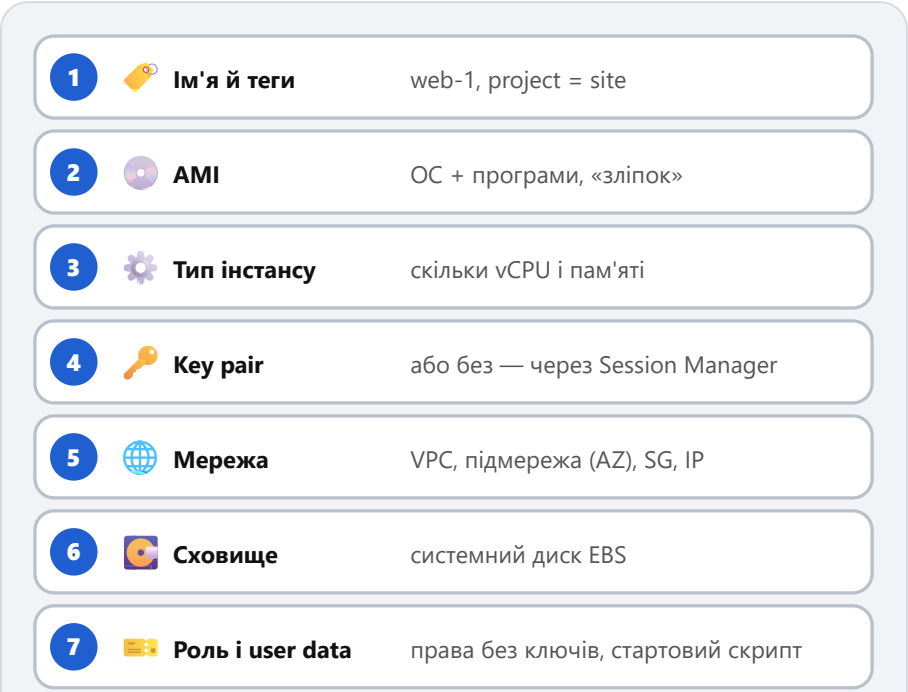
EC2 — автопрокат комп'ютерів

Amazon EC2 («і-сі-ту», Elastic Compute Cloud — еластична обчислювальна хмара) — сервіс оренди **віртуальних серверів**. Кожен окремий сервер називається **інстанс** (instance). Образ з простого курсу — **автопрокат**: обираєш модель, опції й платиш за час.

- Платиш **посекундно** (для Linux і Windows, мінімум — хвилина) або погодинно для деяких систем.
- Повний контроль: будь-яка ОС, будь-які програми — але й відповідальність за ОС і патчі твоя (пам'ятаєш модель спільної відповідальності).
- Інстанс живе в **одній AZ** — у тій підмережі, яку ти обереш.

Запуск сервера по кроках

Майстер запуску в консолі ставить сім запитань. Тепер ти розумієш кожне:



Сім кроків запуску EC2: від імені й образу до ролі й стартового скрипту

1. **Ім'я й теги** — `web-1`, `project = site`.
2. **AMI** — з якого «зліпка» створити сервер: ОС і вже встановлені програми.
3. **Тип інстансу** — скільки vCPU і пам'яті (про типи — нижче).

4. **Key pair** — ключ для входу по SSH. Можна обійтися без нього, якщо заходити через Session Manager.
5. **Мережа** — VPC, **підмережа** (а отже й AZ), чи потрібна публічна IP-адреса, які **Security Groups**.
6. **Сховище** — розмір і тип системного диска EBS.
7. **Додатково** — **IAM-роль** (instance profile, щоб серверу не знадобилися ключі) і **user data** — скрипт, який виконається при першому старті (наприклад, встановить вебсервер).

AMI — зліпок комп'ютера

AMI («ей-ем-ай», Amazon Machine Image) — **шаблон** сервера: операційна система, налаштування й встановлені програми. З одного AMI можна запустити скільки завгодно однакових серверів.

- Бувають AMI від AWS (Amazon Linux, Ubuntu, Windows Server), від сторонніх компаній (AWS Marketplace) і **власні**.
- **Golden AMI** («золотий образ») — власний AMI з уже встановленим і налаштованим застосунком. Новий сервер з нього готовий за хвилину, а не за пів години встановлення — це дуже прискорює масштабування.

- AMI **регіональний**. Щоб запустити такий самий сервер в іншому регіоні (наприклад, для аварійного відновлення), AMI **копіюють** у той регіон.

Типи інстансів: модельний ряд

Тип — це «модель авто» з певним набором процесора й пам'яті. Сімейства позначають літерами:

Сімейство	Образ	Для чого
T, M — загального призначення	універсал на щодень	сайти, невеликі бази, більшість задач
C — compute optimized	спорткар	кодування відео, розрахунки, ігрові сервери
R, X — memory optimized	вантажівка з великим кузовом	бази й кеші в пам'яті, аналітика
I, D — storage optimized	авто з величезним швидким багажником	бази з дуже високими IOPS на локальних дисках
P, G, Inf, Trn — прискорювачі	авто з турбіною (GPU)	машинне навчання, графіка

Як читати назву `m7g.2xlarge`:

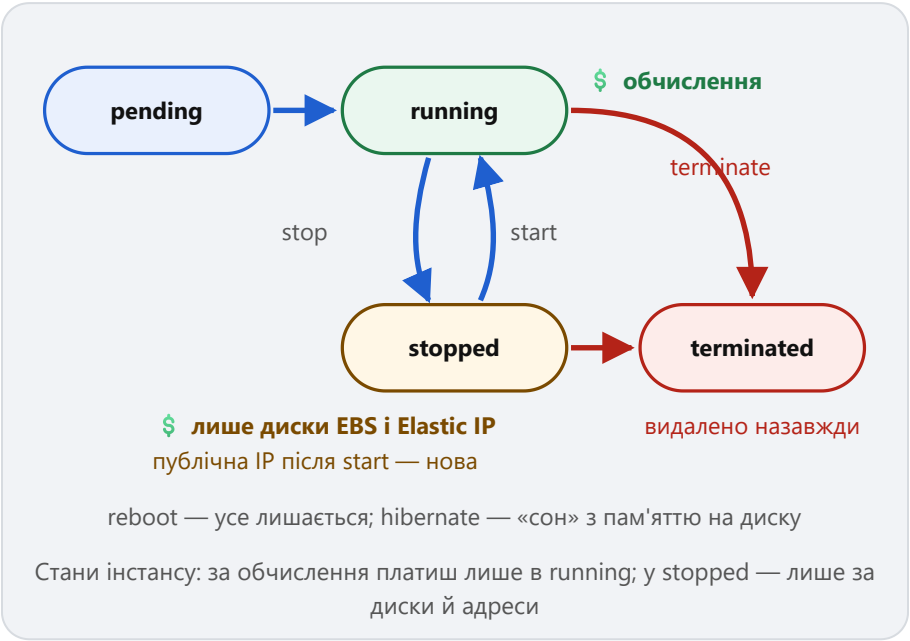


Розбираємо назву m7g.2xlarge: сімейство, покоління, особливості, розмір

- **m** — сімейство (загального призначення);
- **7** — покоління (що більше, то новіше й вигідніше);
- **g** — особливість: процесор **Graviton** (ARM, помітно краща ціна за ту саму роботу). Інші літери: **a** — AMD, **i** — Intel, **d** — локальні NVMe-диски, **n** — посилена мережа;
- **2xlarge** — розмір. Розміри йдуть так: `nano`, `micro`, `small`, `medium`, `large`, `xlarge`, `2xlarge`, `4xlarge` … — кожен наступний приблизно **вдвічі** більший.

Сімейство **T** особливе — **burstable** («берстабл», з ривками): сервер зазвичай працює на невелику частку процесора, а в тиху годину накопичує **CPU-кредити**, які витрачає на короткі ривки. Як скарбничка енергії. Ідеально для сайтів, що більшість часу відпочивають.

Життєвий цикл сервера і гроші



Стани інстансу: за обчислення платиш лише в running; у stopped — лише за диски й адреси

- **pending** — готується до старту.
- **running** — працює; **платиш за обчислення**.
- **stopped** — зупинений. За обчислення **не платиш**, але диски EBS і Elastic IP продовжують коштувати. Дані на EBS зберігаються.
- **terminated** — видалений назавжди. Системний диск за замовчуванням видаляється разом із ним.
- **reboot** (перезавантаження) — усе лишається як було, навіть IP.

- **Hibernate** («гайбернейт», сон ноутбука) — вміст оперативної пам'яті зберігається на системний диск (він має бути зашифрований), і після пробудження застосунок продовжує з «теплою» пам'яттю — швидше, ніж холодний старт.

⚠️ Звичайна публічна IP-адреса **змінюється** після stop/start. Потрібна постійна — Elastic IP.

User data, метадані й перевірки стану

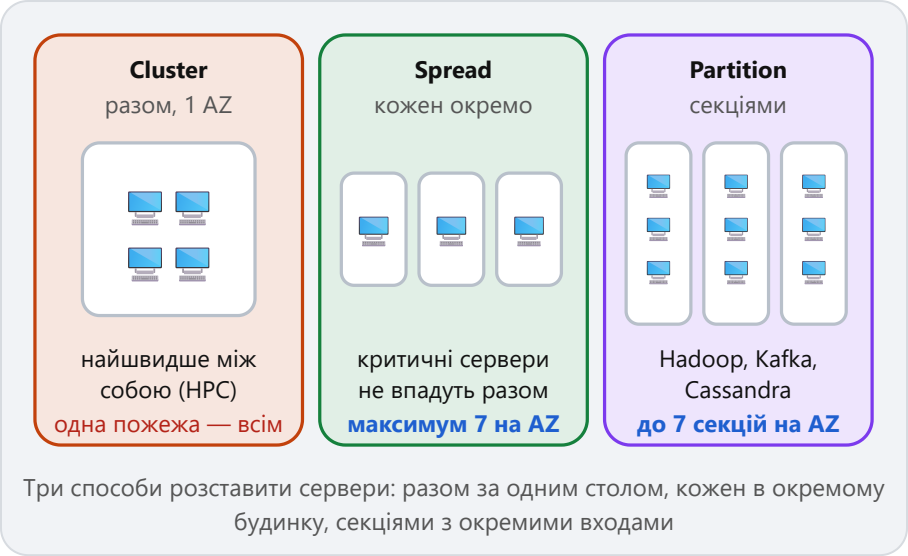
- **User data** — скрипт, що виконується при **першому** старті: встановити програми, завантажити налаштування.
- **Instance metadata** — внутрішня «довідка» сервера про себе: його ID, адреси, тимчасові ключі ролі. Безпечна версія — **IMDSv2** (з токенами сесії).
- **Status checks** — AWS постійно перевіряє сервери двома перевірками:
  - **system** — проблема з **фізичним хостом** AWS. Ліки: **stop/start** (сервер переїде на інший хост) або тривога CloudWatch з дією **recover**;
  - **instance** — проблема **всередині** ОС (завис, закінчилася пам'ять). Ліки: перезавантаження, виправлення.

Мережа сервера

- **ENI** («і-ен-ай», elastic network interface) — мережева карта сервера з приватною адресою. Її можна **перечепити** на інший сервер — і той отримає ту саму приватну адресу (так роблять швидке перемикання на запасний сервер).
- **Elastic IP** — постійна публічна адреса (платна).
- **ENA** — посилена мережа до 100+ Gbps. **EFA** («і-еф-ей», Elastic Fabric Adapter) — надшвидкий зв'язок між серверами для наукових розрахунків (HPC), в обхід операційної системи.

Placement groups: разом чи окремо

Іноді важливо, **як** фізично розставлені твої сервери. Для цього є **placement groups** («плейсмент груп»):



Три способи розставити сервери: разом за одним столом, кожен в окремому будинку, секціями з окремими входами

- **Cluster** — **команда за одним столом**: сервери поруч в **одній AZ**, найменша затримка й найвища швидкість обміну між ними. Для HPC (разом з EFA). Ризик: одна «пожежа» зачепить усіх.
- **Spread** — **кожен в окремому будинку**: кожен сервер на окремому обладнанні, щонайбільше **7 серверів на AZ** у групі. Для кількох **критичних** серверів, які не мають зламатися разом.
- **Partition** — **секції з окремими входами**: групи стійок (до 7 секцій на AZ), у кожній — багато серверів. Для великих розподілених систем на кшталт Hadoop, Kafka, Cassandra, які самі розкладають копії по секціях.

Більший сервер чи більше серверів

- **Вертикально** (більший тип) — сервер треба **зупинити**, змінити тип і запустити. Є межа.
- **Горизонтально** (більше серверів) — без зупинки, через групу автомасштабування. Про це — через розділ.

🧪 **Спробуй у справжньому AWS: Запусти свій перший сервер і подивися, як змінюється IP (15 хвилин)**

На безкоштовному плані запуск інстансу ще й додає кредити. Головне — наприкінці видалити сервер.

1. У консолі відкрий **EC2** → **Launch instance**.
2. Ім'я: `test-1`. AMI: **Amazon Linux**. Тип: обери такий, що позначений **Free tier eligible**.
3. Key pair: **Proceed without a key pair** — входити по SSH ми не будемо.
4. Network settings: лиши default VPC; зніми галочку, що дозволяє SSH з будь-якої адреси (для цієї вправи вхід не потрібен).
5. **Launch instance**, потім відкрий список інстансів.
6. Подивися на вкладку деталей: **Availability Zone**, **Private IPv4**, **Public IPv4**, тип інстансу, AMI.

7. **Instance state** → **Stop**. Коли стан стане `stopped`, — **Start**. Порівняй публічну адресу: вона змінилася, а приватна — ні.
8. Прибери за собою: **Instance state** → **Terminate**.

📖 **Словничок**

EC2 («і-сі-ту») — оренда віртуальних серверів; **інстанс** — один сервер.

AMI («ей-ем-ай») — шаблон сервера (ОС + програми); регіональний; **golden AMI** — з готовим застосунком.

Тип інстансу — сімейство + покоління + особливості + розмір (`m7g.2xlarge`).

Graviton — процесори AWS на ARM (літера `g`), краща ціна за роботу.

Burstable (T) — інстанси з CPU-кредитами для коротких ривків.

running / stopped / terminated — працює (платиш) / зупинений (платиш за диски) / видалений.

Hibernate — «сон» зі збереженням пам'яті на зашифрований системний диск.

User data — скрипт першого старту. **IMDSv2** — безпечний доступ до метаданих.

System / instance status check — проблема хоста AWS (stop/start, recover) / проблема ОС (reboot).

ENI / Elastic IP / EFA — мережева карта, яку можна перечепити / постійна публічна IP / надшвидкий зв'язок для HPC.

Placement group: cluster / spread / partition — разом / кожен окремо (7 на AZ) / секціями (до 7 на AZ).

🧩 **Міні-тест: перевір себе**

? Що таке AMI?

А Шаблон-«зліпок» сервера: ОС і програми, з якого запускають нові інстанси

Б Тип процесора

В Мережевий інтерфейс

Г Ціна сервера

Відповідь: А. З одного AMI можна запустити скільки завгодно однакових серверів. Golden AMI — з уже встановленим застосунком.

? Як прочитати тип m7g.2xlarge?

А Модель 7, два гігабайти пам'яті

Б Сімейство m, покоління 7, процесор Graviton, розмір 2xlarge

В Сімейство g, сім серверів, два диски

Г Мобільний сервер сьомої серії

Відповідь: Б. Сімейство — покоління — особливості — розмір. Літера g — Graviton.

? Потрібно кодувати відео — багато обчислень на процесорі. Яке сімейство підходить найкраще?

А R

Б C

В I

Г T

Відповідь: Б. C — compute optimized, «спорткар». R — багато пам'яті, I — швидкі локальні диски.

? Сервер перевели в стан stopped. За що AWS далі братиме гроші?

А. За обчислення, як і раніше

Б. За диски EBS і Elastic IP, але не за обчислення

В. Ні за що

Г. Лише за мережевий трафік минулого місяця

Відповідь: Б. У stopped процесор не працює, але диски й зарезервовані адреси лишаються — і коштують.

? HPC-кластер: сервери мають обмінюватися даними з мінімальною затримкою. Що обрати?

- **А** Cluster placement group разом з EFA
- **Б** Spread placement group
- **В** Сервери в різних регіонах
- **Г** Partition placement group з однією секцією на регіон

Відповідь: А. Cluster — «команда за одним столом» в одній AZ; EFA — надшвидкий зв'язок між вузлами.

? Три критичні сервери не повинні відмовити разом через спільне обладнання. Що обрати?

- **А** Cluster placement group
- **Б** Spread placement group
- **В** Один великий сервер
- **Г** Hibernate

Відповідь: Б. Spread розміщує кожен сервер на окремому обладнанні (до 7 на AZ).

? Як безпечніше отримувати метадані й тимчасові ключі ролі на сервері?

- **А** Записати ключі у файл
- **Б** Використовувати IMDSv2
- **В** Вимкнути метадані назавжди
- **Г** Відкрити метадані в інтернет

Відповідь: Б. IMDSv2 вимагає токен сесії й захищає від частини атак.

✅ **Головне**

EC2 — «автопрокат» віртуальних серверів з посекундною оплатою; інстанс живе в **одній AZ**.

Запуск: ім'я → **AMI** → **тип** → ключ → **мережа й SG** → **диск** → **роль** і **user data**. **Golden AMI** прискорює старт; AMI **копіюють** в інший регіон.

Типи: **T/M** — щоденні, **C** — процесор, **R/X** — пам'ять, **I/D** — швидкі локальні диски, **P/G** — GPU; `g` — **Graviton**.

Stopped — платиш лише за диски й Elastic IP; публічна IP міняється після stop/start; **hibernate** зберігає пам'ять. **System** check → stop/start або recover.

Placement: **cluster** (разом, HPC + EFA), **spread** (окремо, 7 на AZ), **partition** (секціями, Hadoop/Kafka/Cassandra).

💳 Як платити за сервери і не переплатити

👴 Простими словами

За сервери можна платити як за таксі (On-Demand), як за абонемент зі знижкою (Savings Plans, Reserved) або купувати «гарячі» залишки дешево, але з ризиком, що заберуть (Spot).

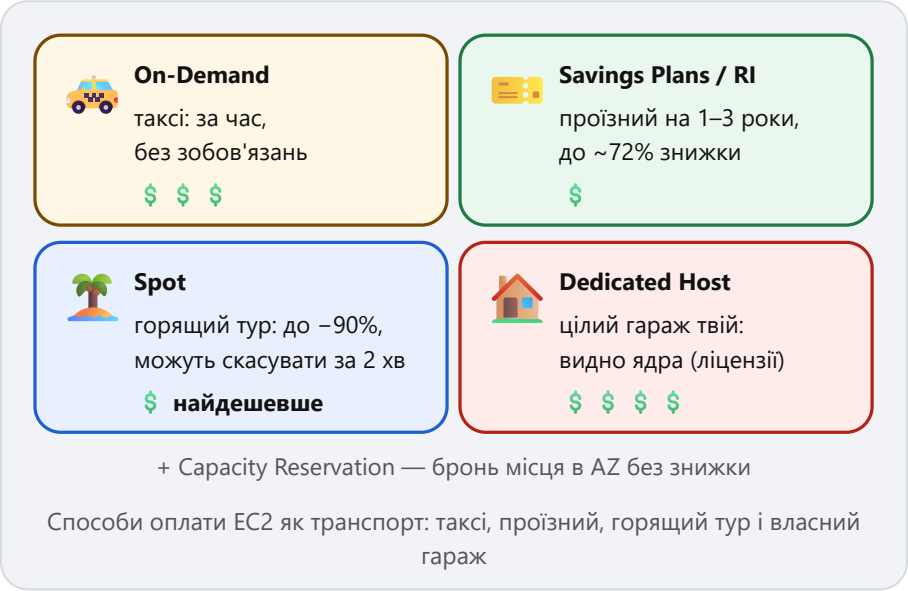
🎯 Що ти дізнаєшся

- п'ять способів платити за EC2 — від таксі до власного гаража
- чим Savings Plans відрізняються від Reserved Instances
- що таке Spot і коли він небезпечний
- навіщо Dedicated Host і Capacity Reservation
- як скласти вигідну суміш для реального сайту

🥐 Перший рахунок за сервери «Хмаринки» здивував Олю: сайт працює цілодобово за найдорожчим тарифом, нічні звіти займають потужний сервер, а в Чорну п'ятницю серверів треба втричі більше, ніж у звичайний вівторок. «Ми платимо за таксі щодня з дому на роботу, — зітхає Тарас. — Час купити проїзний».

П'ять способів платити

Образ з простого курсу — **транспорт**. Робота та сама, а платити можна дуже по-різному.



Способи оплати EC2 як транспорт: таксі, проїзний, горящий тур і власний гараж

On-Demand — таксі

On-Demand («он-діманд», на вимогу) — платиш за секунди чи години роботи, **без жодних зобов'язань**. Увімкнула програма сервер — лічильник пішов, вимкнула — зупинився.

- Найгнучкіше і **найдорожче** за годину.
- Для коротких і непередбачуваних задач, нових застосунків, тестів.

Savings Plans — проїзний на роки

Savings Plans («сейвінгз планз») — ти **зобов'язуєшся витрачати певну суму на годину** (наприклад, 5 доларів) протягом **1 або 3 років**, а AWS дає за це знижку. Два головні види:

- **Compute Savings Plan** — знижка до **~66%** на **будь-яке** сімейство, розмір, регіон і ОС, а також на **Fargate і Lambda**. Можна вільно міняти сервери — знижка «переїде» за тобою.
- **EC2 Instance Savings Plan** — знижка до **~72%**, але прив'язана до **конкретного сімейства в конкретному регіоні** (розмір і ОС міняти можна).

Що більше платиш наперед (без передоплати / частково / повністю) і що довший строк (1 чи 3 роки), то більша знижка.

🆕 3 грудня 2025 року є ще **Database Savings Plans** — до 35% знижки на бази даних (RDS, Aurora, DynamoDB та інші) за зобов'язання на рік.

Reserved Instances — лізинг конкретного авто

Reserved Instances (RI, «резервд інстансиз») — старший спосіб: бронюєш **конкретний тип** інстансу на 1 або 3 роки.

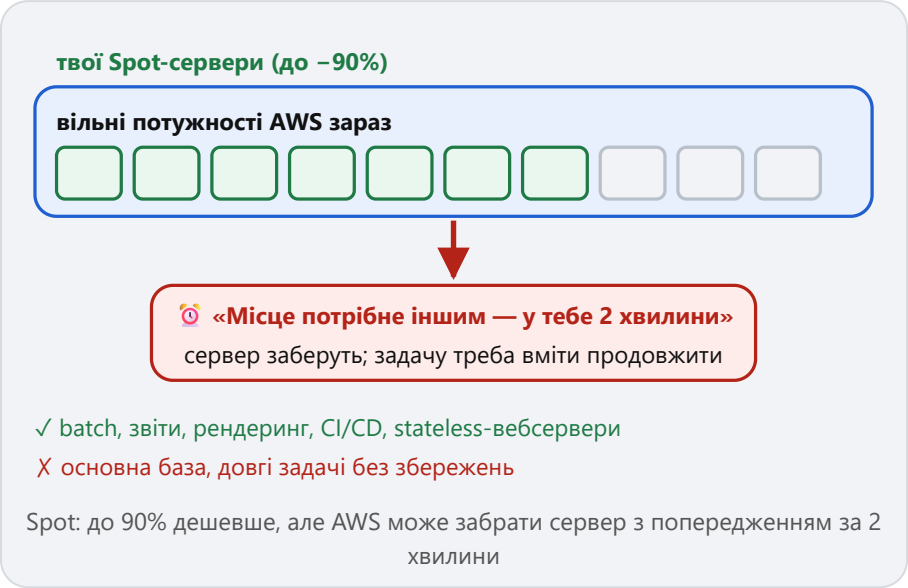
- **Standard RI** — знижка до ~72%, але змінити сімейство не можна (зате непотрібні RI можна продати на RI Marketplace).
- **Convertible RI** — можна міняти сімейство й ОС, знижка менша.

Savings Plans гнучкіші й сьогодні зазвичай вигідніші для EC2. Але RI досі існують, зокрема для баз даних і кешів. На іспиті обидва варіанти — «довгострокова знижка за постійне навантаження».

Spot — горящий тур

Spot Instances («спот») — AWS продає **вільні потужності**, які зараз простоюють, зі знижкою **до 90%**. Але є умова: коли ці потужності знадобляться іншим, AWS **забере** сервер, попередивши лише **за 2 хвилини**.

Образ — **горящий тур**: дуже дешево, але поїздку можуть скасувати.



Spot ідеальний для задач, які **можна перервати й продовжити**:

- пакетна обробка й нічні звіти;
- рендеринг відео, аналітика великих даних;
- збірка й тестування коду (CI/CD);
- вебсервери **без стану** за балансувальником (впав один — решта працює).

⚠️ Spot **не** для основної бази даних і **не** для довгих задач без проміжних збережень (checkpoints) — перервали на 95% роботи, і все спочатку.

Щоб не залежати від одного типу серверів, використовують **Spot Fleet** чи **EC2 Fleet** зі стратегією **price-capacity-optimized** — обирати з кількох типів ті, де вільних потужностей найбільше, а ціна найкраща.

Dedicated — цілий гараж тільки твій

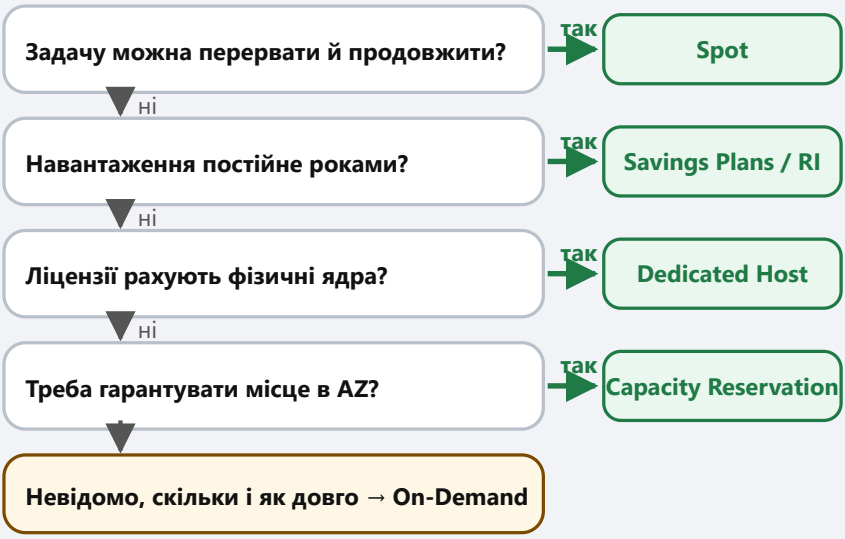
- **Dedicated Host** («дедікейтед хост») — **цілий фізичний сервер** лише для тебе, і ти **бачиш його сокети й ядра**. Потрібно, коли ліцензії на програми рахуються **за фізичними ядрами** (Windows Server, SQL Server, Oracle — BYOL, «принеси свою ліцензію») або коли цього вимагає регулятор. Найдорожчий варіант.
- **Dedicated Instance** — твої інстанси стоять на обладнанні, яке **не ділиш** з іншими клієнтами, але хостом ти не керуєш і ядер не бачиш.

Capacity Reservation — бронь місця без знижки

On-Demand Capacity Reservation — **гарантія**, що в конкретній AZ для тебе знайдеться потрібна кількість серверів певного типу, на будь-який строк. **Знижки не**

дає — лише впевненість (знижку можна додати поверх через Savings Plans). Для важливої події, коли ризикувати не можна.

Як вибрати: дерево рішень



Дерево вибору способу оплати: перервність, постійність, ліцензії, гарантія місця

Суміш для «Хмаринки»

Реальні системи рідко платять одним способом. Тарас склав таку суміш:

- **Два сервери сайту працюють 24/7 роками** → Compute Savings Plan на 3 роки. Постійна база — постійна знижка.

- **Додаткові сервери у вечірні піки й у Чорну п'ятницю** → On-Demand, який сам додає й прибирає автомасштабування.
- **Нічні звіти** → Spot: якщо сервер заберуть — звіт перезапуститься.
- **Старий бухгалтерський сервер з ліцензією на ядра** → Dedicated Host.
- **Тестові сервери** → вимикати на ночі й вихідні за розкладом.



кількість серверів «Хмаринки» за добу

Суміш способів оплати: постійна база — Savings Plans, піки — On-Demand, нічні задачі — Spot

Ще кілька способів заощадити

- **Правильний розмір** (right-sizing) — не брати «вантажівку» під «пакет з хлібом». Сервіс **Compute Optimizer** підкаже, які сервери завеликі.
- **Graviton** — та сама робота дешевше.
- **Вимикати непотрібне** за розкладом (EventBridge Scheduler — про нього в розділі про черги й події).

📖 **Словничок**

On-Demand («он-діманд») — «таксі»: без зобов'язань, найдорожче за годину.

Savings Plans — зобов'язання витрачати $/год 1–3 роки. **Compute SP** (до ~66%, будь-що + Fargate і Lambda), **EC2 Instance SP** (до ~72%, сімейство в регіоні).

Database Savings Plans — до 35% на бази даних (1 рік).

Reserved Instances (RI) — бронь конкретного типу на 1–3 роки: Standard (до ~72%) і Convertible.

Spot («спот») — вільні потужності до 90% дешевше; забирають з попередженням за **2 хвилини**.

Spot Fleet / EC2 Fleet — набір Spot і On-Demand з кількох типів; стратегія price-capacity-optimized.

Dedicated Host — цілий фізичний сервер, видно сокети й ядра (BYOL).

Dedicated Instance — обладнання не ділиш з іншими, але хостом не керуєш.

Capacity Reservation — гарантія місця в AZ без знижки.

Right-sizing / Compute Optimizer — правильний розмір / порадник щодо розміру.

🧩 Міні-тест: перевір себе

? Нічні звіти можна перезапустити, якщо щось перерветься. Потрібно найдешевше. Що обрати?

- **А** Spot Instances
- **Б** On-Demand
- **В** Dedicated Host
- **Г** Capacity Reservation

Відповідь: А. Переривні задачі — ідеальні для Spot: знижка до 90%.

? Сайт працює 24/7 роками. Потрібна знижка і свобода міняти сімейства серверів або перейти на Fargate. Що обрати?

- **А** Standard Reserved Instances
- **Б** EC2 Instance Savings Plan
- **В** Compute Savings Plan
- **Г** Spot

Відповідь: В. Compute SP діє на будь-яке сімейство, регіон, ОС, а також на Fargate і Lambda.

? Ліцензії Windows Server компанії прив'язані до фізичних ядер. Що обрати?

- **А** Dedicated Instances
- **Б** Dedicated Hosts
- **В** Spot
- **Г** On-Demand на спільному обладнанні

Відповідь: Б. Лише Dedicated Host показує фізичні сокети й ядра — це потрібно для таких ліцензій (BYOL).

? За скільки часу AWS попереджає, перш ніж забрати Spot-сервер?

- **А** За 2 секунди
- **Б** За 2 хвилини
- **В** За 2 години
- **Г** За 2 дні

Відповідь: Б. Лише 2 хвилини — тому Spot для задач, які не шкода перервати.

? На важливу подію потрібно гарантувати місткість серверів у конкретній AZ без довгострокового зобов'язання. Що обрати?

А Spot

Б Savings Plans на 3 роки

В On-Demand Capacity Reservation

Г Dedicated Instance

Відповідь: В. Capacity Reservation гарантує місце на будь-який строк; знижки не дає.

? Нова задача з непередбачуваним навантаженням на кілька тижнів. Який спосіб оплати найрозумніший?

- **А** On-Demand
- **Б** Reserved Instances на 3 роки
- **В** Dedicated Host
- **Г** Spot для бази даних

Відповідь: А. Коли невідомо, скільки і як довго, — «таксі» без зобов'язань.

✅ **Головне**

On-Demand — таксі: без зобов'язань, найдорожче. **Savings Plans** — проїзний на 1–3 роки: **Compute** (~66%, будь-що + Fargate/Lambda) чи **EC2 Instance** (~72%, сімейство в регіоні). **RI** — бронь конкретного типу.

Spot — горящий тур: до **90%** дешевше, але забирають за **2 хвилини**; лише для переривних задач і серверів без стану.

Dedicated Host — фізичний сервер з видимими ядрами (ліцензії BYOL); **Dedicated Instance** — просто не ділиш залізо.

Capacity Reservation — гарантія місця без знижки.

Суміш: постійна база → SP/RI, піки → On-Demand з автомасштабуванням, переривне → Spot; плюс right-sizing, Graviton і вимкнення за розкладом.

⚖️ Коли клієнтів стає багато: балансувальники й автомасштабування

👴 Простими словами

Коли клієнтів багато, ставиш кілька серверів, а балансувальник, як адміністратор у черзі, розводить людей по вільних касах. Автомасштабування само відкриває нові каси в пік і закриває вночі.

🎯 Що ти дізнаєшся

- навіщо балансувальник навантаження і як він перевіряє здоров'я серверів
- що означають «рівень 3, 4 і 7» — на прикладі листа
- чим ALB, NLB і Gateway Load Balancer відрізняються і коли який
- як група автомасштабування тримає потрібну кількість серверів і замінює хворі
- політики масштабування — від «термостата» до прогнозу
- чому сервери мають бути stateless

🥐 Чорна п'ятниця. О 10:00 на сайт «Хмаринки» зайшли одночасно три тисячі людей, і єдиний сервер «ліг». Поки Тарас його перезапускав, клієнти пішли до конкурентів. «Більше такого не буде, — обіцяє він. — Поставимо адміністратора на вході й менеджера зміни, який сам кликатиме підмогу».

Адміністратор на вході і менеджер зміни

Згадай супермаркет у годину пік:

- **Балансувальник навантаження** (load balancer) — **адміністратор біля входу**: розводить покупців по касах, щоб жодна не була перевантажена, і не посилає людей до зачиненої каси.
- **Автомасштабування** (Auto Scaling) — **менеджер зміни**: черги ростуть — відкриває нові каси, спад — закриває зайві, касир захворів — ставить заміну.

Разом вони дають горизонтальне масштабування, знайоме з розділу про застосунки: не більший сервер, а **більше серверів**.

Що вміє балансувальник

Elastic Load Balancing (ELB) — сімейство балансувальників AWS. Кожен з них:

- дає **одну точку входу** (ім'я в DNS) для всіх серверів;
- розподіляє запити між серверами в **кількох AZ**;
- постійно **перевіряє здоров'я** серверів (health checks): стукає, наприклад, на `/health` і не надсилає запити тим, хто не відповідає;
- може **завершувати HTTPS** на собі (сертифікат ACM), розвантажуючи сервери;
- масштабується сам — керувати ним не треба.

Сервери, між якими балансувальник розподіляє запити, об'єднують у **target group** («таргет груп», цільова група).

Рівні 3, 4 і 7: образ листа

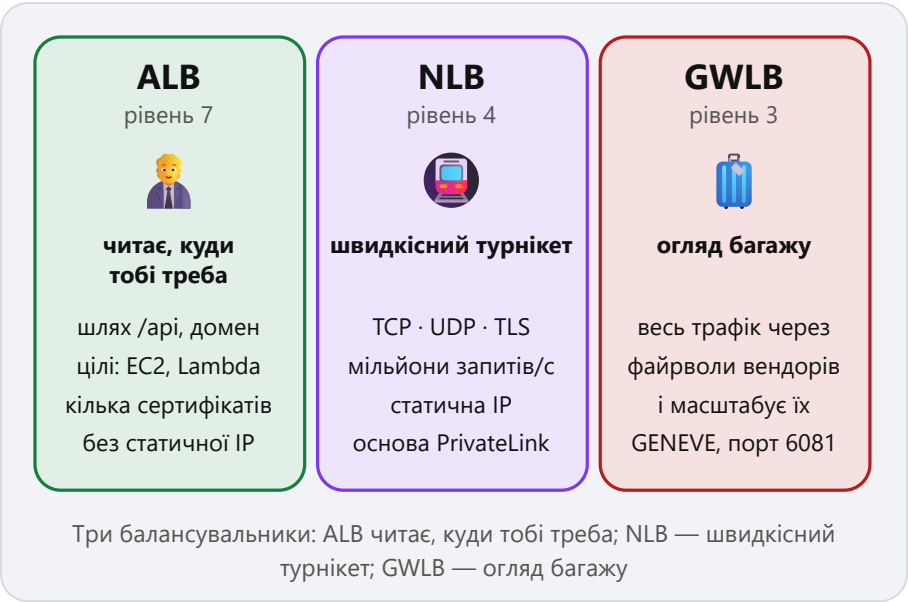
Балансувальники відрізняються тим, **наскільки глибоко** вони «читають» трафік. Інженери описують це рівнями (з моделі OSI). Уяви лист:



- **Рівень 3** (мережевий) — **адреса на конверті**: IP-адреси.
- **Рівень 4** (транспортний) — **номер дверей**: протокол TCP чи UDP і порт.
- **Рівень 7** (прикладний) — **зміст листа**: HTTP-запит, шлях `/menu`, домен, заголовки, cookies.

Що вищий рівень, то «розумніші» рішення можна ухвалювати, але й більше роботи на кожен запит.

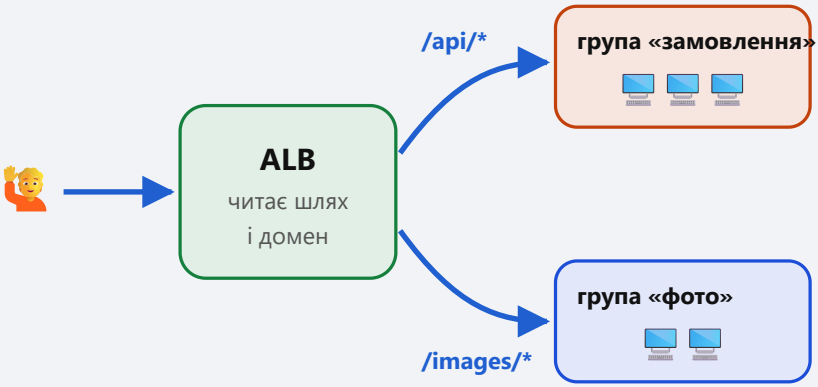
Три балансувальники



Три балансувальники: ALB читає, куди тобі треба; NLB — швидкісний турнікет; GWLB — огляд багажу

- **ALB** («ей-ел-бі», Application Load Balancer, **рівень 7**) — адміністратор, що **читає, куди тобі треба**:
  - маршрутизує за **шляхом** (`/api` → одні сервери, `/images` → інші), **доменом** (`api.khmarynka.ua`), заголовками, параметрами, методом;
  - цілі — сервери EC2, IP-адреси, **функції Lambda**, контейнери ECS (з динамічними портами);
  - тримає **кілька сертифікатів** для різних доменів (SNI), вміє **автентифікацію** через Cognito, перенаправляє HTTP на HTTPS;
  - **статичної IP-адреси не має**.

- **NLB** («ен-ел-бі», Network Load Balancer, **рівень 4**) — **швидкісний турнікет**:
  - TCP, UDP, TLS; **мільйони запитів за секунду**, наднизька затримка;
  - **статична IP-адреса в кожній AZ** (можна Elastic IP) — зручно, коли клієнт вносить адресу у свій «білий список»;
  - зберігає справжню IP-адресу клієнта; основа для **PrivateLink**.
- **GWLB** («гейтвей ел-бі», Gateway Load Balancer, **рівень 3**) — **пункт огляду багажу**: прозоро пропускає **весь** трафік через віртуальні апарати сторонніх вендорів — файрволи, системи виявлення вторгнень — і масштабує їх. (Технічно — протокол GENEVE, порт 6081.)
- **CLB** — старий Classic Load Balancer; для нових рішень не обирають.



одна адреса для клієнта — різні сервери за шляхом

ALB маршрутизує за шляхом: /api — на сервери замовлень, /images — на сервери фото

🎯 `/api` і `/images` на різні групи серверів → **ALB**. Статичний IP балансувальника → **NLB** (або Global Accelerator). Весь трафік через файрвол вендора → **GWLB**.

Корисні дрібниці балансувальників

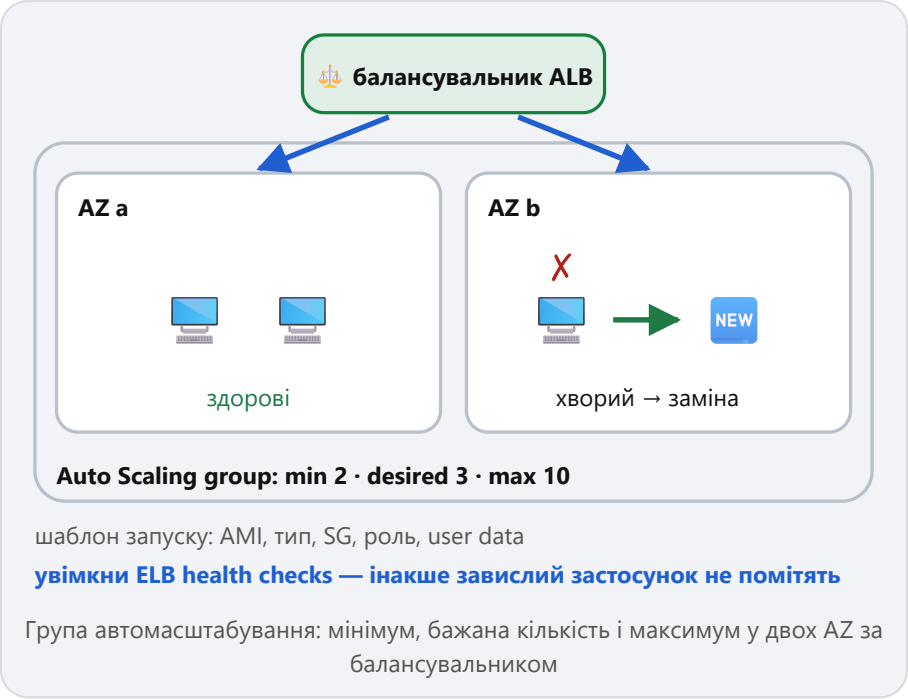
- **Cross-zone load balancing** — розподіляти запити рівномірно між **усіма** серверами в усіх AZ. В ALB увімкнено за замовчуванням і безкоштовно; в NLB і GWLB — вимкнено, а трафік між зонами платний.

- **Deregistration delay** (connection draining) — коли сервер виводять, балансувальник дає йому дообслуговувати поточні запити (за замовчуванням 300 секунд).
- **Sticky sessions** («прилиплі сесії») — завжди відправляти клієнта на той самий сервер. Це **милиця**: правильне рішення — **stateless-сервери**, а сесії — у спільному сховищі (ElastiCache чи DynamoDB). Пам'ятаєш дошку замовлень з розділу про застосунки?
- **Access logs** — журнал усіх запитів до балансувальника в S3.

Auto Scaling group: менеджер зміни

Auto Scaling group (ASG, «ей-ес-джі») — група однакових серверів, кількість яких змінюється автоматично. Налаштування:

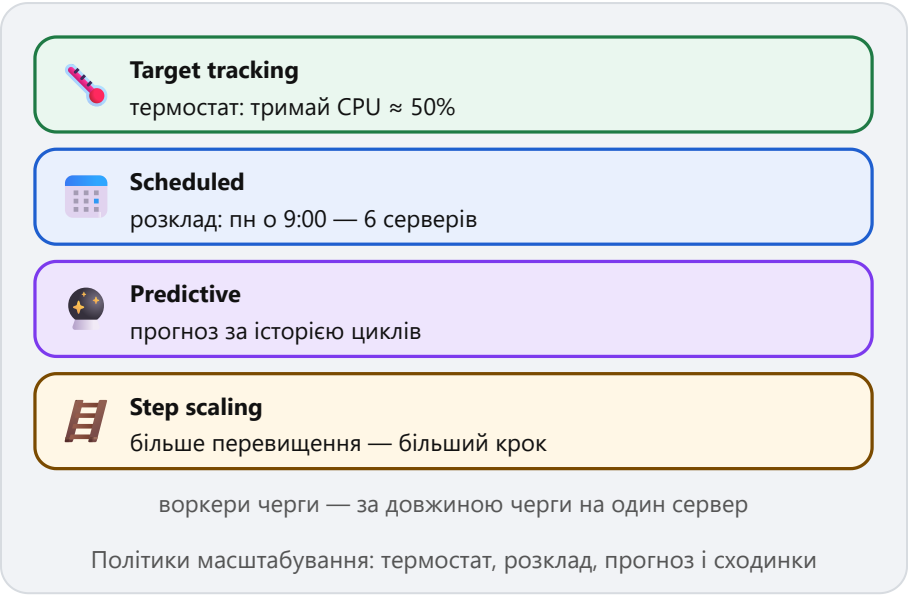
- **Launch template** («лонч темплейт», шаблон запуску) — **що** запускати: AMI, тип, Security Groups, роль, user data.
- **Min / desired / max** — не менше ніж, бажано стільки, не більше ніж. Наприклад, 2 / 2 / 10.
- **Кілька AZ** — ASG розкладає сервери по зонах і вирівнює їх.
- **Заміна хворих** — сервер не пройшов перевірку здоров'я → ASG його видаляє й запускає новий.



⚠️ За замовчуванням ASG перевіряє лише, чи **увімкнений** сервер (статус EC2). Якщо застосунок на сервері **завис**, але сам сервер «живий», ASG нічого не помітить. **Увімкни ELB health checks** в ASG — тоді група дивитиметься на ту саму перевірку, що й балансувальник.

Ще одне налаштування — **health check grace period**: скільки часу після запуску ASG не перевіряє здоров'я, щоб застосунок встиг стартувати. Якщо нові сервери одразу «хворі», бо довго запускаються, — збільш цей період.

Політики масштабування: як менеджер вирішує



Політики масштабування: термостат, розклад, прогноз і сходинки

- **Target tracking** («таргет трекінг») — **термостат**: «тримай середнє завантаження процесора біля 50%» чи «не більше 1 000 запитів на сервер». Найпростіша і рекомендована.
- **Step scaling** — **сходинки**: мале перевищення — додати 1 сервер, велике — додати 4.
- **Simple scaling** — старий спосіб: одна дія і пауза.
- **Scheduled** («шедьюлд») — **за розкладом**: «щопонеділка о 9:00 — 6 серверів», «щоночі — 2».
- **Predictive** («предиктів») — **прогноз**: машинне навчання вивчає історію циклічних піків і додає сервери **заздалегідь**.

Для серверів, що обробляють **чергу завдань**, масштабують за **довжиною черги на один сервер** (backlog per instance) — про черги в розділі про черги й події.

Між масштабуваннями діє пауза (**cooldown**, за замовчуванням 300 секунд) — щоб нові сервери встигли «прогрітися» й менеджер не запанікував.

Просунуті можливості

- **Lifecycle hooks** («лайфсайкл хукс») — зробити щось перед стартом сервера чи перед його видаленням: встановити програми, **забрати логи**.
- **Warm pools** («ворм пулз») — запас **заздалегідь підготовлених** серверів для застосунків, що довго стартують.
- **Instance refresh** — поступово замінити всі сервери групи на новий AMI **без простою**.
- **Termination policy** — кого прибирати першим при зменшенні (спрощено: з AZ, де серверів найбільше, → зі старим шаблоном → ближчого до нової години оплати).
- **Mixed instances policy** — база з On-Demand плюс Spot, кілька типів серверів.

💡 Старий застосунок, що вміє жити лише на **одному** сервері, теж можна зробити самовідновлюваним: **ASG з min = max = 1** плюс ELB health checks. Зламався сервер — група сама поставить новий.

🥐 Тепер сайт «Хмаринки» живе за ALB у двох AZ, у групі 2 / 2 / 10 з термостатом «50% процесора» і розкладом «у п'ятницю ввечері — щонайменше 4». Кошики клієнтів лежать не на серверах, а в спільному сховищі. У наступну Чорну п'ятницю група сама виросла до 9 серверів, а до ночі повернулася до двох.

📖 **Словничок**

Load balancer / ELB — «адміністратор на вході»; сімейство балансувальників AWS.

Target group — група цілей, між якими розподіляють запити.

Health check — перевірка здоров'я цілей (наприклад, `/health`).

Рівні 3 / 4 / 7 — IP-адреси / протокол і порт / зміст HTTP.

ALB — рівень 7: маршрути за шляхом, доменом, заголовками; Lambda як ціль; SNI; без статичної IP.

NLB — рівень 4: TCP/UDP, мільйони запитів, статична IP, основа PrivateLink.

GWLB — рівень 3: трафік через апарати безпеки вендорів (GENEVE, 6081).

Cross-zone load balancing / deregistration delay / sticky sessions — рівномірно між зонами / дообслуговування перед виведенням / «прилипання» до сервера.

Auto Scaling group, ASG — група серверів з min / desired / max у кількох AZ.

Launch template — шаблон запуску: AMI, тип, SG, роль, user data.

ELB health checks в ASG — замінювати сервери, чий застосунок не відповідає.

Grace period / cooldown — пауза на старт / пауза між масштабуваннями.

Target tracking / step / scheduled / predictive — термостат / сходинки / розклад / прогноз.

Lifecycle hooks / warm pools / instance refresh — дії при старті й видаленні / прогрітий запас / заміна на новий AMI.

🧩 **Міні-тест: перевір себе**

? Запити /api і /images мають іти на різні групи серверів. Що обрати?

- **А** ALB з маршрутизацією за шляхом
- **Б** NLB
- **В** GWLB
- **Г** Route 53 simple routing

Відповідь: А. ALB читає зміст запиту (рівень 7) і вміє розводити за шляхом.

? Клієнт хоче внести IP-адресу балансувальника у свій «білий список». Що обрати?

- **А** ALB
- **Б** NLB з Elastic IP (або Global Accelerator)
- **В** CLB
- **Г** Балансувальник не потрібен

Відповідь: Б. ALB статичної IP не має. NLB має статичну IP у кожній AZ.

? Весь трафік має проходити через файрвол-апарат стороннього вендора, який треба масштабувати. Що обрати?

- **А** ALB
- **Б** NLB
- **В** Gateway Load Balancer
- **Г** CloudFront

Відповідь: В. GWLB — «огляд багажу»: прозоро пропускає трафік через апарати безпеки.

? Застосунок на сервері завис, балансувальник бачить сервер хворим, але ASG його не замінює. Що зробити?

- **А** Увімкнути ELB health checks в Auto Scaling group
- **Б** Збільшити max у групі
- **В** Вимкнути перевірки здоров'я
- **Г** Перейти на Spot

Відповідь: А. За замовчуванням ASG бачить лише, чи сервер увімкнений. ELB health check дивиться на сам застосунок.

? Відомо: щопонеділка о 9:00 різкий пік. Яка політика масштабування?

- **А** Target tracking
- **Б** Scheduled scaling
- **В** Simple scaling
- **Г** Жодна

Відповідь: Б. Відомий час піку — розклад. Циклічні піки, які вивчає машинне навчання, — predictive.

? Потрібно тримати середнє завантаження процесора біля 50%. Яка політика?

- **А** Target tracking
- **Б** Scheduled
- **В** Step scaling без аларму
- **Г** Predictive

Відповідь: А. Target tracking — «термостат»: сам додає й прибирає сервери, щоб тримати метрику біля цілі.

? Користувач втрачає кошик, коли його запит потрапляє на інший сервер. Яке рішення найкраще для хмари?

А Sticky sessions назавжди

Б Зберігати сесії поза серверами, у спільному сховищі (ElastiCache або DynamoDB)

В Один великий сервер без масштабування

Г Вимкнути балансувальник

Відповідь: Б. Stateless-сервери + сесії в спільному сховищі. Sticky sessions — лише милиця.

✅ **Головне**

Балансувальник — адміністратор на вході: одна точка входу, кілька AZ, перевірки здоров'я, HTTPS на собі. **ASG** — менеджер зміни: min/desired/max, кілька AZ, заміна хворих.

Рівні: **3** — адреса на конверті, **4** — номер дверей, **7** — зміст листа.

ALB (7): шлях, домен, заголовки, Lambda, SNI, Cognito; **без статичної IP**. **NLB** (4): TCP/UDP, мільйони запитів, **статична IP**, PrivateLink. **GWLB** (3): апарати безпеки.

В ASG вмикай **ELB health checks**; довгий старт → **grace period** або **warm pools**; оновлення AMI → **instance refresh**; логи перед видаленням → **lifecycle hook**.

Політики: **target tracking** (термостат), **scheduled** (розклад), **predictive** (прогноз), **step** (сходинки). Сервери — **stateless**, сесії — у спільному сховищі; старий застосунок — ASG min = max = 1.

Serverless: Lambda і API Gateway — код без серверів

👴 Простими словами

Lambda — кухар на виклик: прийшов, приготував, пішов — платиш лише за приготовані страви. API Gateway — офіціант, який приймає замовлення від клієнтів і передає їх кухарю.

🎯 Що ти дізнаєшся

- що означає «код запускається на подію» і як працює функція Lambda
- головні ліміти Lambda — 15 хвилин, пам'ять, одночасність
- що таке холодний старт і як його прибрати
- reserved і provisioned concurrency, версії й псевдоніми
- що робить API Gateway — «офіціант» для твого API
- коли Lambda — не найкращий вибір

🥐 «Хмаринка» запускає мобільний застосунок: клієнти замовляють торти з телефона, а фото торта, яке кондитер завантажує в сховище, має автоматично зменшуватися до мініатюри. «Ще сервери? — стогне Оля. — Їх же треба латати, масштабувати, оплачувати вночі». — «Цього разу — без серверів, — каже Тарас. — Будемо платити лише за мілісекунди, коли код справді працює».

Подія → код

Serverless-код працює за принципом **дверного дзвінка**: поки ніхто не дзвонить, ніхто й не біжить відчиняти. Подзвонили — **подія** (event) — і хтось відчиняє.

Приклади подій для «Хмаринки»:

- клієнт натиснув «Замовити» в застосунку → запит до API;
- кондитер завантажив фото торта в сховище S3;
- настала 2:00 ночі (подія розкладу);
- у черзі з'явилося нове повідомлення;
- у базі змінився запис.

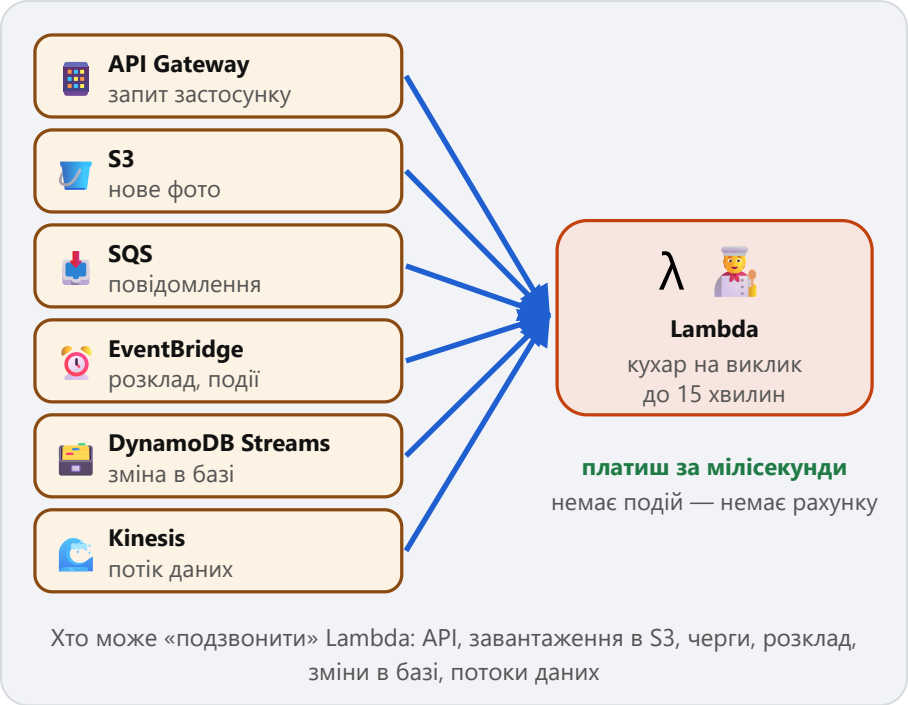
Lambda — кухар на виклик

AWS Lambda («лямбда») — сервіс, який запускає твій код **у відповідь на подію**. Образ з простого курсу — **кухар на виклик**: приходить, готує страву, іде — і ти платиш лише за приготоване.

Функція Lambda складається з:

- **коду** й **обробника** (handler) — з якого місця коду починати;
- **середовища виконання** (runtime): Python, Node.js, Java, .NET, Go, Ruby тощо;
- **пам'яті** — від 128 MB до 10 240 MB; від пам'яті **пропорційно залежить і процесор** (окремо CPU не налаштовується);
- **тайм-ауту** — максимум **15 хвилин**;
- **ролі виконання** (execution role) — що функції дозволено робити в AWS.

Платиш за **кількість викликів** і за **тривалість** у мілісекундах (з урахуванням обраної пам'яті). Немає викликів — немає рахунку.



Що може викликати Lambda: **API Gateway**, балансувальник **ALB**, події **S3**, черги **SQS**, теми **SNS**, **EventBridge** (зокрема розклад), зміни в **DynamoDB** (Streams), потоки **Kinesis**, **Cognito**, **CloudFront** (Lambda@Edge).

Ліміти, які треба знати напам'ять

Що	Ліміт
Тривалість одного виклику	**15 хвилин**
Пам'ять	128 MB – 10 240 MB (CPU росте разом з пам'яттю)
Тимчасовий диск `/tmp`	512 MB – 10 GB
Розмір запиту/відповіді	6 MB синхронно, 1 MB асинхронно

Що	Ліміт
Розмір коду	50 MB zip, 250 MB розпакований, образ контейнера — 10 GB
Одночасних виконань	1 000 на регіон за замовчуванням (можна підняти)

⚠️ Головне правило іспиту: задача, що триває **довше 15 хвилин**, — **не для Lambda**. Її розбивають на кроки через **Step Functions** або запускають у контейнерах (**ECS/Fargate**) чи в **AWS Batch**.

Холодний старт і як його прибрати

Коли функцію кличуть уперше (або після перерви), AWS мусить підготувати для неї середовище: завантажити код, запустити runtime. Це **холодний старт** (cold start) — **кухар щойно прийшов і розкладає ножі**. Зазвичай це частки секунди, але для деяких мов і великих програм — помітно.

- **Provisioned concurrency** («провіжнд конкаренсі») — **кухарі вже чекають на кухні**: AWS тримає задане число заздалегідь прогрітих середовищ. Холодних стартів немає, але за «чергування» платиш.
- **SnapStart** — швидкий старт зі знімка вже ініціалізованого середовища (для Java, Python, .NET).

Одночасність: скільки кухарів водночас

Concurrency — скільки копій функції працюють **одночасно**. За замовчуванням — до 1 000 на регіон для всіх функцій разом.

- **Reserved concurrency** («резервд») — **гарантує** функції її частку й водночас **обмежує** її зверху. Наприклад, «не більше 50 одночасно» — щоб функція не завалила базу тисячею з'єднань.
- **Provisioned concurrency** — **прогріті** середовища проти холодного старту (див. вище).

> ⚠️ Не плутай: **reserved** — ліміт і гарантія частки; **provisioned** — прогрів.

Lambda і бази даних

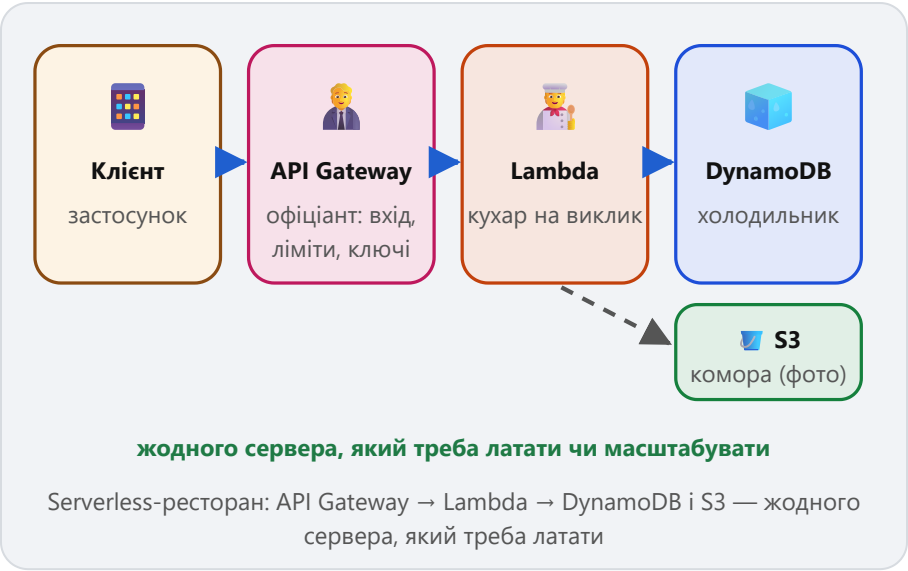
Якщо функції потрібна база RDS у приватній підмережі, функцію підключають **до VPC**. (Тоді в інтернет вона виходить лише через NAT.) Але є пастка: тисячі одночасних функцій відкривають тисячі з'єднань — і база «задихається». Ліки — **RDS Proxy**: «секретар», що тримає пул з'єднань і роздає їх функціям. Код майже не змінюється — міняється лише адреса бази.

Асинхронні виклики, версії й псевдоніми

- **Асинхронний** виклик (наприклад, від S3 чи SNS): Lambda сама повторює невдалі спроби, а безнадійні події можна відправити в **DLQ** (чергу «відмовників») або налаштувати **Destinations** — куди відправляти результат успіху чи помилки.
- **Версії** — незмінні «знімки» коду функції (v1, v2…).
- **Псевдоніми** (aliases) — «ярлики» на версії: `prod` → v5. Псевдонім може ділити трафік між двома версіями, наприклад **90% на v5 і 10% на v6** — поступовий випуск з можливістю миттєвого відкату.
- **Lambda + SQS**: функція сама забирає повідомлення з черги; тайм-аут «невидимості» черги (visibility timeout) ставлять щонайменше **у 6 разів більшим** за тайм-аут функції, а невдалі повідомлення — у DLQ.

API Gateway — офіціант

Мобільному застосунку потрібна «адреса», куди надсилати замовлення. Цим займається **Amazon API Gateway** («ей-пі-ай гейтвей») — **офіціант**: приймає замовлення, перевіряє, чи ти гість, стежить, щоб не було натовпу, і передає на кухню (Lambda чи інший сервіс).



Serverless-ресторан: API Gateway → Lambda → DynamoDB і S3 — жодного сервера, який треба латати

Три види API:

- **REST API** — найповніший: **кеш** відповідей, **API-ключі й usage plans** (окремі ліміти для різних клієнтів), WAF, перевірка запитів.
- **HTTP API** — простіший і **дешевший**, з авторизацією через токени JWT.
- **WebSocket API** — **двосторонній** зв'язок у реальному часі: чати, живі статуси замовлень.

Що ще варто знати:

- **Типи точок входу**: edge-optimized (через CloudFront), regional, **private** (лише з VPC через interface endpoint).

- **Throttling** (обмеження): за замовчуванням 10 000 запитів за секунду на акаунт у регіоні (сплеск — 5 000); понад ліміт клієнт отримує помилку **429**. **Usage plans** задають окремі ліміти й квоти для різних клієнтів.
- **Тайм-аут** інтеграції — за замовчуванням **29 секунд** (для regional і private REST API можна збільшити); розмір запиту — до 10 MB.
- **Авторизація**: IAM, **Cognito User Pools**, **Lambda authorizer** (власна функція, що перевіряє токен).
- **Stages** (dev, prod) і **canary** — частина трафіку йде на нову версію API.

🎯 «Serverless REST API з мінімальним адмініструванням» → **API Gateway + Lambda + DynamoDB**.

Коли Lambda — не найкращий вибір

- Задача довша за **15 хвилин**.
- **Постійне високе навантаження 24/7** — тоді сервери чи контейнери з довгостроковою знижкою зазвичай дешевші за мільйони мілісекунд Lambda.
- Потрібне особливе залізо (GPU) чи повний контроль над ОС.

🧪 Спробуй у справжньому AWS: Створи свою першу функцію Lambda (10 хвилин, безкоштовно)

На безкоштовному плані створення функції ще й додає кредити, а мільйон викликів на місяць для Lambda — завжди безкоштовний.

1. У консолі відкрий **Lambda** → **Create function** → **Author from scratch**.
2. Ім'я: `hello-khmarynka`, runtime: **Python** (найновіший), решту залиш як є → **Create function**.
3. У редакторі коду заміни повідомлення в `return` на своє, наприклад `"Свіжі круасани!"`, і натисни **Deploy**.
4. Натисни **Test**, створи тестову подію з будь-якою назвою й запусти.
5. Подивися на результат: відповідь функції, **Duration** (скільки мілісекунд працювала) і **Max memory used**.
6. Запусти тест ще раз — друга тривалість зазвичай менша: перший раз був холодний старт.
7. Прибери за собою: **Actions** → **Delete function**.

📖 **Словничок**

Serverless — код на подію без керування серверами; оплата за використання.

Lambda («лямбда») — функції на подію; до 15 хв; пам'ять 128 MB–10 GB (CPU разом з нею).

Handler / runtime — точка входу в код / мова й середовище виконання.

Execution role — права функції в AWS.

Cold start — затримка першого запуску середовища.

Concurrency — скільки копій функції працює одночасно (1 000 на регіон за замовчуванням).

Reserved concurrency — гарантія частки й стеля для функції.

Provisioned concurrency / SnapStart — прогріті середовища / швидкий старт зі знімка.

RDS Proxy — пул з'єднань між функціями й базою.

DLQ / Destinations — куди йдуть невдалі події / результати.

Версія / псевдонім (alias) — знімок коду / ярлик, що може ділити трафік 90/10.

API Gateway — «офіціант»: REST, HTTP і WebSocket API.

Usage plan — ліміти й квоти для окремого клієнта API; помилка **429** — «забагато запитів».

Lambda authorizer — власна функція перевірки токенів для API.

🧩 Міні-тест: перевір себе

? Який максимальний час виконання одного виклику Lambda?

- **А** 1 хвилина
- **Б** 15 хвилин
- **В** 1 година
- **Г** Без обмежень

Відповідь: Б. 15 хвилин — одна з найважливіших цифр іспиту.

? Задача триває 40 хвилин. Що обрати?

А Lambda з максимальною пам'яттю

Б Не Lambda: AWS Batch, ECS/Fargate або розбити на кроки через Step Functions

В Lambda з provisioned concurrency

Г Дві функції Lambda по 20 хвилин без оркестрації

Відповідь: Б. Lambda — до 15 хвилин. Довше — контейнери, Batch або покроковий процес.

? API на Lambda відповідає повільно через холодні старти, а це критично. Що допоможе?

А Provisioned concurrency (для Java — ще й SnapStart)

Б Reserved concurrency

В Зменшити пам'ять функції

Г Збільшити тайм-аут API Gateway

Відповідь: А. Provisioned concurrency тримає середовища прогрітими. Reserved лише обмежує й гарантує частку.

Тисячі функцій Lambda вичерпують з'єднання до бази RDS. Що допоможе з мінімальними змінами коду?

А Перенести базу в публічну підмережу

Б RDS Proxy

В Збільшити тайм-аут функції

Г Вимкнути VPC

Відповідь: Б. RDS Proxy тримає пул з'єднань і роздає їх функціям.

? Функцію треба обмежити, щоб вона ніколи не запускала більше 50 копій одночасно. Що використати?

- **А** Reserved concurrency
- **Б** Provisioned concurrency
- **В** SnapStart
- **Г** Alias

Відповідь: А. Reserved concurrency — стеля (і водночас гарантія) для конкретної функції.

? Різним клієнтам API потрібні різні ліміти запитів і API-ключі. Що обрати?

- **А** CloudFront
- **Б** API Gateway usage plans
- **В** Lambda layers
- **Г** Route 53 weighted routing

Відповідь: Б. Usage plans задають ліміти й квоти для окремих клієнтів за API-ключами.

? Як збільшити потужність процесора для функції Lambda?

А Окремим налаштуванням CPU

Б Збільшити пам'ять функції — процесор росте разом з нею

В Перейти в інший регіон

Г Увімкнути provisioned concurrency

Відповідь: Б. У Lambda процесор окремо не налаштовується — лише разом з пам'яттю.

✅ **Головне**

Lambda — код на **подію**, оплата за виклики й мілісекунди. Тригери: API Gateway, ALB, S3, SQS, SNS, EventBridge, DynamoDB Streams, Kinesis.

Ліміти: **15 хвилин**, **128 MB–10 GB** пам'яті (CPU разом з нею), **1 000** одночасних на регіон, payload 6 MB / 1 MB. Довше 15 хв → **Step Functions, ECS/Fargate, Batch**.

Cold start → **provisioned concurrency** або **SnapStart**; **reserved concurrency** — стеля й гарантія; багато з'єднань до RDS → **RDS Proxy**.

Псевдоніми ділять трафік між версіями (90/10); невдалі асинхронні події → **DLQ / Destinations**; SQS: visibility timeout ≥ **6 ×** тайм-аут функції.

API Gateway — «офіціант»: REST (кеш, **usage plans**, WAF), HTTP (дешевше), **WebSocket** (двосторонній); throttling → **429**; тайм-аут **29 с**; авторизація IAM, Cognito, Lambda authorizer.

📦 Контейнери: ECS, EKS, Fargate, Beanstalk і Batch

👶 Простими словами

Контейнер — коробка, в якій програма лежить разом з усім потрібним, тож працює однаково будь-де. ECS і EKS — диспетчери, що розставляють коробки, а Fargate — коли й самі полиці під коробки тобі не треба доглядати.

🎯 Що ти дізнаєшся

- де зберігаються образи контейнерів
- що таке оркестрація і навіщо «диспетчер порту»
- як влаштований ECS — task definition, task, service — і чим EC2 launch type відрізняється від Fargate
- коли обирати EKS
- що робить Elastic Beanstalk і як він безпечно випускає нові версії
- для чого AWS Batch

🥐 Команда розробників «Хмаринки» запакувала систему замовлень у контейнери. Тепер вона однаково працює на ноутбуці й на сервері — жодного «а в мене все працювало». «Але де ж їх запускати? — питає Оля. — І хто стежитиме, щоб вони не падали?» — «Для цього є диспетчери порту», — відповідає Тарас.

Нагадування: образ і контейнер

- **Образ** (image) — «креслення» контейнера: програма з усім потрібним, запакована в один файл.
- **Контейнер** — запущений образ.

Образи треба десь зберігати. В AWS для цього є **Amazon ECR** («і-сі-ар», Elastic Container Registry) — **склад образів**: приватний, з перевіркою образів на вразливості й копіюванням в інші регіони.

Оркестрація: диспетчер порту

Коли контейнерів десятки й сотні, потрібен **диспетчер порту** — **оркестратор**, який:

- вирішує, **на якому сервері** запустити кожен контейнер;

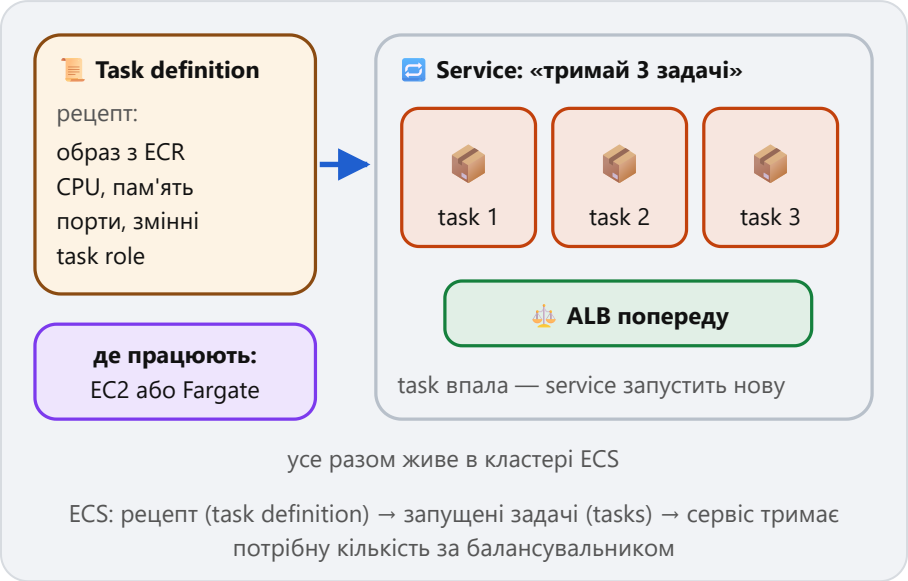
- **перезапускає** контейнери, що впали;
- **масштабує** — додає копії, коли навантаження росте;
- під'єднує контейнери до **балансувальника**.

В AWS два головні оркестратори: **ECS** і **EKS**.

Amazon ECS — диспетчер від AWS

Amazon ECS («i-ci-ec», Elastic Container Service) — власний оркестратор AWS, простіший у налаштуванні. Чотири слова, які треба знати:

- **Task definition** — **рецепт**: який образ, скільки CPU і пам'яті, які порти, змінні, яка роль.
- **Task** — одна **запущена копія** за рецептом (один чи кілька контейнерів разом).
- **Service** — «тримай завжди **N** задач запущеними» + автоматичне під'єднання до балансувальника й масштабування.
- **Cluster** — логічна група, де все це живе.



Права для контейнерів — через **task IAM role**: кожен контейнер отримує лише свої права (наприклад, писати в S3), без ключів. А ALB вміє спрямовувати запити до контейнерів навіть на динамічних портах.

Де працюють контейнери: EC2 чи Fargate

ECS (і EKS) може запускати контейнери двома способами:

- **EC2 launch type** — на **твоїх** серверах EC2 у кластері. Ти керуєш серверами: розміром, патчами, масштабуванням. Буває дешевше при стабільному навантаженні, потрібно для GPU й особливих вимог.

- **Fargate** («фаргейт») — **serverless** для контейнерів: серверів ти не бачиш зовсім. Вказуєш, скільки CPU і пам'яті потрібно задачі, і платиш саме за це. Образ з простого курсу — **порт сам дає крани й причали**.

🎯 «Запускати контейнери без керування серверами» → **Fargate** (з ECS або EKS).

Amazon EKS — керований Kubernetes

Kubernetes («кубернетес», K8s) — найпопулярніший у світі оркестратор з відкритим кодом, «стандарт індустрії». **Amazon EKS** («і-кей-ес») — керований Kubernetes в AWS: AWS доглядає «мозок» кластера, а ти запускаєш свої контейнери.

EKS обирають, коли:

- компанія **вже використовує Kubernetes** і має для нього інструменти й досвід;
- важлива **переносимість** між хмарами та власним дата-центром.

Для запуску на власних серверах є **ECS Anywhere** і **EKS Anywhere**.

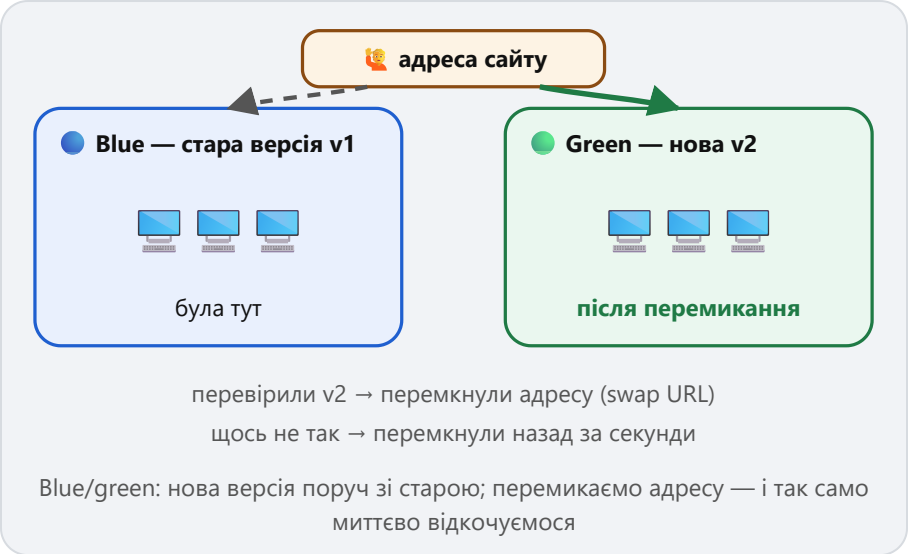
Elastic Beanstalk — ресторан під ключ

AWS Elastic Beanstalk («елестік бінсток») — **ресторан під ключ**: розробник приносить код (Java, .NET, Node.js, Python, PHP, Go чи Docker), а Beanstalk сам створює сервери EC2, групу автомасштабування, балансувальник і моніторинг. Це PaaS — «платформа як послуга».

- Ресурси лишаються видимими й доступними — Beanstalk просто керує ними за тебе.
- ⚠️ Beanstalk **не serverless**: він створює звичайні сервери EC2.

Beanstalk уміє по-різному випускати нові версії:

- **all at once** — усі сервери одразу (швидко, але з простоєм);
- **rolling** — частинами; **rolling with additional batch** — частинами з додатковою партією, щоб не втрачати потужність;
- **immutable** — нова версія на **нових** серверах, старі видаляють після перевірки;
- **traffic splitting** — частина трафіку на нову версію;
- **blue/green** — повна копія середовища з новою версією, потім **підміна адреси** (swap URL). Відкат — миттєвий.



AWS Batch — цех пакетних задач

AWS Batch — цех для великих замовлень без поспіху: тисячі пакетних задач будь-якої тривалості (обробка даних, розрахунки, рендеринг) стоять у **чергах**, а Batch сам запускає потрібні сервери — EC2, **Spot** чи Fargate — і прибирає їх після роботи.

🎯 «Тисячі пакетних задач з чергами, дешево, на Spot, довше за 15 хвилин» → **AWS Batch**.

Що обрати



Потреба	Вибір
Короткий код на подію, до 15 хв, без серверів	Lambda
Контейнери без серверів, будь-яка тривалість	Fargate (ECS або EKS)
Контейнери, простий «рідний» оркестратор AWS	ECS
Уже є Kubernetes, переносимість	EKS
Просто задеплоїти вебзастосунок, не думаючи про інфраструктуру	Elastic Beanstalk
Тисячі пакетних задач у чергах	AWS Batch
Повний контроль над ОС, особливе залізо, ліцензії	EC2

📖 Словничок

ECR («і-сі-ар») — приватний склад образів контейнерів.

Оркестратор — «диспетчер порту»: розміщує, перезапускає й масштабує контейнери.

ECS («і-сі-ес») — оркестратор AWS: task definition, task, service, cluster.

Task IAM role — права конкретного контейнера.

EC2 launch type / Fargate — контейнери на твоїх серверах / без серверів.

Kubernetes, EKS («кубернетес», «і-кей-ес») — стандартний оркестратор / його керована версія в AWS.

ECS/EKS Anywhere — ті самі інструменти на власних серверах.

Elastic Beanstalk — PaaS «під ключ»; створює EC2, ASG, ELB; не serverless.

Rolling / immutable / blue/green — випуск частинами / на нових серверах / поруч зі старим з миттєвим перемиканням.

AWS Batch — черги й запуск пакетних задач будь-якої тривалості, зокрема на Spot.

🧩 Міні-тест: перевір себе

? Потрібно запускати контейнери, не керуючи жодними серверами. Що обрати?

- **А** EC2 launch type
- **Б** Fargate (з ECS або EKS)
- **В** Elastic Beanstalk
- **Г** Dedicated Hosts

Відповідь: Б. Fargate — serverless для контейнерів: платиш за CPU й пам'ять задачі.

? Компанія вже використовує Kubernetes і хоче зберегти свої інструменти. Що обрати?

- **А** ECS
- **Б** Amazon EKS
- **В** Lambda
- **Г** AWS Batch

Відповідь: Б. EKS — керований Kubernetes.

? Розробники хочуть просто задеплоїти вебзастосунок, не розбираючись з серверами, балансувальниками й масштабуванням. Що обрати?

- **А** AWS Elastic Beanstalk
- **Б** AWS CloudFormation з нуля
- **В** Власні сервери EC2 вручну
- **Г** Amazon ECR

Відповідь: А. Beanstalk — «ресторан під ключ»: приносиш код, він сам створює інфраструктуру.

? Тисячі пакетних задач із чергами, будь-якої тривалості, якомога дешевше на Spot. Що обрати?

- **А** Lambda
- **Б** AWS Batch
- **В** API Gateway
- **Г** Elastic Beanstalk

Відповідь: Б. Batch сам керує чергами й серверами, зокрема Spot.

? Контейнеру ECS потрібен доступ до S3. Як дати права правильно?

- **А** Вписати ключі в образ
- **Б** Через ECS task IAM role
- **В** Дати контейнеру root-ключі
- **Г** Зробити сховище публічним

Відповідь: Б. Task role — права саме цього контейнера, без ключів.

? Де зберігати приватні образи контейнерів з перевіркою на вразливості?

- **А** Amazon ECR
- **Б** Amazon S3 Glacier
- **В** AWS Batch
- **Г** Route 53

Відповідь: А. ECR — приватний реєстр образів.

? Нову версію треба випустити з мінімальним ризиком і миттєвим відкатом. Що обрати в Beanstalk?

- **А** All at once
- **Б** Blue/green (підміна адреси)
- **В** Вимкнути старе середовище заздалегідь
- **Г** Rolling без перевірки

Відповідь: Б. Blue/green: нова версія поруч, перемикання адреси — і так само назад.

✅ **Головне**

ECR — склад образів; оркестратор — «диспетчер порту».

ECS: task definition (рецепт) → task (копія) → service (тримає N і під'єднує до ALB); права — **task role**.

EC2 launch type — свої сервери; **Fargate** — контейнери **без серверів**.

EKS — керований **Kubernetes** (уже є K8s, переносимість); **Anywhere** — на власних серверах.

Beanstalk — PaaS під ключ (не serverless), **blue/green** для безпечних випусків; **Batch** — черги пакетних задач будь-якої тривалості, зокрема на **Spot**.

Черги й події: як частини системи розмовляють без поспіху

👴 Простими словами

Черга (SQS) — талончики в поліклініці: прийшов, узяв номерок, лікар візьме, коли звільниться. SNS — гучномовець: одне оголошення чують усі, хто підписався.

🎯 Що ти дізнаєшся

- чому пряме з'єднання частин системи — ризик, а черга — порятунок
- як працює SQS — таймаут невидимості, DLQ, Standard і FIFO
- як SNS розсилає одне повідомлення багатьом і що таке fan-out
- що робить EventBridge і його планувальник
- як Step Functions ведуть багатокроковий процес з повторами й ручним схваленням
- коли потрібен Amazon MQ

🥐 Чорна п'ятниця знову: сайт витримав (завдяки автомасштабуванню), але кухонна система не встигала приймати замовлення — і частина просто загубилася. А ще кожне замовлення має потрапити на кухню, до бухгалтерії й у сервіс SMS. «Досі всі ці частини дзвонили одна одній напряму, — пояснює Тарас. — Якщо хтось не бере слухавку — дзвінок пропадає. Настав час поліклініки з талончиками».

Дзвінки напряму — і чим вони погані

Уяви, що кожна частина системи **дзвонить** наступній і чекає відповіді: сайт → кухня → склад → SMS. Це **тісне зв'язування** (tight coupling):

- кухня не встигає — сайт теж «висить»;
- SMS-сервіс упав — замовлення не оформлюється зовсім;
- у пік дзвінки просто губляться.

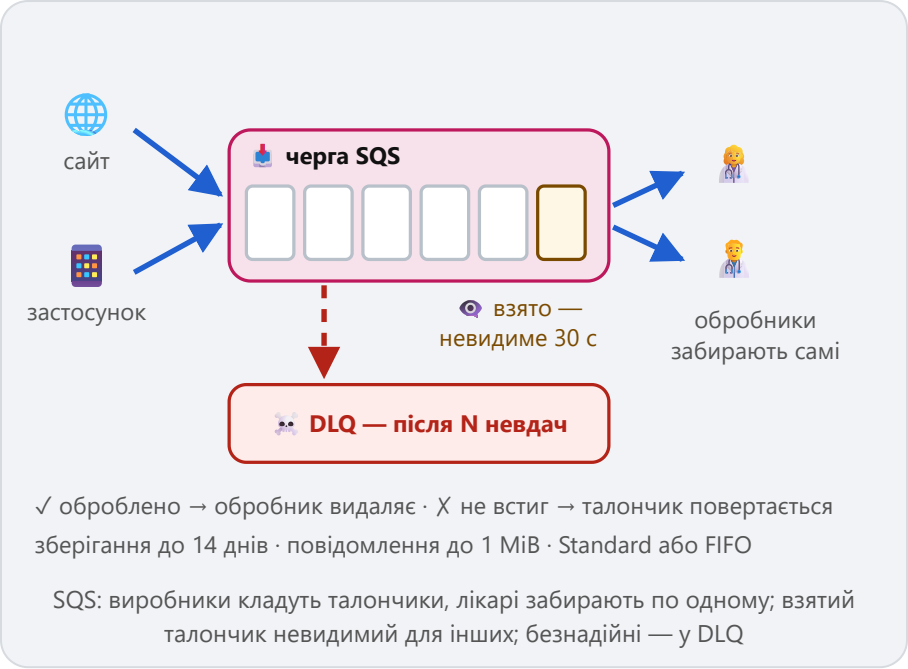
Слабке зв'язування (loose coupling) — частини спілкуються **через посередника**: кладуть повідомлення, а інші забирають його, коли готові. Кожна частина працює у своєму темпі, падіння однієї не валить інших, а

в пік повідомлення чекають. Це і є **decoupling** — «розв'язування», одне з улюблених слів іспиту.

SQS — талончики в поліклініці

Amazon SQS («ес-к'ю-ес», Simple Queue Service — проста служба черг) — керована **черга** повідомлень. Образ з простого курсу — **талончики в поліклініці**: пацієнти (повідомлення) чекають, а лікарі (обробники) беруть по одному у своєму темпі.

- **Виробник** (producer) кладе повідомлення в чергу; **споживач** (consumer) **сам забирає** його (модель **pull** — «забирай сам»).
- Повідомлення зберігаються від 1 хвилини до **14 днів** (за замовчуванням — 4 дні). Обробник може лежати кілька днів — замовлення дочекаються.
- Розмір повідомлення — до **1 MiB** (з серпня 2025; раніше 256 KB). Більше — через **Extended Client Library**: саме повідомлення лежить у S3, а в черзі — посилання (до 2 GB).
- Черга масштабується практично безмежно — керувати нею не треба.



Visibility timeout — талончик у руках лікаря

Коли обробник забрав повідомлення, воно **не зникає** з черги, а стає **невидимим** для інших на певний час — **visibility timeout** («візібіліті тайм-аут», за замовчуванням 30 секунд, максимум 12 годин). Це **талончик у руках лікаря**.

- Обробник упорався → він сам **видаляє** повідомлення (DeleteMessage). Черга сама нічого не видаляє!
- Не встиг за відведений час → талончик **повертається** в чергу, і його візьме хтось інший.

⚠️ Якщо обробка довша за visibility timeout, повідомлення **оброблять двічі**. Ліки — **збільшити visibility timeout**.

DLQ, long polling і затримка

- **Dead-letter queue** (DLQ, «ді-ел-к'ю», черга «відмовників») — окрема черга для **безнадійних** повідомлень, які не вдалося обробити після N спроб (maxReceiveCount). Їх аналізують окремо, а основна черга не забивається «отруйними» повідомленнями.
- **Long polling** — обробник чекає нових повідомлень до **20 секунд**, а не смикає чергу щосекунди з порожніми відповідями. Дешевше й ефективніше.
- **Delay queue** — відкласти доставку нових повідомлень до 15 хвилин.

Standard чи FIFO

- **Standard** («стандарт») — майже безмежна швидкість, але повідомлення може прийти **двічі** (at-least-once) і **не за порядком**. Обробники мають бути готові до дублів.

- **FIFO** («фіфо», first in — first out) — **строгий порядок** (у межах групи повідомлень MessageGroupId) і **обробка рівно один раз** (дублікати відсіюються протягом 5 хвилин). Швидкість обмежена: 300 викликів за секунду (3 000 повідомлень з пакетами по 10), а в режимі high-throughput — у рази більше. Назва черги закінчується на `.fifo`.

> 🎯 Фінансові транзакції строго по порядку і без дублів → **SQS FIFO**. Замовлення губляться в піки → **SQS** між компонентами.

Ще дві корисні речі:

- щоб SNS чи S3 могли класти повідомлення в чергу, їй потрібна **queue policy** (політика ресурсу), що це дозволяє;
- обробників черги зручно масштабувати групою автомасштабування за **довжиною черги на один сервер** (backlog per instance).

SNS — гучномовець

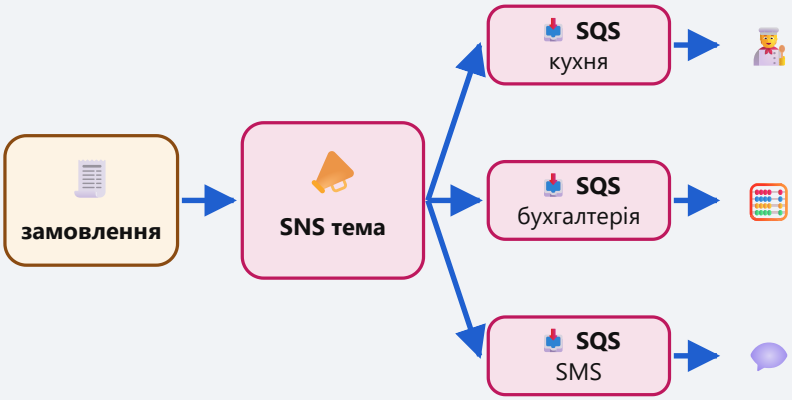
Amazon SNS («ес-ен-ес», Simple Notification Service) — **гучномовець**: одне оголошення чують усі одразу. Модель **push** — «кажу всім».

- Є **тема** (topic), на неї **підписуються**: черги SQS, функції Lambda, HTTP-адреси, email, SMS, мобільні push-сповіщення, Firehose.
- **Фільтри** (filter policy) — кожен підписник отримує лише свої типи повідомлень («бухгалтерії — лише оплачені замовлення»).
- Є **FIFO-теми** — разом з FIFO-чергами для порядку й без дублів.
- Розмір повідомлення — до 256 KB (з вересня 2026 за налаштуванням — до 1 MiB).

⚠️ SNS **не черга**: повідомлення він не зберігає «на потім». Якщо підписник недоступний, повідомлення може пропасти.

Fan-out: гучномовець + поштові скриньки

Звідси класичний шаблон **fan-out** («фен-аут», «віяло»): **SNS → кілька черг SQS**. Кожна система отримує **власну копію** замовлення у **власну чергу** й обробляє у своєму темпі. SMS-сервіс лежить? Його повідомлення спокійно чекають у черзі.



кожен отримує свою копію й обробляє у своєму темпі;
SMS-сервіс лежить — його повідомлення чекають у черзі

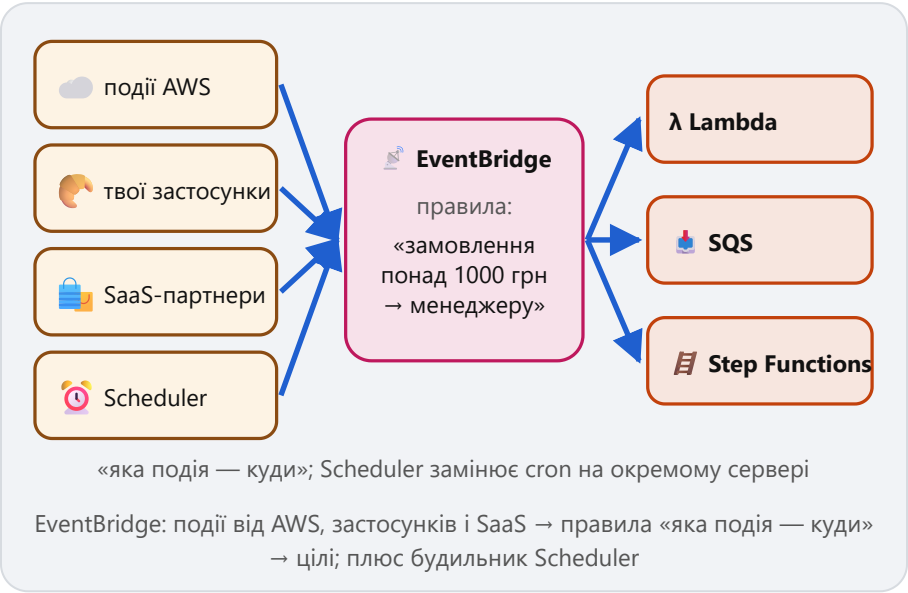
Fan-out: одне замовлення → SNS → окрема черга SQS для кухні, бухгалтерії й SMS; кожен обробляє у своєму темпі

🎯 «Одне повідомлення мають незалежно обробити кілька систем» → **SNS fan-out** → **кілька SQS**.

EventBridge — диспетчерська з правилами

Amazon EventBridge («івент брідж», колишні CloudWatch Events) — **диспетчерська**: сюди стікаються **події** — від сервісів AWS («сервер зупинився», «файл з'явився»), від **твоїх застосунків** і навіть від **SaaS-партнерів** (Zendesk, Shopify, Datadog…). А **правила** з шаблонами вирішують: «**яка** подія — **куди**» (у Lambda, SQS, Step Functions тощо).

- **EventBridge Scheduler** — **будильник** без серверів: «щоночі о 2:00 запусти звіт», «через 30 днів нагадай». Замість скрипту з cron на окремому сервері.
- Ще вміє: **архів і повтор** подій, реєстр схем подій, шини між акаунтами, **Pipes** (джерело → фільтр і збагачення → ціль).



«яка подія — куди»; Scheduler замінює cron на окремому сервері

EventBridge: події від AWS, застосунків і SaaS → правила «яка подія — куди» → цілі; плюс будильник Scheduler

Step Functions — головлікар з планом лікування

AWS Step Functions («степ фанкшнз») — **головлікар з покроковим планом**: описуєш процес як послідовність кроків (state machine — «машина станів»), а сервіс веде його сам.

- послідовні кроки, **розгалуження** («якщо оплата не пройшла — ...»), **паралельні** гілки;
- автоматичні **повтори** при помилках і обробка збоїв (retry/catch);
- **очікування** й **ручне схвалення** (процес чекає, поки людина натисне «так»);
- прямі інтеграції з сотнями сервісів AWS — без «клею» з функцій.

Два види:

- **Standard** — до **1 року**, кожен крок рівно один раз, повна історія виконання. Для довгих бізнес-процесів.
- **Express** — до **5 хвилин**, мільйони коротких виконань, дешевше.



Standard — до року (довгі процеси) · Express — до 5 хв (масово)

Step Functions: процес замовлення торта з перевіркою оплати, повторами, ручним схваленням кондитера й доставкою

🎯 «Багатокроковий процес з повторами й ручним схваленням на кілька днів» → **Step Functions Standard**.

Amazon MQ і AppFlow

- **Amazon MQ** — керований **ActiveMQ** чи **RabbitMQ**: для **перенесення** старих застосунків, які вже говорять стандартними протоколами черг (**JMS, AMQP, MQTT, STOMP**), **без переписування коду**. Для нових застосунків обирають SQS і SNS.

- **Amazon AppFlow** — переносить дані між SaaS-сервісами (Salesforce, SAP, Zendesk, Slack) і AWS (S3, Redshift) **без коду**, за розкладом чи подією.

Що обрати — одна таблиця

Потреба	Сервіс
Черга між частинами, обробники самі забирають задачі	SQS
Одне повідомлення — багатьом одержувачам	SNS (fan-out → SQS)
Маршрутизація подій за вмістом, SaaS-події, розклад	EventBridge
Строгий порядок і без дублів	SQS FIFO / SNS FIFO
Потік даних у реальному часі, кілька споживачів, перемотка	Kinesis Data Streams (розділ про потоки)
Багатокроковий процес з помилками, повторами, схваленням	Step Functions
Перенести застосунок з JMS/AMQP/MQTT без переписування	Amazon MQ

🥐 Тепер сайт «Хмаринки» лише кладе замовлення в тему SNS і одразу каже клієнтові «Дякуємо!». Кухня, бухгалтерія й SMS мають власні черги SQS. Великі кастомні торти ведуть Step Functions: перевірка оплати → схвалення кондитером → випікання → доставка. У наступну Чорну п'ятницю не загубилося жодне замовлення — просто кухонна черга ненадовго виросла до кількох сотень.

📖 **Словничок**

Tight / loose coupling — тісне / слабке зв'язування; **decoupling** — розв'язування через посередника.

SQS («ес-к'ю-ес») — черга: pull, зберігання до 14 днів (4 за замовчуванням), повідомлення до 1 МіВ.

Visibility timeout — час «невидимості» взятого повідомлення (30 с за замовчуванням, до 12 год).

DeleteMessage — обробник сам видаляє оброблене повідомлення.

DLQ («ді-ел-к'ю») — черга безнадійних повідомлень після N спроб.

Long polling — чекати нових повідомлень до 20 с замість частих порожніх запитів.

Standard / FIFO — швидко з можливими дублями й перемішуванням / строгий порядок і рівно один раз.

SNS («ес-ен-ес») — pub/sub, push: тема і підписники; фільтри; нічого не зберігає.

Fan-out («фен-аут») — SNS → кілька SQS, у кожного обробника своя копія.

EventBridge («івент брідж») — шина подій з правилами; **Scheduler** — будильник без серверів.

Step Functions — покроковий процес: Standard (до року) і Express (до 5 хв).

Amazon MQ — керований ActiveMQ/RabbitMQ для старих протоколів. **AppFlow** — дані між SaaS і AWS без коду.

🧩 Міні-тест: перевір себе

? У пікові години частина замовлень губиться між сайтом і кухнею. Що допоможе?

- **А** Поставити чергу SQS між компонентами
- **Б** Купити потужніший сервер кухні
- **В** SNS без черг
- **Г** Збільшити тайм-аут браузера

Відповідь: А. Черга зберігає замовлення до 14 днів і згладжує піки — нічого не губиться.

? Одне замовлення мають незалежно обробити кухня, бухгалтерія і SMS-сервіс. Що обрати?

- **А** Одна спільна черга SQS для всіх
- **Б** SNS fan-out → окрема черга SQS для кожного
- **В** Три окремі дзвінки напряму
- **Г** Kinesis без споживачів

Відповідь: Б. Fan-out: кожен отримує власну копію у власну чергу й обробляє у своєму темпі.

? Фінансові транзакції мають оброблятися строго по порядку й без дублів. Що обрати?

- **А** SQS Standard
- **Б** SQS FIFO
- **В** SNS без фільтрів
- **Г** EventBridge Scheduler

Відповідь: Б. FIFO — строгий порядок у групі й обробка рівно один раз.

? Повідомлення з черги обробляються двічі, бо обробка триває довше, ніж очікувалось. Що зробити?

- **А** Збільшити visibility timeout
- **Б** Зменшити visibility timeout
- **В** Вимкнути DLQ
- **Г** Перейти на SNS

Відповідь: А. Поки талончик «у руках лікаря», інші його не бачать. Замалий тайм-аут — і талончик повертається в чергу.

? Деякі повідомлення постійно падають з помилкою й забивають чергу. Що налаштувати?

А Long polling

Б Dead-letter queue

В Delay queue

Г FIFO

Відповідь: Б. DLQ збирає безнадійні повідомлення після кількох невдалих спроб — для окремого аналізу.

? Процес із кількох кроків з повторами при помилках і ручним схваленням, що може тривати кілька днів. Що обрати?

А Одна функція Lambda

Б Step Functions Standard

В Step Functions Express

Г SQS Standard

Відповідь: Б. Standard живе до року й уміє чекати людину. Express — до 5 хвилин.

? Потрібно щоночі о 2:00 запускати задачу без власних серверів. Що обрати?

- **А** EventBridge Scheduler
- **Б** Cron на окремому сервері EC2
- **В** SNS FIFO
- **Г** Amazon MQ

Відповідь: А. Планувальник EventBridge — «будильник» без серверів. Cron на сервері — єдина точка відмови.

? Старий застосунок працює з RabbitMQ; треба перенести в AWS без переписування коду. Що обрати?

А SQS

Б SNS

В Amazon MQ

Г Kinesis

Відповідь: В. Amazon MQ — керований RabbitMQ/ActiveMQ для застосунків на стандартних протоколах черг.

✅ **Головне**

Слабке зв'язування через посередника: піки згладжуються, падіння однієї частини не валить інших (**decouple**, **buffer**).

SQS — черга (pull): до **14 днів**, до **1 MiB**; **visibility timeout** (двічі обробляють → збільшити); обробник сам **видаляє**; **DLQ** для безнадійних; **long polling** до 20 с; **Standard** (дублі можливі) чи **FIFO** (порядок, рівно один раз).

SNS — гучномовець (push), нічого не зберігає; **fan-out** SNS → кілька SQS; фільтри для підписників.

EventBridge — події від AWS, застосунків і SaaS за правилами; **Scheduler** — cron без серверів.

Step Functions — покрокові процеси з повторами й схваленням (**Standard** до року, **Express** до 5 хв); **Amazon MQ** — старі протоколи без переписування; **AppFlow** — SaaS ↔ AWS без коду.

🗃️ Три види сховищ: блоки, файли й об'єкти

👴 Простими словами

Сховища бувають трьох видів: диск, прикручений до одного комп'ютера (блоки), спільна папка для багатьох (файли) і безрозмірний склад, куди кладеш речі з ярликами (об'єкти).

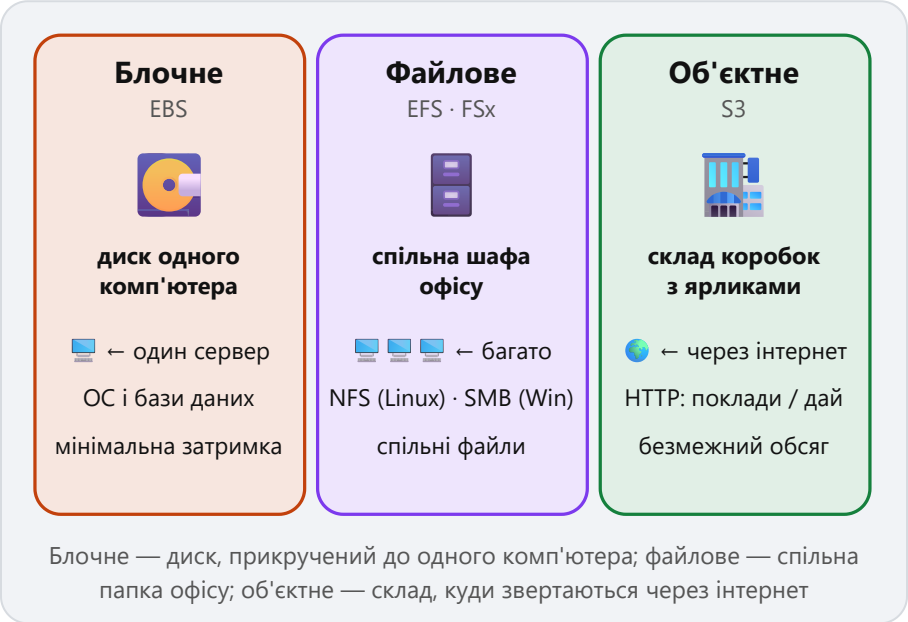
🎯 Що ти дізнаєшся

- чим блочне, файлове й об'єктне сховища відрізняються — на образах з життя
- що таке IOPS, пропускна здатність і затримка для дисків
- чим випадковий доступ відрізняється від послідовного
- що означає «постійне» і «тимчасове» сховище
- який сервіс AWS до якого виду належить

🥐 У «Хмаринки» вже багато даних: база замовлень, тисячі фото тортів, спільна папка з рецептами для всіх кондитерів, щоденні бекапи. «Покладу все в одне місце», — пропонує Оля. «Це як тримати борошно, торти й документи в одній шафі, — усміхається Тарас. — Зберігати можна по-різному, і кожен спосіб — для свого».

Три способи тримати дані

Образ з простого курсу — **три способи зберігати документи**:



Блочне — диск, прикручений до одного комп'ютера; файлове — спільна папка офісу; об'єктне — склад, куди звертаються через інтернет

Блочне сховище: зошит з пронумерованими сторінками

Блочне сховище (block storage) — це «сирий» диск, поділений на пронумеровані **блоки**, як зошит зі сторінками. Операційна система сервера сама розмічає його під файли й папки.

- Підключається до **одного** сервера (як жорсткий диск у комп'ютері).
- Дуже **мала затримка**: можна змінити одну «сторінку», не чіпаючи решти.
- Для **операційної системи** й **баз даних**.

В AWS: **EBS** (мережевий диск для EC2) і **instance store** (локальні диски самого фізичного сервера).

Файлове сховище: спільна шафа офісу

Файлове сховище (file storage) — звичні **папки й файли**, які одночасно бачать **багато** серверів по мережі. Як спільна шафа з папками в офісі, до якої мають доступ усі працівники.

- Працює через мережеві протоколи: **NFS** (для Linux, порт 2049) або **SMB** (для Windows, порт 445).
- Для спільних документів, вмісту сайтів, домашніх папок працівників, спільних даних для багатьох серверів.

В AWS: **EFS** (для Linux) і сімейство **FSx** (для Windows і спеціальних задач).

Об'єктне сховище: безрозмірний склад

Об'єктне сховище (object storage) — **склад коробок з ярликами**. Кожен **об'єкт** — це файл разом з описом (метаданими) і унікальним **ключем**-адресою.

- Звертаються через **інтернет-запити** (HTTP API): «поклади коробку», «дай коробку».
- Місткість практично **безмежна**.
- Папок насправді немає: «папка» — лише частина ключа (`cakes/2026/tort.jpg`).
- Не можна змінити шматочок файлу — лише **замінити об'єкт цілком**. І не можна «змонтувати» такий склад як диск для бази даних.

В AWS: **S3** — про нього наступний розділ.

Як міряють швидкість сховища

Ти вже знаєш ці слова — тепер про диски:

- **IOPS** («ай-опс») — скільки **окремих операцій** читання чи запису за секунду. Образ — **скільки коробок за секунду комірник виносить з комори**. Важливо для баз даних, де багато дрібних операцій.

- **Пропускна здатність** (throughput) — скільки **мегабайтів за секунду**. Образ — **скільки кілограмів за секунду** йде через двері. Важливо для відео, логів, аналітики — великих файлів.
- **Затримка** (latency) — скільки чекати **одну** відповідь.

І ще одна пара слів:

- **Випадковий доступ** — дрібні шматочки з різних місць (база даних шукає то одне замовлення, то інше). Потрібні високі IOPS — **SSD**.
- **Послідовний доступ** — великі шматки підряд (читаємо відео чи журнал від початку до кінця). Важлива пропускна здатність — тут згодяться й дешевші **HDD**.

Постійне чи тимчасове

- **Постійне** (persistent) сховище переживає вимкнення сервера: EBS, EFS, FSx, S3.
- **Тимчасове** (ephemeral, «іфемерал») — зникає разом із сервером: **instance store**. Образ з простого курсу — **записи крейдою на дошці**: найшвидше, але стирається при зупинці. Лише для кешу й тимчасових даних.

Що є що — одна таблиця

	Блочне	Файлове	Об'єктне
Образ	диск одного комп'ютера	спільна шафа офісу	склад з ярликами
Хто користується	один сервер	багато серверів	будь-хто через інтернет
Як звертаються	як до диска	NFS / SMB	HTTP-запити
Обсяг	заданий розмір	росте за потребою	практично безмежний
Для чого	ОС, бази даних	спільні файли	фото, відео, бекапи, логи, озера даних
AWS	EBS, instance store	EFS, FSx	S3

🎯 Мнемоніка з простого курсу: **Блок — один сервер (EBS). Файли — багато серверів (EFS для Linux, FSx для Windows і HPC). Об'єкти — через HTTP (S3).**

📖 Словничок

Блочне сховище — «сирий» диск для одного сервера: EBS, instance store.

Файлове сховище — спільні папки для багатьох серверів: EFS, FSx.

Об'єктне сховище — склад об'єктів з ключами через HTTP: S3.

NFS / SMB — мережеві протоколи спільних папок для Linux (порт 2049) / Windows (порт 445).

IOPS («ай-опс») — операцій за секунду; **throughput** — мегабайтів за секунду; **latency** — час однієї відповіді.

Випадковий / послідовний доступ — дрібні шматочки звідусіль / великі шматки підряд.

Persistent / ephemeral — переживає вимкнення / зникає разом із сервером.

🧩 Міні-тест: перевір себе

? Базі даних на сервері потрібен диск з мінімальною затримкою. Який вид сховища?

- **А** Блочне (EBS)
- **Б** Об'єктне (S3)
- **В** Файлове через SMB
- **Г** Архів

Відповідь: А. База працює з дрібними шматочками — їй потрібен блочний диск, підключений до сервера.

? Спільна папка, яку одночасно використовують 50 Linux-серверів. Що обрати?

- **А** Блочний диск на кожному сервері
- **Б** Файлове сховище (EFS)
- **В** Instance store
- **Г** Об'єктне сховище як диск

Відповідь: Б. Файлове сховище — спільна «шафа» для багатьох серверів. Для Linux — EFS.

? Мільйони фото тортів, доступ через інтернет, обсяг росте без меж. Що обрати?

- **А** Блочний диск
- **Б** Instance store
- **В** Об'єктне сховище (S3)
- **Г** Файлову систему Windows

Відповідь: В. Склад об'єктів: практично безмежний і доступний через HTTP.

? Чи можна «змонтувати» S3 як диск для бази даних?

А Так, це найкращий варіант

Б Ні: S3 — об'єктне сховище, а базі потрібен блочний диск

В Лише в регіоні us-east-1

Г Так, якщо ввімкнути шифрування

Відповідь: Б. Об'єкти можна лише класти й брати цілком через HTTP. Для бази — EBS.

? Що таке IOPS?

А Кількість операцій читання чи запису за секунду

Б Об'єм диска в гігабайтах

В Швидкість інтернету

Г Кількість серверів

Відповідь: А. IOPS — «скільки коробок за секунду». Мегабайти за секунду — це пропускна здатність.

? Відеофайли читають великими шматками від початку до кінця. Що тут найважливіше?

- **А** IOPS
- **Б** Пропускна здатність (МБ/с)
- **В** Кількість ядер
- **Г** DNS

Відповідь: Б. Послідовне читання великих файлів — справа пропускної здатності; тут згодяться і дешеві HDD.

✅ **Головне**

Блочне — диск одного сервера (EBS, instance store): ОС і бази. **Файлове** — спільна шафа для багатьох (EFS — Linux/NFS, FSx — Windows/SMB і спецзадачі). **Об'єктне** — склад через HTTP (S3): безмежний, але не диск.

IOPS — операції за секунду (бази, SSD); **throughput** — МБ/с (відео, логи, HDD); **latency** — час відповіді.

Випадковий доступ → високі IOPS; **послідовний** → пропускна здатність.

Instance store — тимчасовий («крейда на дошці»); решта — постійні.

Мнемоніка: **Блок — один сервер. Файли — багато серверів. Об'єкти — через HTTP.**

🪣 S3 докладно: безрозмірний склад

👴 Простими словами

S3 — безрозмірний склад. Кладеш будь-які файли, кожен отримує свій ярлик-адресу. Можна вибрати дешевші «далекі полиці» для рідко потрібного й правила, які самі переносять старе туди.

🎯 Що ти дізнаєшся

- як влаштовані bucket, об'єкт і ключ
- що означають «одинадцять дев'яток»
- вісім класів зберігання — від передпокою до сховища в горах — і правило «30 · 90 · 180»
- як lifecycle сам переносить дані
- версії, реплікація й Object Lock
- як захистити сховище і дати тимчасовий доступ
- як зробити S3 швидшим

🥐 Фото тортів «Хмаринки» займають уже 200 ГБ і щодня їх більшає. Бухгалтерія вимагає зберігати документи 10 років, і так, щоб ніхто не міг їх підправити. А старі фото, яким уже рік, ніхто не відкриває, але викидати шкода. «Для всього цього — S3, — каже Тарас. — Безрозмірний склад, де для кожної речі є своя полиця за своєю ціною».

Bucket, об'єкт і ключ

Amazon S3 («ес-срі», Simple Storage Service — простий сервіс зберігання) — об'єктне сховище AWS. Образ з простого курсу — **безрозмірний склад**:

- **Bucket** («бакет», відро) — **окремий склад** з назвою, **унікальною на весь світ** (бо вона стає частиною інтернет-адреси): `khmarynka-photos`. Назва — малими латинськими літерами, від 3 до 63 символів. Bucket живе **в одному регіоні**.
- **Object** (об'єкт) — **коробка**: сам файл плюс опис (метадані).
- **Key** (ключ) — **ярлик-адреса** коробки: `cakes/2026/tort-17.jpg`. «Папки» — лише частина ключа, яку консоль показує як папки. Частину ключа до останнього слеша називають **префіксом** (prefix).

Розміри:

- один об'єкт — до **50 TB** (з грудня 2025; раніше — 5 TB, і на іспиті може трапитися стара цифра);
- одним запитом можна завантажити до **5 GB**; для великих файлів — **multipart upload** (завантаження частинами паралельно; рекомендують від 100 MB, обов'язково понад 5 GB). Зірвалася одна частина — перевантажують лише її.

Одинадцять дев'яток

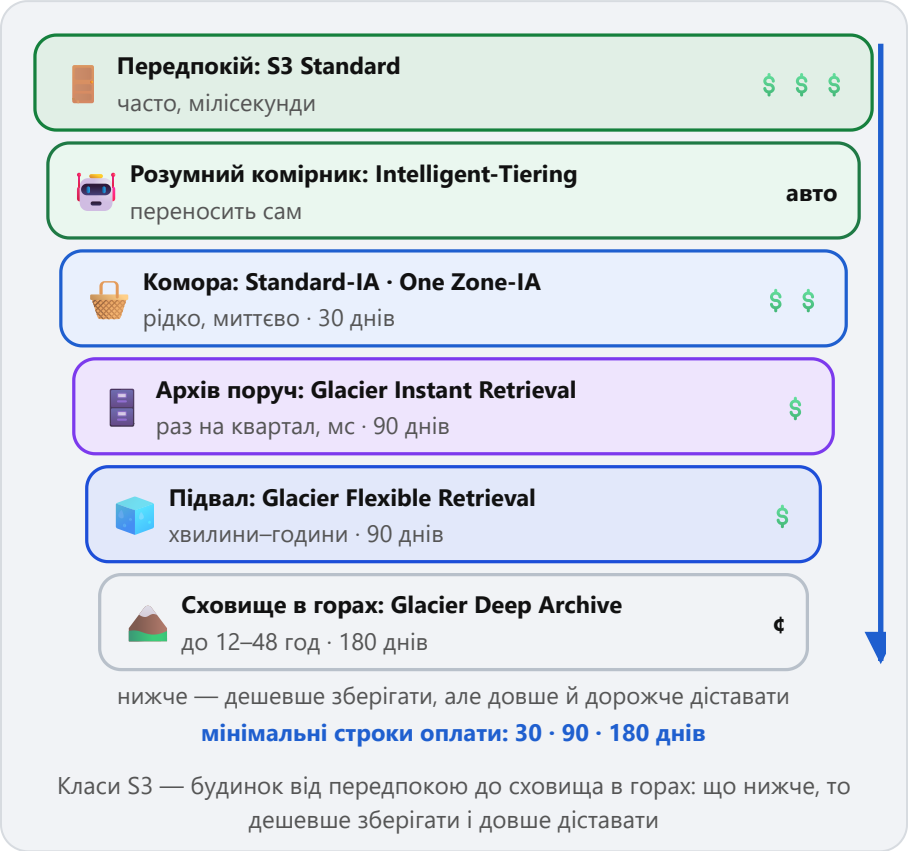
S3 розраховано на надійність зберігання **99,999999999%** — «одинадцять дев'яток». Образно: з 10 мільйонів коробок у середньому губиться одна **за 10 000 років**. Для цього більшість класів тримає копії **щонайменше у трьох AZ**.

Ще кілька властивостей: доступність класу Standard — 99,99%; щойно записаний об'єкт одразу видно для читання (strong read-after-write consistency) — ніяких «зачекай хвилинку».

> ⚠️ Пам'ятаєш: **durability** (речі цілі) ≠ **availability** (двері відчинені). 11 дев'яток — це саме durability.

Класи зберігання: будинок від передпокою до сховища в горах

Не всі речі потрібні однаково часто. Тому в S3 є **класи зберігання** — полиці з різною ціною: що рідше потрібна річ, то дешевше її тримати, але дорожче й довше діставати.



Класи S3 — будинок від передпокою до сховища в горах: що нижче, то дешевше зберігати і довше діставати

Клас	Образ	Для чого	Мін. строк оплати	Доступ
Standard	передпокій	часто потрібні дані	—	мілісекунди

Клас	Образ	Для чого	Мін. строк оплати	Доступ
Intelligent-Tiering	розумний комірник	доступ невідомий чи змінний	—	мілісекунди (архівні рівні — години)
Standard-IA	комора	рідко, але миттєво	30 днів	мілісекунди + плата за отримання
One Zone-IA	комора в одному районі	рідко, дані можна відтворити	30 днів	мілісекунди, лише одна AZ
Glacier Instant Retrieval	архів поруч	~раз на квартал, але миттєво	90 днів	мілісекунди
Glacier Flexible Retrieval	підвал	архіви й бекапи	90 днів	від хвилин до годин
Glacier Deep Archive	сховище в горах	7–10+ років для регулятора	180 днів	до 12–48 годин
Express One Zone	касова стрічка	найшвидший, одна AZ	1 година	одноцифрові мілісекунди

Кілька важливих деталей:

- **Мнемоніка мінімальних строків: 30 · 90 · 180** (IA · Glacier · Deep Archive). Видалив раніше — однаково заплатиш за весь мінімальний строк.
- У Standard-IA, One Zone-IA і Glacier Instant Retrieval мінімальний розмір для оплати — **128 KB** (дрібний файл оплачується як 128 KB).
- **Glacier Flexible Retrieval** дістають трьома темпами: **Expedited** (1–5 хв), **Standard** (3–5 год), **Bulk** (5–12 год, безкоштовно). Гарантувати швидкий темп навіть у пік допомагає **provisioned capacity**.

- **Deep Archive**: Standard — до 12 годин, Bulk — до 48 годин.
- **Intelligent-Tiering** — **розумний комірник**: сам переносить об'єкти між рівнями (30 днів без звернень → «рідко», 90 → архівний миттєвий; за бажанням — і в глибші архіви). Бере невелику плату за нагляд, зате **без плати за отримання**. Ідеальний, коли **не знаєш**, як часто читатимуть дані.
- **One Zone-IA** і **Express One Zone** живуть в **одній AZ**: втратиш зону — можеш втратити дані. Лише для того, що можна відтворити (мініатюри, копії).

🎯 Рідко, але миттєво → **Standard-IA**. Невідомий патерн → **Intelligent-Tiering**. Роки для регулятора, 48 годин не страшно → **Deep Archive**. Відтворювані дані дешево → **One Zone-IA**.

Lifecycle: речі самі переїжджають з віком

Lifecycle rules («лайфсайкл рулз», правила життєвого циклу) — **автоматичне переселення** речей з віком. Наприклад, для фото «Хмаринки»:



- через 30 днів — у Standard-IA;
- через рік — у Glacier Flexible Retrieval;
- через 10 років — видалити.

Правила вміють **переносити** (transition) і **видаляти** (expiration), зокрема старі версії об'єктів і незавершені завантаження частинами. У Standard-IA не можна перенести раніше ніж через **30 днів** після створення.

Як зрозуміти, коли переносити? **Storage Class Analysis** проаналізує, як часто читають дані, і підкаже. А **S3 Storage Lens** покаже загальну картину використання й витрат.

Versioning: машина часу

Versioning («версіонінг») — **машина часу** для сховища: кожен перезапис створює **нову версію**, а стара

лишається. Видалення лише ставить **delete marker** («позначку видалення») — стару версію можна повернути.

- Захист від випадкового видалення й перезапису (і від деяких вірусів-шифрувальників).
- **MFA Delete** — щоб остаточно видалити версію, потрібен код MFA (вмикає лише root).

Replication: філія складу

Replication («реплікейшн») — автоматичне копіювання **нових** об'єктів в інший bucket:

- **CRR** (Cross-Region Replication) — в **інший регіон**: аварійне відновлення, близькість до клієнтів, вимоги законів;
- **SRR** (Same-Region Replication) — у **тому самому** регіоні: зібрати логи з кількох сховищ, копія робочих даних у тестовий акаунт.

Умови й деталі: versioning має бути **увімкнений з обох боків**; реплікуються лише **нові** об'єкти, а вже наявні — окремою операцією **S3 Batch Replication**; **Replication Time Control** (RTC) гарантує, що 99,99% об'єктів скопіюються за 15 хвилин.

Object Lock: сейф з таймером

Object Lock («обджект лок») — **WORM**-захист (write once, read many — «записати раз, читати багато»): об'єкт не можна змінити чи видалити до кінця строку.

- **Compliance mode** — **ніхто**, навіть root, не видалить і не змінить до кінця строку. Для регуляторів.
- **Governance mode** — можуть обійти лише користувачі з особливим дозволом.
- **Legal hold** — «заморожено» без строку, доки не знімуть.

> 🎯 «Ніхто, навіть адміністратор, не може видалити чи змінити записи 7 років» → **Object Lock, compliance mode**.

Безпека складу

- **Block Public Access** — «заборона публічного доступу» — **увімкнена за замовчуванням**. Щоб сховище стало публічним, її треба свідомо вимкнути.
- Старі списки доступу (ACL) за замовчуванням **вимкнені**; доступ дають **bucket policy** та IAM-політиками.
- **Шифрування**: SSE-S3 за замовчуванням, SSE-KMS для аудиту (розділ про шифрування).

- **Presigned URL** («пресайнд ю-ар-ел») — **тимчасовий пропуск на одну коробку**: посилання, що дозволяє скачати чи завантажити конкретний приватний об'єкт протягом заданого часу. Діє з правами того, хто його підписав.
- **Access Points** — окремі «входи» до великого складу для різних команд і застосунків, кожен зі своєю політикою (можна «лише з нашої VPC»). **Multi-Region Access Points** — один глобальний вхід до копій у кількох регіонах.
- **CORS** — дозвіл браузеру з іншого домену звертатися до сховища.

Приклад bucket policy, що забороняє будь-який доступ **без HTTPS**:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "DenyHTTP",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:*",
    "Resource": ["arn:aws:s3:::khmarynka-photos", "arn:aws:s3:::khmarynka-pho
    "Condition": { "Bool": { "aws:SecureTransport": "false" } }
  }]
}
```

Прочитай її самостійно: **Deny**, **усім** (`*`), **усі дії S3**, на сховище і всі об'єкти, **якщо** з'єднання не захищене. Ти

вже вмієш це читати!

Швидкість S3

- S3 витримує **3 500 запитів на запис і 5 500 на читання за секунду на кожен префікс**. Потрібно більше — розкладай ключі по **кількох префіксах** (`cakes/a/...`, `cakes/b/...`).
- **Multipart upload** — великі файли частинами паралельно.
- **Transfer Acceleration** — клієнти з далеких країн завантажують через найближчу edge-локацію, а далі — швидкою мережею AWS.
- **Byte-range fetches** — скачувати різні частини великого файлу паралельно.

Ще кілька вмінь

- **Event notifications** — «щойно з'явився файл — повідом»: у SNS, SQS, Lambda чи EventBridge. Так фото торта автоматично стає мініатюрою.
- **Static website hosting** — S3 уміє сам роздавати простий статичний сайт, але лише по HTTP. Для HTTPS і власного домену ставлять **CloudFront + OAC** (сховище лишається приватним).

- **Requester Pays** — за завантаження платить той, хто завантажує (для великих публічних наборів даних).
- **S3 Batch Operations** — одна дія над мільярдами об'єктів: скопіювати, змінити теги, відновити з архіву, викликати функцію.
- **Inventory** — список усіх об'єктів з властивостями.
- S3 Select і S3 Object Lambda закриті для нових клієнтів — для запитів до даних у S3 використовують Athena (розділ про аналітику).

🧪 **Спробуй у справжньому AWS: Створи склад і поділися файлом через тимчасове посилання (10 хвилин)**

Кілька мегабайтів у S3 коштують копійки; наприкінці ми все видалимо.

1. У консолі відкрий **S3** → **Create bucket**.
2. Назва — унікальна, малими латинськими літерами, наприклад `khmarynka-test-` + сьогоднішня дата + твої ініціали. Регіон — свій.
3. **Block Public Access** залиш увімкненим. Натисни **Create bucket**.
4. Відкрий bucket → **Upload** → додай будь-яке фото → **Upload**.

5. Спробуй відкрити **Object URL** з деталей об'єкта — отримаєш **AccessDenied**: сховище приватне.
6. Тепер обери об'єкт → **Actions** (або Object actions) → **Share with a presigned URL**, вкажи 5 хвилин і скопіюй посилання.
7. Відкрий його в браузері — фото показується. Через 5 хвилин те саме посилання перестане працювати.
8. Прибери за собою: видали об'єкт, потім **Delete bucket**.

📖 **Словничок**

S3 («ес-срі») — об'єктне сховище: 11 дев'яток надійності.

Bucket («бакет») — «склад» з унікальною на весь світ назвою, живе в регіоні.

Object / key / prefix — «коробка» / її повна адреса / частина адреси до останнього слеша.

Multipart upload — завантаження частинами (від 100 MB рекомендовано, понад 5 GB обов'язково).

Класи: Standard, Intelligent-Tiering, Standard-IA, One Zone-IA, Glacier Instant / Flexible / Deep Archive, Express One Zone.

30 · 90 · 180 — мінімальні строки оплати IA · Glacier · Deep Archive.

Lifecycle rules — автоматичні перенесення між класами й видалення.

Versioning / delete marker / MFA Delete — машина часу / позначка видалення / видалення версій лише з MFA.

CRR / SRR / RTC — реплікація в інший регіон / у той самий / гарантія «99,99% за 15 хв».

Object Lock — WORM: compliance (ніхто), governance (з особливим дозволом), legal hold (без строку).

Block Public Access — заборона публічного доступу, увімкнена за замовчуванням.

Presigned URL — тимчасове посилання на приватний об'єкт.

Access Points — окремі входи з власними політиками.

Transfer Acceleration — прискорене завантаження через edge-локації.

Event notifications — сповіщення про нові чи видалені об'єкти.

🧩 Міні-тест: перевір себе

? Який найбільший розмір одного об'єкта S3 зараз?

- **А** 5 GB
- **Б** 5 TB
- **В** 50 TB
- **Г** Без обмежень

Відповідь: В. З грудня 2025 — 50 TB (раніше 5 TB). Одним запитом — до 5 GB, більше — частинами.

? Дані читають рідко, але коли треба — миттєво. Який клас?

- **А** S3 Standard-IA
- **Б** Glacier Deep Archive
- **В** Glacier Flexible Retrieval
- **Г** Express One Zone

Відповідь: А. «Комора»: рідко, але за мілісекунди. Раз на квартал — ще дешевше в Glacier Instant Retrieval.

? Документи треба зберігати 10 років для регулятора, доступ протягом 48 годин прийнятний, потрібно найдешевше. Який клас?

А Standard

Б Standard-IA

В Glacier Deep Archive

Г Intelligent-Tiering

Відповідь: В. «Сховище в горах» — найдешевше, діставати до 12–48 годин.

? Ніхто, навіть root, не повинен мати змоги видалити чи змінити записи протягом 7 років. Що обрати?

А Versioning

Б Object Lock у режимі compliance

В Object Lock у режимі governance

Г Block Public Access

Відповідь: Б. Compliance — «сейф з таймером», який не відчинить ніхто. Governance можна обійти з особливим дозволом.

? Невідомо, як часто читатимуть дані, і вручну керувати класами не хочеться. Що обрати?

- **А** S3 Intelligent-Tiering
- **Б** One Zone-IA
- **В** Glacier Deep Archive
- **Г** Express One Zone

Відповідь: А. «Розумний комірник» сам переносить об'єкти, без плати за отримання.

? Клієнту треба тимчасово дати скачати приватний файл. Що використати?

- **А** Зробити сховище публічним
- **Б** Presigned URL
- **В** Вимкнути шифрування
- **Г** Transfer Acceleration

Відповідь: Б. Presigned URL — тимчасовий пропуск на одну коробку з правами того, хто його підписав.

? Кожен новий об'єкт має автоматично копіюватися в інший регіон. Що налаштувати?

А Cross-Region Replication (з увімкненим versioning з обох боків)

Б Lifecycle rule

В Object Lock

Г Multipart upload

Відповідь: А. CRR копіює нові об'єкти; наявні — через Batch Replication.

? Застосунок робить дуже багато запитів до S3 і впирається в ліміт. Що допоможе?

А Один довгий префікс для всіх файлів

Б Розподілити ключі по кількох префіксах

В Перейти на Glacier

Г Вимкнути versioning

Відповідь: Б. Ліміти 3 500 / 5 500 запитів за секунду діють на кожен префікс. Більше префіксів — більше швидкості.

✅ **Головне**

Bucket (унікальна назва, регіон) → **об'єкти** з **ключами**; «папки» — префікси. Об'єкт до **50 TB**, запит до 5 GB, великі — **multipart**. **11 дев'яток** — durability.

Класи: **Standard** → **Intelligent-Tiering** (невідомий доступ) → **Standard-IA / One Zone-IA** → **Glacier Instant / Flexible / Deep Archive**; мінімуми **30 · 90 · 180**; One Zone — лише відтворюване.

Lifecycle — автоперенесення й видалення; **versioning** — машина часу; **CRR/SRR** — копії (потрібен versioning, нові об'єкти); **Object Lock compliance** — не видалить ніхто.

Безпека: **Block Public Access** за замовчуванням, bucket policy + IAM, шифрування, **presigned URL**, Access Points; сайт з HTTPS → **CloudFront + OAC**.

Швидкість: **3 500 / 5 500** запитів за секунду **на префікс**, multipart, **Transfer Acceleration**; події → SNS/SQS/Lambda/EventBridge.

💿 Диски й файлові системи: EBS, instance store, EFS, FSx

👴 Простими словами

EBS — диск, прикручений до одного комп'ютера. Instance store — швидка тимчасова шухляда, яка очищається, коли комп'ютер зупиняють. EFS і FSx — спільні папки, які бачать багато комп'ютерів одночасно.

🎯 Що ти дізнаєшся

- як працює диск EBS і чому він живе в одній AZ
- типи EBS — від універсального gp3 до гоночного io2 і вантажного st1
- знімки дисків і як перенести диск в іншу зону чи регіон
- чим небезпечний instance store
- коли потрібна EFS, а коли одна з файлових систем FSx

🥐 У «Хмаринки» три нові задачі. База замовлень на сервері гальмує на дешевому диску. Кондитерам у двох цехах потрібна одна спільна папка з рецептами. А бухгалтерія працює на Windows і хоче звичну мережеву папку з входом під корпоративними обліковками. «Три задачі — три різні сховища», — каже Тарас.

EBS — диск, прикручений до сервера

Amazon EBS («і-бі-ес», Elastic Block Store) — блочний диск для серверів EC2. Образ з простого курсу — **жорсткий диск, прикручений до одного комп'ютера** (насправді він мережевий, але працює як власний).

- Диск живе в **одній AZ** — підключити його можна лише до сервера **в тій самій** зоні.
- Зазвичай підключений до **одного** сервера. Виняток — **Multi-Attach** для io1/io2: до 16 серверів у тій самій AZ (для кластерних програм, що вміють ділити диск).
- Диск **живе окремо** від сервера: його можна відчепити й прикрутити до іншого сервера в тій самій AZ. Системний диск за замовчуванням видаляється разом із сервером (прапорець «delete on termination»).

- Шифрування — однією галочкою, а для регіону можна ввімкнути **шифрування за замовчуванням**.

Типи EBS: від універсала до вантажівки



Тип	Образ	Для чого	Головні цифри
gp3 (SSD)	універсал	за замовчуванням: системні диски, більшість застосунків	базово 3 000 IOPS і 125 MiB/s **незалежно від розміру**; до 80 000 IOPS, 2 000 MiB/s, 64 TiB
gp2 (SSD)	старий універсал	старший тип	IOPS залежать від розміру: 3 на GiB, максимум 16 000
io2 Block Express (SSD)	гоночний болід	критичні бази	до 256 000 IOPS, 4 000 MiB/s, затримка менша за мілісекунду, Multi-Attach

Тип	Образ	Для чого	Головні цифри
io1 (SSD)	старий гоночний	старший тип	до 64 000 IOPS
st1 (HDD)	вантажівка	великі послідовні дані: логи, big data	до 500 MiB/s; **не може бути системним**
sc1 (HDD)	холодний фургон	рідко потрібні дані, найдешевше	до 250 MiB/s; **не може бути системним**

Кілька висновків, які часто трапляються на іспиті:

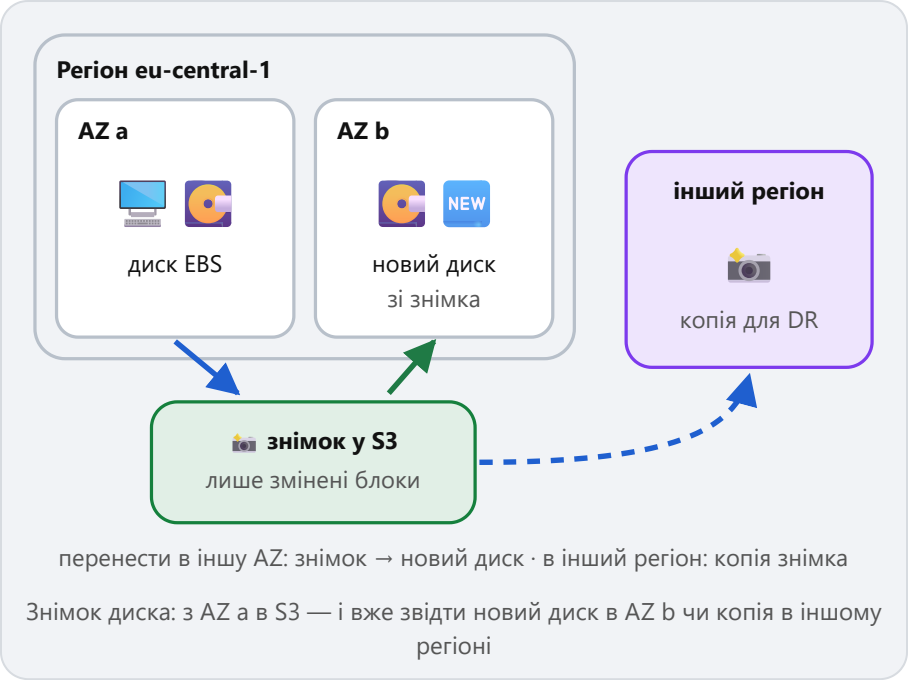
- **gp3 дешевший за gp2** і дозволяє купити потрібні IOPS окремо від розміру. Перехід gp2 → gp3 робиться **без зупинки** (Elastic Volumes) — і рахунок зменшується.
- Потрібно **понад 80 000 IOPS** на одному диску → **io2 Block Express**.
- HDD-диски (st1, sc1) дешеві, але **не можуть бути системними** і погано працюють з дрібними випадковими операціями.

Знімки: фотоальбом диска

Snapshot («снепшот») — **фотографія** диска на певний момент, яка зберігається в S3 (надійно, у кількох AZ). Знімки **інкрементні**: після першого повного кожен наступний зберігає лише **змінені** блоки — як фотоальбом, куди вклеюють лише нові сторінки.

Що можна робити зі знімками:

- **створити новий диск** з знімка — у **будь-якій AZ** регіону. Так і **переносять диск в іншу зону**: знімок → новий диск там;
- **скопіювати знімок в інший регіон** — для аварійного відновлення чи переїзду;
- автоматизувати: **Data Lifecycle Manager** (DLM) чи **AWS Backup** — «знімок щодня, зберігати 30 днів»;
- **Snapshot Archive** — дешевше для рідко потрібних знімків (відновлення довше);
- **Recycle Bin** — «кошик» для випадково видалених знімків;
- **Fast Snapshot Restore** — новий диск одразу працює на повній швидкості.



Elastic Volumes — змінити розмір, тип чи IOPS диска **на ходу**, без зупинки сервера.

Instance store — крейда на дошці

Instance store — диски, фізично вставлені в той самий сервер, на якому працює твій інстанс. **Найшвидші** (найвищі IOPS), але **тимчасові**: дані **зникають**, коли інстанс зупиняють, видаляють або коли ламається фізичний сервер.

Образ з простого курсу — **записи крейдою на дошці**. Лише для кешу, буферів, тимчасових файлів — або коли програма сама тримає копії на кількох серверах.

⚠️ Instance store **не переживає stop**. EBS — переживає.

EFS — спільна мережева папка для Linux

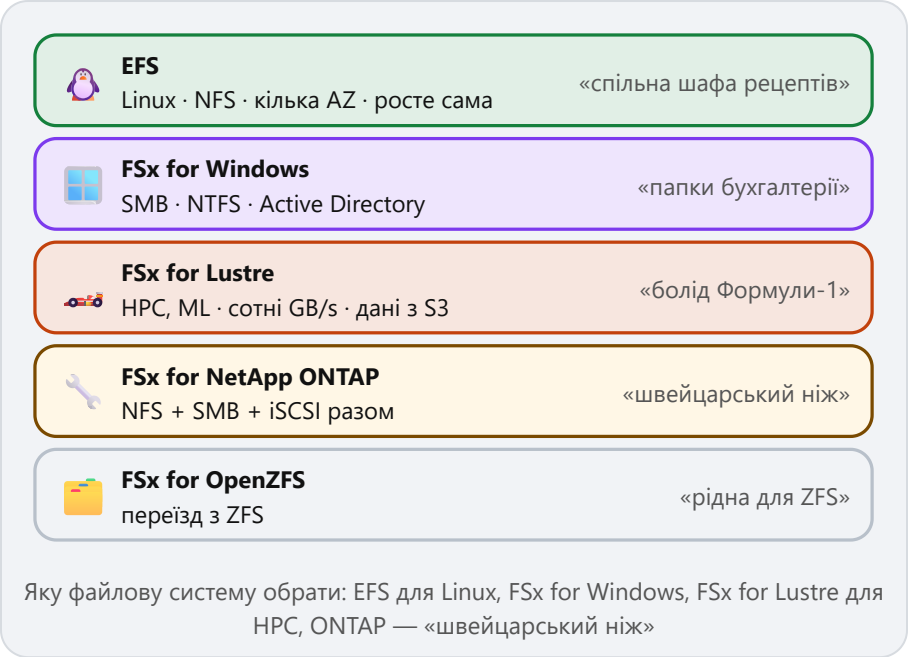
Amazon EFS («і-еф-ес», Elastic File System) — керована файлова система **NFS для Linux**. Образ — **спільна шафа з рецептами**, до якої одночасно ходять кондитери з обох цехів.

- Одночасно для **тисяч** серверів EC2, контейнерів (ECS, EKS) і навіть функцій Lambda.
- Працює **в кількох AZ** (є й дешевша One Zone).
- **Росте сама** — від кілобайтів до петабайтів; платиш за використане.
- Класи: **Standard**, **Infrequent Access**, **Archive** — і lifecycle, що сам переносить рідко відкривані файли в дешевші класи.
- Режими швидкості: **Elastic** (рекомендований — підлаштовується сам), Provisioned, Bursting.
- У кожній AZ створюють **mount target** — «точку підключення». Її Security Group має пропускати **NFS, порт 2049** від серверів. Забули — сервери не підключаться (timeout).
- Дорожча за EBS за гігабайт, зате **спільна**.

FSx — спеціалізовані файлові сервери

Amazon FSx («еф-ес-екс») — сімейство керованих файлових систем під особливі задачі:

- **FSx for Windows File Server** — спільні папки для **Windows**: протокол **SMB**, права NTFS, інтеграція з **Active Directory**, кілька AZ. Для бухгалтерії «Хмаринки» — саме воно.
- **FSx for Lustre** («ластер») — **болід Формули-1** для наукових розрахунків, машинного навчання, рендерингу: сотні гігабайтів за секунду, затримка менше мілісекунди, **уміє читати дані прямо з S3** і писати результати назад. Буває тимчасовою (scratch, дешево) чи постійною (persistent).
- **FSx for NetApp ONTAP** — **швейцарський ніж**: NFS, SMB і iSCSI одночасно, Linux, Windows і macOS, знімки, клони, стиснення. Для переїзду з обладнання NetApp.
- **FSx for OpenZFS** — для переїзду з файлових систем ZFS.



Яку файлову систему обрати: EFS для Linux, FSx for Windows, FSx for Lustre для HPC, ONTAP — «швейцарський ніж»

Що обрати — одна таблиця

Потреба	Вибір
Диск для одного сервера, бази, системи	EBS (зазвичай gp3)
Понад 80 000 IOPS на одному диску	io2 Block Express
Дешевий диск для великих послідовних даних	st1
Найвища швидкість, тимчасові дані	instance store
Спільні файли для Linux-серверів у різних AZ	EFS
Спільні файли для Windows з Active Directory	FSx for Windows File Server
HPC, ML, дані в S3, сотні GB/s	FSx for Lustre
NFS + SMB + iSCSI разом, переїзд з NetApp	FSx for NetApp ONTAP

🥐 Базу «Хмаринки» перенесли на gp3 з потрібними IOPS — гальмування зникло, а рахунок навіть зменшився. Кондитери обох цехів бачать одну папку EFS, а бухгалтерія — звичну мережеву папку FSx for Windows з тими самими паролями, що й на роботі.

📖 **Словничок**

EBS («і-бі-ес») — блочний диск для EC2; одна AZ; зазвичай один сервер.

Multi-Attach — один io1/io2 диск до кількох серверів в одній AZ.

gp3 / gp2 / io2 / io1 / st1 / sc1 — універсал / старий універсал / гоночний / старий гоночний / вантажівка HDD / холодний HDD.

Snapshot («снепшот») — інкрементний знімок диска в S3; з нього — новий диск у будь-якій AZ; можна копіювати в інший регіон.

DLM / AWS Backup — автоматизація знімків.

Elastic Volumes — змінити тип, розмір, IOPS без зупинки.

Instance store — найшвидший, але тимчасовий диск самого фізичного сервера.

EFS («і-еф-ес») — спільна NFS для Linux у кількох AZ; порт 2049; класи з lifecycle.

Mount target — точка підключення EFS у кожній AZ.

FSx for Windows / Lustre / NetApp ONTAP / OpenZFS — SMB з AD / HPC і S3 / NFS+SMB+iSCSI / переїзд з ZFS.

🧩 Міні-тест: перевір себе

? Базі на EC2 потрібно понад 100 000 стабільних IOPS на одному диску. Що обрати?

- **А** gp2
- **Б** gp3
- **В** io2 Block Express
- **Г** st1

Відповідь: В. gp3 — до 80 000 IOPS. Більше — io2 Block Express (до 256 000).

? Потрібен дешевий диск для великих логів, які читають послідовно. Що обрати?

- **А** io2
- **Б** st1
- **В** instance store
- **Г** gp3 з максимальними IOPS

Відповідь: Б. st1 — «вантажівка» HDD для великих послідовних даних. Системним він бути не може.

? Як зменшити витрати на диски gp2 без простою?

- **А** Перейти на gp3 через Elastic Volumes
- **Б** Перейти на io2
- **В** Видалити знімки й сподіватися
- **Г** Зупинити сервери

Відповідь: А. gp3 дешевший, а IOPS купуються окремо від розміру. Зміна — на ходу.

? Диск EBS треба перенести в іншу зону доступності. Як?

А Просто відчепити й підключити до сервера в іншій AZ

Б Зробити знімок і створити з нього новий диск в іншій AZ

В Змінити тип диска

Г Це неможливо

Відповідь: Б. EBS живе в одній AZ. Переносять через знімок.

? Потрібна спільна файлова система для сотень Linux-серверів у різних AZ. Що обрати?

- **А** EBS Multi-Attach
- **Б** Amazon EFS
- **В** FSx for Windows File Server
- **Г** Instance store

Відповідь: Б. EFS — спільна NFS-папка для Linux у кількох AZ, росте сама.

? Windows-застосункам потрібні спільні файли з входом через Active Directory. Що обрати?

- **А** EFS
- **Б** FSx for Windows File Server
- **В** S3
- **Г** FSx for Lustre

Відповідь: Б. EFS — лише для Linux (NFS). Для Windows (SMB + AD) — FSx for Windows.

? Наукові розрахунки на сотнях серверів мають читати дані з S3 з величезною швидкістю. Що обрати?

- **А** FSx for Lustre
- **Б** EFS One Zone
- **В** sc1
- **Г** Glacier

Відповідь: А. Lustre — «болід» для HPC і ML, інтегрований з S3.

? Сервери EC2 не можуть підключити EFS — з'єднання обривається за тайм-аутом. Що перевірити насамперед?

- **А** Тип диска EBS
- **Б** Чи пропускає Security Group точки підключення порт 2049 (NFS)
- **В** Чи увімкнено versioning
- **Г** Регіон сертифіката

Відповідь: Б. Mount target має дозволяти NFS на порт 2049 від серверів.

✅ **Головне**

EBS — диск в **одній AZ**, зазвичай для одного сервера (Multi-Attach — io1/io2). **gp3** — за замовчуванням і дешевший за gp2; **io2 Block Express** — понад 80 000 IOPS; **st1/sc1** — дешеві HDD, не системні.

Snapshots — інкрементні знімки в S3: новий диск у будь-якій AZ, копія в інший регіон; автоматизація — **DLM** чи **AWS Backup**; **Elastic Volumes** — зміни без зупинки.

Instance store — найшвидший, але зникає при stop.

EFS — спільна **NFS для Linux** у кількох AZ, росте сама, порт **2049**, класи з lifecycle.

FSx: **Windows** (SMB + AD), **Lustre** (HPC + S3), **NetApp ONTAP** (NFS + SMB + iSCSI), **OpenZFS**.

🚚 Переїзд у хмару і гібрид: Storage Gateway, DataSync, Snowball, DMS, MGN

👴 Простими словами

Переїзд у хмару — як переїзд квартири. Дрібниці везеш мережею (DataSync), гори коробок — вантажівкою (Snowball), а бази й сервери переносять спеціальні грузчики (DMS, MGN), поки ти працюєш.

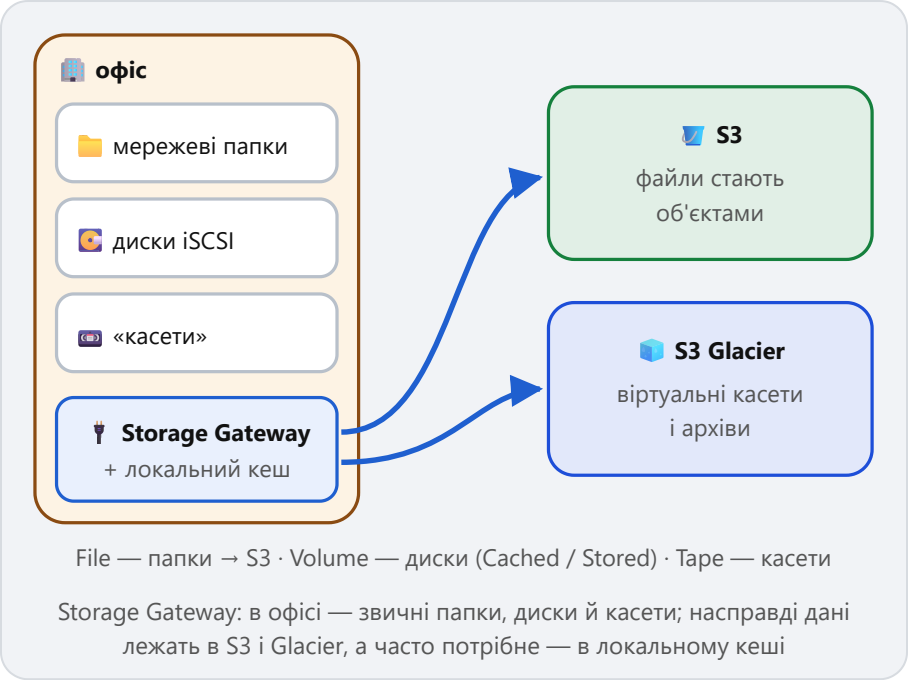
🎯 Що ти дізнаєшся

- як офіс «живе на два доми» зі Storage Gateway
- як порахувати, скільки днів займе перенесення даних мережею
- коли DataSync, коли Snowball, а коли Transfer Family
- як перенести базу даних без зупинки — DMS і SCT
- як переносять цілі сервери — MGN
- сім стратегій переїзду «7 R»

🥐 «Хмаринка» купила маленьку мережу пекарень у сусідньому місті — разом зі старою серверною. Там сховище NAS на 40 ТБ фото й документів, стара база Oracle, 30 старих серверів і полиця касет з резервними копіями. «Переїжджаємо в AWS, — вирішує Оля. — Але так, щоб пекарні не зупинялися ні на день».

Жити на два доми: Storage Gateway

Не все можна перенести одразу. Іноді програми в офісі мають і далі працювати зі звичними мережевими папками чи дисками, але дані мають лежати в хмарі. Для цього є **AWS Storage Gateway** («сторидж гейтвей») — **перехідник між офісом і хмарою**. Це віртуальна машина (чи пристрій) в офісі з **локальним кешем**: часто потрібне — під рукою, решта — в AWS.



- **S3 File Gateway** — звичні **мережеві папки** (NFS, SMB) в офісі, а кожен файл стає **об'єктом у S3** (далі — lifecycle, Glacier, усе, що ти знаєш про S3).
- **Volume Gateway** — **диски iSCSI** для офісних серверів:
  - **Cached** — основні дані в S3, а «гарячі» — в локальному кеші;
  - **Stored** — **усі** дані локально, а в AWS — асинхронні резервні копії (як знімки EBS).

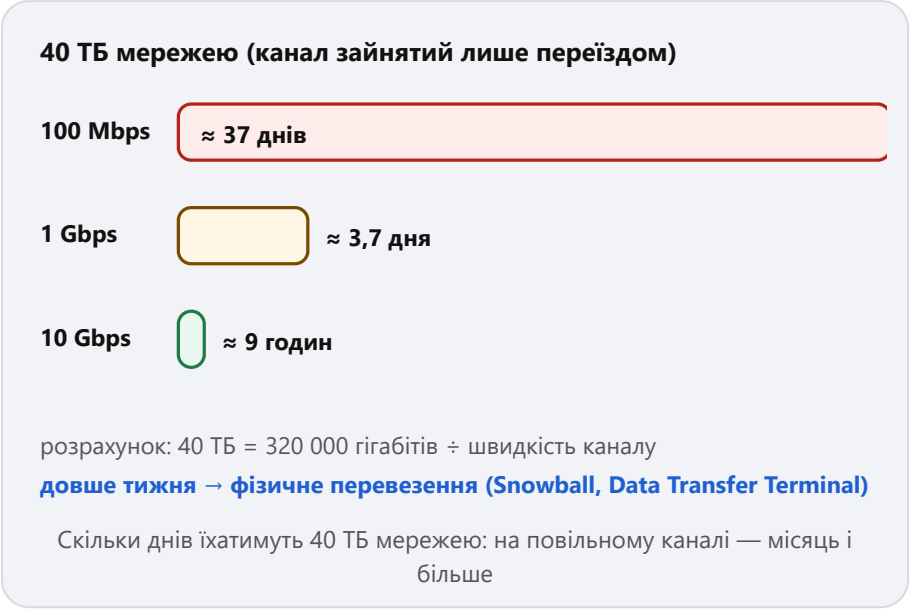
- **Tape Gateway** — **віртуальні касети**: наявні програми резервного копіювання «думають», що пишуть на стрічку, а насправді — в S3 і Glacier. Прощавайте, фізичні касети.

(FSx File Gateway — кеш для FSx for Windows — з жовтня 2024 недоступний новим клієнтам.)

Скільки їхатимуть 40 терабайтів?

Перш ніж обирати спосіб, треба порахувати. Пам'ятаєш правило «біти ділимо на 8»?

- 40 TB = 40 000 GB = 320 000 гігабітів.
- Канал **100 Mbps** (0,1 Gbps): 320 000 / 0,1 = 3,2 мільйона секунд ≈ **37 днів** — і це якщо канал віддати лише під переїзд.
- Канал **1 Gbps**: ≈ **3,7 дня**.



Скільки днів їхатимуть 40 ТБ мережею: на повільному каналі — місяць і більше

Орієнтир: якщо передача мережею займе **більше тижня** — розглядай **фізичне** перевезення.

DataSync — вантажівка по трасі

AWS DataSync («дата-синк») — **онлайн**-перенесення й синхронізація великих обсягів **мережею**:

- з офісних сховищ (NFS, SMB, HDFS, інші об'єктні сховища й хмари) — у **S3, EFS, FSx** (і між сервісами AWS);
- ставлять невеликого **агента** в офісі;
- шифрування, **перевірка цілісності**, збереження метаданих і прав доступу;

- **розклад** і обмеження швидкості, щоб не «з'їсти» весь канал у робочий час.

🎯 «Перенести 50 ТБ з NAS у S3 мережею, за розкладом, з перевіркою цілісності» → **DataSync**.

Snowball — контейнер поштою

Коли мережею надто довго чи дорого, дані везуть **фізично**. **AWS Snowball Edge** («сноубол едж») — захищений пристрій-валіза на десятки терабайтів: його привозять, ти записуєш дані, відправляєш назад — і AWS завантажує їх у S3. Образ з простого курсу — **контейнер, який везуть поштою, коли траса надто повільна**. Snowball уміє й обчислювати на місці, де немає нормального інтернету.

🆕 З 7 листопада 2025 року Snowball Edge **недоступний новим клієнтам**. AWS радить DataSync, **Data Transfer Terminal** (фізичні точки AWS, куди можна привезти диски й швидко завантажити) або партнерів. Але на іспиті Snowball ще може бути правильною відповіддю: «сотні терабайтів, повільний інтернет, дедлайн — тиждень».

Transfer Family — вікно для партнерів

Партнери «Хмаринки» надсилають накладні по **SFTP** — і не хочуть нічого змінювати. **AWS Transfer Family** — керований **SFTP / FTPS / FTP / AS2**-сервер, який кладе файли просто в **S3** чи EFS. Партнери працюють як раніше.

Бази даних: переїзд магазину без зачинення

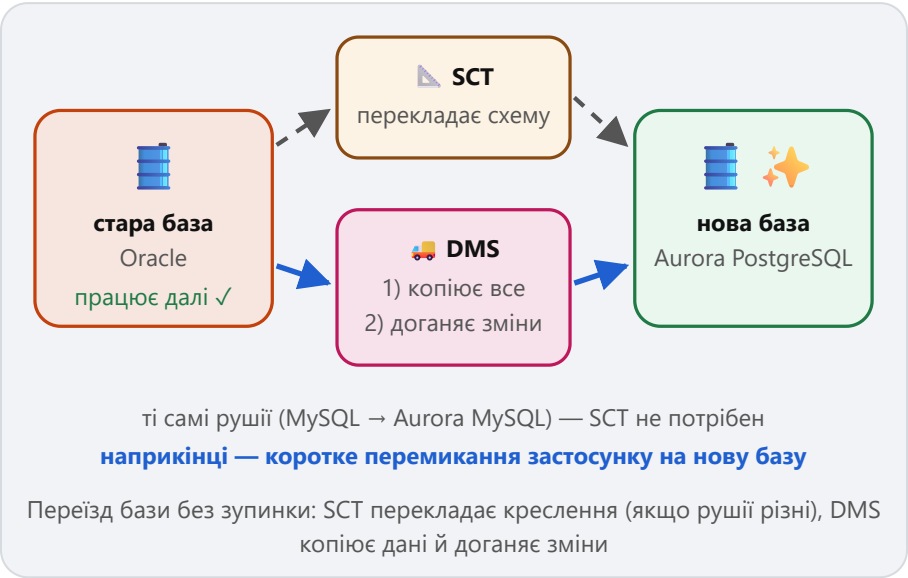
AWS DMS («ді-ем-ес», Database Migration Service) — переносить бази **з мінімальним простоєм**. Образ — **переїзд магазину без зачинення**: товари переносять, а продажі тривають.

1. DMS копіює всі наявні дані з бази-джерела в нову базу.
2. Поки копіює — стара база **працює** як завжди.
3. Далі DMS безперервно доганяє зміни — **CDC** (change data capture, «захоплення змін»).
4. Коли нова база «наздогнала» — застосунок коротко перемикають на неї.

Два випадки:

- **Однаковий рушій** (гомогенна міграція: MySQL → Aurora MySQL, Oracle → RDS Oracle) — вистачить DMS.

- **Різні рушії** (гетерогенна: Oracle → Aurora PostgreSQL, SQL Server → MySQL) — спершу **SCT** («ес-сі-ті», Schema Conversion Tool, або вбудований DMS Schema Conversion) — **перекладач креслень**: перетворює структуру таблиць і код збережених процедур під нову базу. Потім дані везе DMS.



DMS уміє й інші напрямки: у S3, Kinesis, DynamoDB, Redshift. А SCT перекладає ще й сховища даних (наприклад, Teradata → Redshift).

Сервери: перевезти будинок цілком

- **AWS Application Migration Service** (**MGN**) — «**перевезти будинок цілком**»: фізичні сервери, віртуалки VMware чи Hyper-V, сервери інших хмар **як є** (lift-and-shift) переносять у EC2. MGN безперервно копіює диски, а в день переїзду — коротке перемикання.
- **AWS Elastic Disaster Recovery** (**DRS**) — те саме безперервне копіювання, але як **страховка**: резервна копія офісних серверів в AWS з RPO в секунди й RTO у хвилини.
- **VMware Cloud on AWS** — перенести все середовище VMware без змін інструментів.

Сім стратегій переїзду: 7 R

Для кожної програми компанія вирішує, **що з нею робити**. Стратегій сім, усі починаються на R. Образ — **переїзд пекарні**:



Сім R: від «викинути стару піч» до «збудувати нову кухню»

1. **Retire** — **викинути**: програма більше не потрібна.
2. **Retain** — **залишити** поки як є (ще не час чи не можна).
3. **Rehost** — **перевезти як є** (lift-and-shift, MGN).
4. **Relocate** — **перевезти разом з фундаментом**: цілий гіпервізор (VMware Cloud on AWS).
5. **Replatform** — **перевезти й трохи підлатати** (lift, tinker and shift): наприклад, базу — на керовану RDS, застосунок — на Beanstalk. Код майже не змінюється.
6. **Repurchase** — **купити готове**: замість старої програми — SaaS.

7. **Refactor / Re-architect** — **перебудувати** під хмару: serverless, мікросервіси. Найдовше, але найбільше користі.

Що обрати — одна таблиця

Задача	Рішення
Офісні програми пишуть у мережеві папки, а дані — в S3, з локальним кешем	S3 File Gateway
Замінити фізичні касети, лишивши бекап-програми	Tape Gateway
Диски iSCSI: основне в хмарі, кеш локально	Volume Gateway (Cached)
Диски iSCSI: усе локально, копія в AWS	Volume Gateway (Stored)
Перенести десятки ТБ мережею, за розкладом, з перевіркою	DataSync
Сотні ТБ, повільний канал, тиждень на все	Snowball Edge (нові клієнти — Data Transfer Terminal, партнери)
Партнери шлють файли по SFTP, зберігати в S3	Transfer Family
Та сама база, мінімальний простій	DMS
Інша база (Oracle → PostgreSQL)	SCT + DMS
Сотні серверів «як є»	MGN
Аварійна копія офісних серверів в AWS	Elastic Disaster Recovery

🥐 План переїзду «Хмаринки» вийшов такий: 40 ТБ фото — DataSync вночі протягом двох тижнів; касети — Tape Gateway; Oracle → Aurora PostgreSQL — SCT + DMS, перемикання в неділю о 6 ранку за 10 хвилин; 20 серверів — MGN як є (rehost), базу — на RDS (replatform), стару бухгалтерську програму — на SaaS (repurchase), а 5 серверів, якими ніхто не користувався, просто вимкнули (retire).

📖 **Словничок**

Storage Gateway — перехідник офіс ↔ хмара з локальним кешем: File (S3), Volume (Cached / Stored), Tape.

DataSync («дата-синк») — онлайн-перенесення й синхронізація з перевіркою цілісності й розкладом.

Snowball Edge — фізичний пристрій для офлайн-перенесення (новим клієнтам недоступний з листопада 2025).

Data Transfer Terminal — фізичні точки AWS для швидкого завантаження даних.

Transfer Family — керований SFTP/FTPS/FTP/AS2 поверх S3 чи EFS.

DMS («ді-ем-ес») — міграція баз з мінімальним простоєм; **CDC** — безперервне догоняння змін.

SCT («ес-сі-ті») — перетворення схеми й коду бази під інший рушій.

Гомогенна / гетерогенна міграція — той самий рушій / різні рушії.

MGN — перенесення серверів як є (lift-and-shift).

Elastic Disaster Recovery (DRS) — аварійні копії серверів в AWS, RPO секунди.

7 R — Retire, Retain, Rehost, Relocate, Replatform, Repurchase, Refactor.

🧩 **Міні-тест: перевір себе**

? Офісні програми мають зберігати файли в S3 через звичні мережеві папки NFS/SMB, з локальним кешем. Що обрати?

- **А** S3 File Gateway
- **Б** DataSync
- **В** Transfer Family
- **Г** Snowball

Відповідь: А. File Gateway — «перехідник»: в офісі папки, у хмарі — об'єкти S3.

? Треба позбутися фізичних касет, але лишити наявні програми резервного копіювання. Що обрати?

А S3 File Gateway

Б Tape Gateway

В Volume Gateway Stored

Г EFS

Відповідь: Б. Tape Gateway — віртуальні касети для старих бекап-програм, а дані — в S3 і Glacier.

? Перенести 50 ТБ з офісного NAS у S3 мережею, за розкладом, з перевіркою цілісності. Що обрати?

- **А** Storage Gateway
- **Б** AWS DataSync
- **В** Transfer Family
- **Г** CloudFront

Відповідь: Б. DataSync — «вантажівка по трасі»: онлайн, з агентом, перевіркою й розкладом.

? Oracle → Aurora PostgreSQL з мінімальним простоєм. Що обрати?

- **А** Лише DMS
- **Б** SCT для схеми й коду + DMS для даних з CDC
- **В** Знімок і відновлення
- **Г** Snowball

Відповідь: Б. Різні рушії — спершу «перекладач креслень» SCT, потім DMS.

? MySQL в офісі → Aurora MySQL з мінімальним простоєм. Що обрати?

- **А** DMS (SCT не потрібен — рушій той самий)
- **Б** SCT + DMS
- **В** MGN
- **Г** Transfer Family

Відповідь: А. Гомогенна міграція: DMS з безперервною реплікацією змін.

? Сотні серверів треба швидко перенести в EC2 «як є». Що обрати?

- **А** Переписати все під Lambda
- **Б** AWS Application Migration Service (MGN)
- **В** DataSync
- **Г** Storage Gateway

Відповідь: Б. MGN — lift-and-shift цілих серверів.

? Партнери надсилають файли по SFTP і не хочуть нічого змінювати, а зберігати треба в S3. Що обрати?

А DataSync

Б AWS Transfer Family

В S3 File Gateway

Г Direct Connect

Відповідь: Б. Transfer Family — керований SFTP-сервер поверх S3.

? Під час переїзду базу переносять на керовану RDS, майже не змінюючи код застосунку. Яка це стратегія?

А Rehost

Б Replatform

В Refactor

Г Retire

Відповідь: Б. «Перевезти й трохи підлатати» — Replatform.

✅ **Головне**

Storage Gateway — життя на два доми з локальним кешем: **S3 File** (папки → S3), **Volume** (Cached — основне в хмарі, Stored — усе локально), **Tape** (віртуальні касети).

Рахуй час: 40 ТБ на 100 Mbps ≈ **37 днів**; довше тижня — фізично. **DataSync** — онлайн з перевіркою; **Snowball Edge** — офлайн (новим клієнтам — Data Transfer Terminal); **Transfer Family** — SFTP для партнерів.

DMS — база без зупинки (CDC); різні рушії → **SCT + DMS**.

MGN — сервери «як є»; **Elastic Disaster Recovery** — аварійні копії серверів; **VMware Cloud on AWS** — переїзд VMware.

7 R: Retire, Retain, **Rehost**, Relocate, **Replatform**, Repurchase, **Refactor**.

🟦 RDS: керована реляційна база

👴 Простими словами

RDS — база даних «під ключ»: AWS сам ставить, оновлює, робить копії й тримає запасну базу в іншому районі. Ти лише користуєшся.

🎯 Що ти дізнаєшся

- що означає «керована» база і що AWS робить за тебе
- чим Multi-AZ відрізняється від read replica — головна пара питань іспиту
- як працюють бекапи й відновлення «на будь-яку секунду»
- навіщо RDS Proxy
- як захистити базу і безпечно оновлювати її версію
- коли RDS — не найкращий вибір

🥐 Базу замовлень «Хмаринки» досі тримали на звичайному сервері EC2. Тарас ночами ставив патчі й вручну робив копії. А одного ранку диск сервера зламався — і пропали замовлення за цілий день. «Досить, — каже Оля. — Мені потрібен адміністратор бази, який не спить». — «Такий є, — відповідає Тарас. — Він називається RDS».

Що означає «керована» база

Базу даних можна тримати двома способами:

- **На власному сервері EC2** — ти власноруч встановлюєш програму бази, ставиш патчі ОС і бази, налаштовуєш копії, стежиш за дисками, організовуєш запасний сервер. Повний контроль — і повна відповідальність.
- **У керованому сервісі** — **Amazon RDS** («ар-ді-ес», Relational Database Service). Образ з простого курсу — **найнятий адміністратор бази даних**: він сам встановлює, латає, робить бекапи, стежить і в разі аварії перемикає на запасну копію.

Що лишається тобі: структура таблиць і запити, **доступи**, налаштування, політика бекапів, вибір шифрування.

Пам'ятаєш модель спільної відповідальності? На RDS **ОС і рушій бази латає AWS**. Доступу до самої операційної системи немає (виняток — **RDS Custom** для Oracle і SQL Server, коли він справді потрібен).

Рушії RDS: **MySQL**, **PostgreSQL**, **MariaDB**, **Oracle**, **Microsoft SQL Server**, **Db2**. (Є ще Aurora — про неї наступний розділ.)

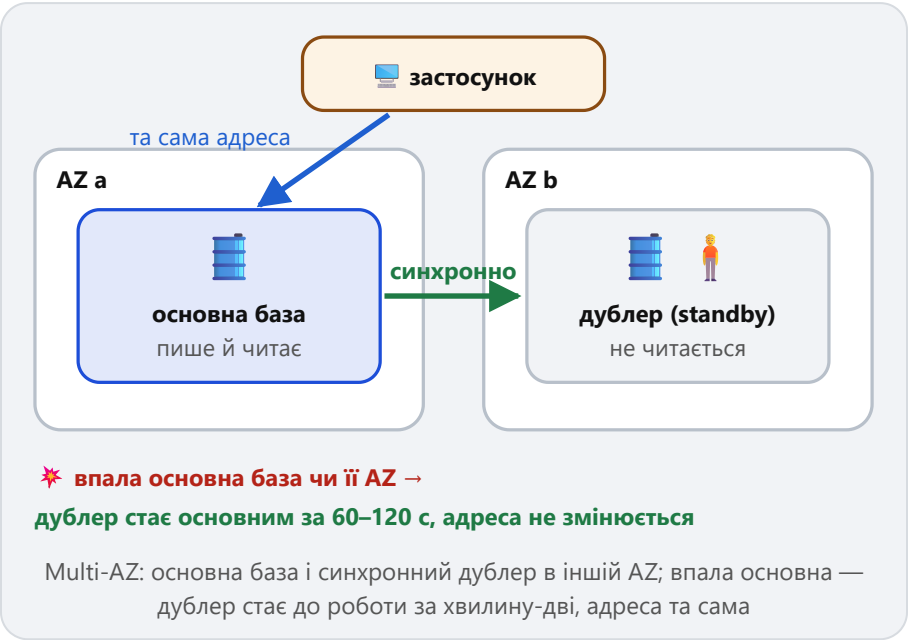
При створенні обираєш **клас інстансу** (як тип сервера EC2: скільки процесора й пам'яті) і **сховище** (gp3 чи io2 — ти вже знаєш ці диски). **Storage autoscaling** — диск бази росте сам, коли закінчується місце.

Multi-AZ — дублер у запасній кімнаті

Multi-AZ («мульті-ей-зі») — **синхронна запасна копія** бази в **іншій** AZ.

- Кожен запис потрапляє **одночасно** в основну базу і в запасну — синхронно (пам'ятаєш синхронну реплікацію з розділу про бази з нуля?).
- Основна база «знепритомніла» чи зламалася її AZ — **автоматичне перемикання** (failover) на запасну, зазвичай за **60–120 секунд**.
- **Адреса підключення не змінюється** — застосунок продовжує працювати з тим самим ім'ям.

Образ з простого курсу — **дублер у запасній кімнаті**: синхронно записує все, але **з клієнтами не говорить**. Звичайний standby у Multi-AZ **не обслуговує читання** — він існує лише для **виживання**.



Multi-AZ: основна база і синхронний дублер в іншій AZ; впала основна — дублер стає до роботи за хвилину-дві, адреса та сама

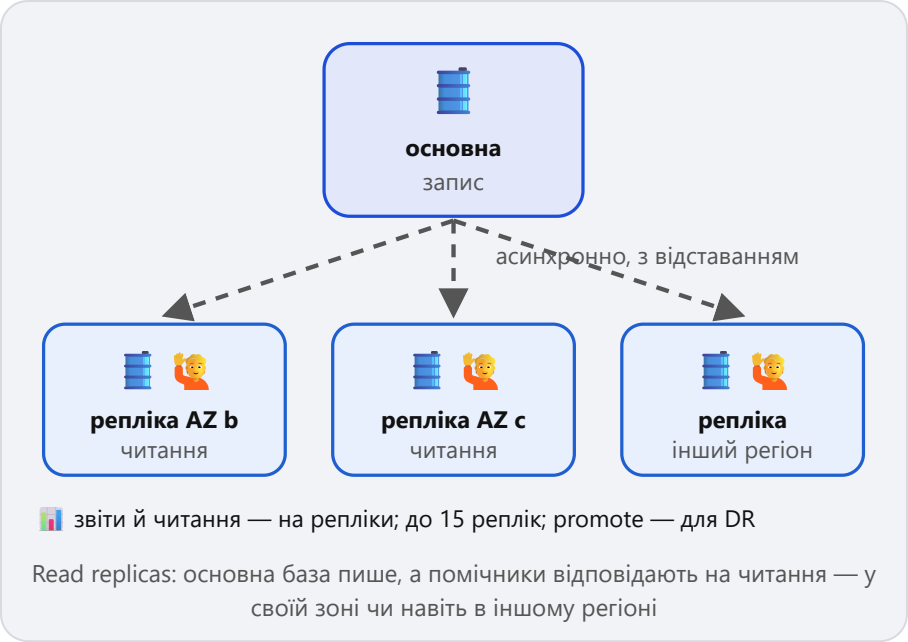
Є ще посилений варіант — **Multi-AZ DB cluster** (для MySQL і PostgreSQL): одна база для запису і **дві** запасні в різних AZ, **з яких можна читати**; перемикання — до ~35 секунд.

Read replicas — помічники, що відповідають на запитання

Read replicas («рід репліказ», репліки для читання) — **асинхронні** копії бази, з яких можна **читати**:

- до **15** реплік (для MySQL, MariaDB, PostgreSQL);
- в інших AZ і навіть в **інших регіонах**;
- копії можуть трохи **відставати** (асинхронно — ти знаєш, що це означає);
- застосунок **сам** надсилає запити на читання на адресу репліки;
- репліку можна **підвищити** (promote) до самостійної бази — так роблять недорогий аварійний план в іншому регіоні;
- автоматично додавати репліки RDS **не вміє** (Aurora — вміє).

Образ з простого курсу — **помічники, які лише відповідають на запитання** (читання) і можуть трохи відставати від останніх новин.



Read replicas: основна база пише, а помічники відповідають на читання — у своїй зоні чи навіть в іншому регіоні

Головна пара іспиту: Multi-AZ чи read replica

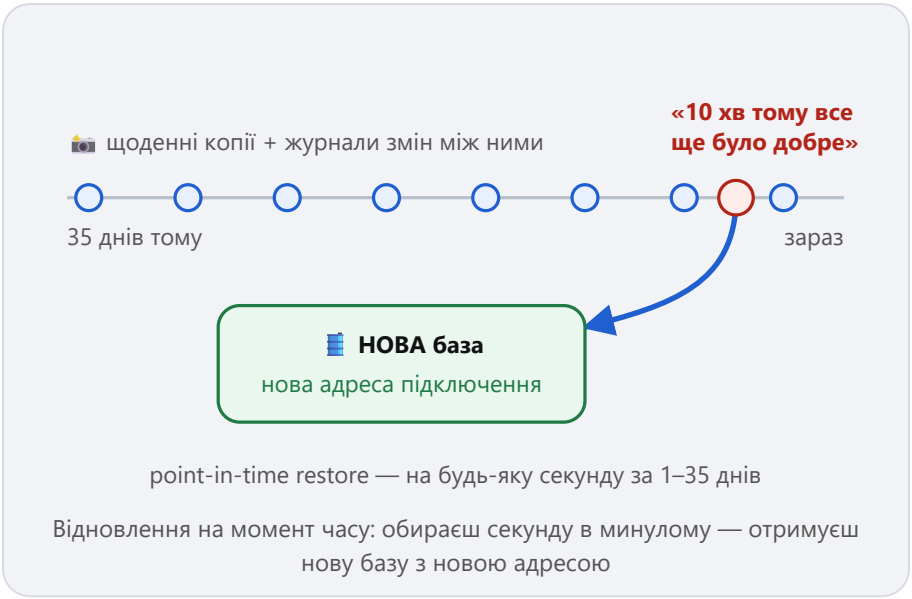
	Multi-AZ	Read replica
Навіщо	**виживання** (висока доступність)	**швидкість читання** (масштабування)
Реплікація	синхронна	асинхронна (можливе відставання)
Читати з копії	ні (у звичайному Multi-AZ)	так
Перемикання	автоматичне, та сама адреса	вручну: promote, нова адреса
Де	інша AZ того самого регіону	будь-яка AZ, інший регіон

Мнемоніка з простого курсу: **Multi-AZ — для виживання. Read Replica — для читання.**

🎯 «Звіти аналітиків гальмують основну базу» → **read replica**. «Автоматичне перемикання бази в іншу AZ» → **Multi-AZ**. «Недорогий DR в іншому регіоні» → **cross-Region read replica** (при аварії — promote).

Бекапи і відновлення на будь-яку секунду

- **Автоматичні бекапи** — щодня плюс журнали змін; зберігаються **1–35 днів**. Завдяки журналам можна відновити базу **на будь-яку секунду** цього періоду (приблизно до останніх 5 хвилин) — **point-in-time restore** (PITR, «відновлення на момент часу»).
- **Ручні знімки** (snapshots) — живуть, **доки не видалиш**, навіть після видалення бази.



⚠️ Відновлення зі знімка чи на момент часу **завжди створює нову базу з новою адресою**. Застосунок треба переналаштувати на неї.

Для тестових баз корисна дрібниця: базу RDS можна **зупинити до 7 днів** (потім вона запуститься сама) — і не платити за процесор, поки ніхто не працює.

RDS Proxy — секретар на телефоні

Уяви тисячу функцій Lambda, кожна з яких відкриває з'єднання з базою на частку секунди. Базі доводиться весь час відкривати й закривати тисячі «телефонних ліній» — і вона задихається.

Amazon RDS Proxy — секретар, що тримає телефонні лінії: тримає **пул** відкритих з'єднань і роздає їх клієнтам.

- витримує тисячі коротких з'єднань (типово — від Lambda);
- перемикання при аварії стає швидшим (до 66%);
- уміє входити через IAM і брати паролі з Secrets Manager;
- код застосунку **не змінюється** — лише адреса бази.

Безпека бази

- База живе в **приватній** підмережі; її Security Group пускає **лише** сервери застосунку на порт бази (3306 для MySQL, 5432 для PostgreSQL). Пам'ятаєш ланцюжок довіри?
- **Шифрування** в спокої вмикається **лише при створенні** (існуючу — через зашифровану копію знімка); у дорозі — TLS. Репліки шифруються так само, як основна база.
- **IAM database authentication** (для MySQL і PostgreSQL) — вхід за тимчасовим токеном замість пароля.
- Паролі — у **Secrets Manager** з автоматичною ротацією.

Оновлення й спостереження

- **Вікна обслуговування** (maintenance window) — коли AWS можна ставити патчі.
- **RDS Blue/Green Deployments** — безпечне оновлення версії бази: поруч створюється копія з новою версією, а перемикання займає менше хвилини, з можливістю відкату.
- **Enhanced Monitoring** — метрики **операційної системи** бази (процеси, пам'ять) з інтервалом до секунди.
- **Performance Insights** — які **SQL-запити** найбільше навантажують базу.

Коли RDS — не найкращий вибір

- Потрібен доступ до ОС чи особливі агенти → **RDS Custom** або база на **EC2**.
- Мільйони простих запитів за ключем, serverless → **DynamoDB** (через розділ).
- Важка аналітика по всій історії → сховище даних **Redshift** (частина про аналітику).
- Максимальна надійність і швидкість реляційної бази → **Aurora** (наступний розділ).

🥐 Базу замовлень «Хмаринки» перенесли на RDS PostgreSQL з Multi-AZ: коли через місяць AWS проводив обслуговування в одній зоні, база сама перемкнулася за півтори хвилини — а сайт навіть не змінив адресу. Звіти бухгалтерії тепер читають з read replica, і основна база їх не помічає.

📖 **Словничок**

RDS («ар-ді-ес») — керовані реляційні бази: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, Db2.

RDS Custom — RDS з доступом до ОС (Oracle, SQL Server).

Storage autoscaling — диск бази росте сам.

Multi-AZ — синхронна запасна копія в іншій AZ, автоматичне перемикання за 60–120 с, та сама адреса; standby не читається.

Multi-AZ DB cluster — 1 база для запису + 2 запасні, з яких можна читати.

Read replica — асинхронна копія для читання (до 15, інші регіони); promote — зробити самостійною.

Automated backups / PITR — щоденні копії й журнали на 1–35 днів / відновлення на будь-яку секунду.

Snapshot — ручний знімок бази, живе, доки не видалиш.

RDS Proxy — пул з'єднань, швидше перемикання, IAM і Secrets Manager.

IAM database authentication — вхід у базу за тимчасовим токеном.

Blue/Green Deployments — безпечне оновлення версії з перемиканням менш ніж за хвилину.

Enhanced Monitoring / Performance Insights — метрики ОС бази / найважчі SQL-запити.

🧩 **Міні-тест: перевір себе**

? Що дає RDS Multi-AZ?

- **А** Синхронну запасну копію в іншій AZ з автоматичним перемиканням і тією самою адресою
- **Б** Швидше читання для звітів
- **В** Копію бази в іншому регіоні для аналітиків
- **Г** Безкоштовну базу

Відповідь: А. Multi-AZ — для виживання: дублер пише синхронно і підхоплює роботу при аварії.

? Чи можна читати дані зі standby у звичайному RDS Multi-AZ?

- **А** Так, завжди
- **Б** Ні, standby існує лише для перемикання при аварії
- **В** Лише вночі
- **Г** Лише з іншого регіону

Відповідь: Б. Для читання — read replicas (або Multi-AZ DB cluster з читабельними standby).

? Звіти аналітиків гальмують основну базу. Що допоможе?

- **А** Multi-AZ
- **Б** Read replica, на яку спрямувати звіти
- **В** Зупинити базу на ніч
- **Г** Збільшити тайм-аут

Відповідь: Б. Репліки для читання забирають на себе запити на читання.

? Потрібно відновити базу на стан 10 хвилин тому. Що зробити — і що буде з адресою?

- **А** Point-in-time restore; він створить нову базу з новою адресою
- **Б** Point-in-time restore; адреса лишиться старою
- **В** Перезавантажити базу
- **Г** Перемкнути Multi-AZ

Відповідь: А. Відновлення завжди створює нову базу — застосунок переналаштовують на нову адресу.

? Тисячі функцій Lambda вичерпують з'єднання до бази RDS. Що допоможе з мінімальними змінами коду?

- **А** Більший клас інстансу
- **Б** RDS Proxy
- **В** Read replica
- **Г** Multi-AZ

Відповідь: Б. Proxy тримає пул з'єднань і роздає їх функціям; у коді міняється лише адреса.

? Потрібен недорогий план аварійного відновлення бази в іншому регіоні. Що обрати?

- **А** Multi-AZ
- **Б** Cross-Region read replica, яку при аварії підвищують (promote)
- **В** Два знімки на день у тому самому регіоні
- **Г** RDS Proxy

Відповідь: Б. Репліка в іншому регіоні — дешевий DR. Але вона асинхронна, тож останні секунди можуть не встигнути.

? Які саме SQL-запити найбільше навантажують базу? Де подивитися?

- **А** Enhanced Monitoring
- **Б** Performance Insights
- **В** CloudTrail
- **Г** VPC Flow Logs

Відповідь: Б. Performance Insights показує найважчі запити; Enhanced Monitoring — метрики ОС.

? Для бази Oracle потрібен доступ до операційної системи, щоб поставити спеціальний агент. Що обрати?

- **А** RDS Custom (або базу на EC2)
- **Б** Звичайний RDS for Oracle
- **В** DynamoDB
- **Г** Aurora Serverless

Відповідь: А. Звичайний RDS доступу до ОС не дає. RDS Custom — дає.

✅ **Головне**

RDS — найнятий адміністратор бази: патчі ОС і рушія, бекапи, перемикання — на AWS; тобі — схема, доступи, налаштування. Доступ до ОС — лише **RDS Custom**.

Multi-AZ — синхронний дублер в іншій AZ, автоматичне перемикання **60–120 с**, та сама адреса, **standby не читається**. Multi-AZ DB cluster — 2 читабельні standby.

Read replicas — асинхронні, до **15**, інші регіони, promote для DR; застосунок сам шле на них читання. **Multi-AZ — для виживання, Read Replica — для читання.**

Бекапи **1–35 днів** + **PITR** на будь-яку секунду; відновлення = **нова база з новою адресою**; тестову базу можна зупинити до 7 днів.

RDS Proxy — пул з'єднань для Lambda; шифрування — лише при створенні; **IAM DB auth**; **Blue/Green** для оновлень; **Performance Insights** — важкі запити.

🚀 Aurora: база, створена для хмари

👴 Простими словами

Aurora — та сама звична база, але перебудована спеціально для хмари: швидша, сама ділить дані на шість копій і швидко перемикається, якщо щось зламалося.

🎯 Що ти дізнаєшся

- чим Aurora відрізняється від звичайного RDS — окреме розумне сховище на 6 копій
- як працюють репліки на спільному сховищі і чотири види адрес
- Global Database — база для кількох регіонів
- Aurora Serverless v2, що «засинає» до нуля
- Backtrack, Cloning, Babelfish та інші особливі вміння

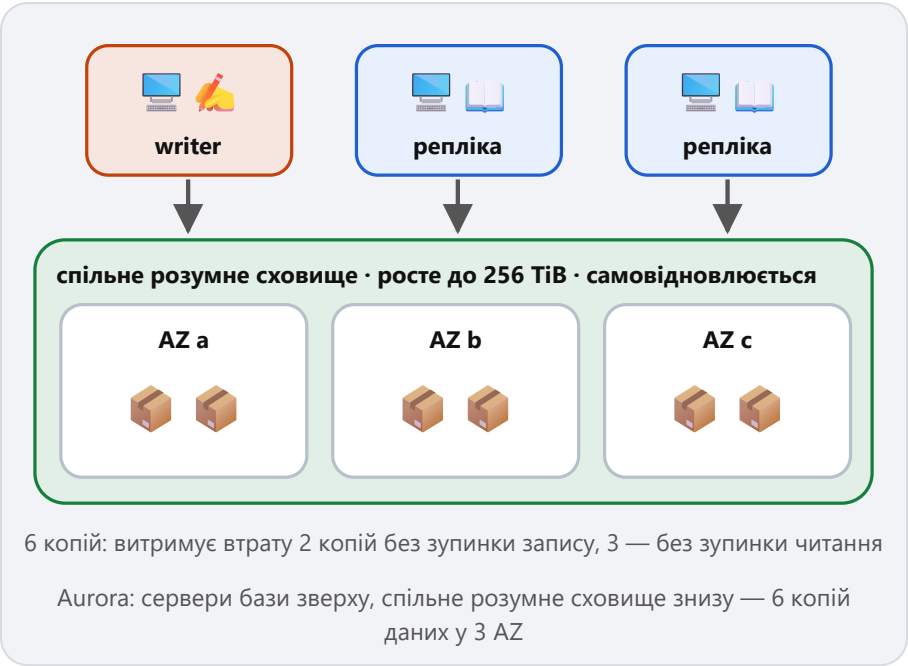
🥐 Мережа «Хмаринка» росте: тепер замовляють і з Канади, де сестра Олі відкриває точку. Канадським клієнтам потрібні швидкі відповіді, вночі база майже не працює, а нещодавно розробник випадково видалив половину таблиці знижок — і відновлення з бекапу зайняло пів години. «Хочу базу, яка вміє все це сама», — каже Оля. — «Тоді тобі до Aurora», — усміхається Тарас.

Що таке Aurora

Amazon Aurora («аврора») — власна реляційна база AWS, **сумісна з MySQL і PostgreSQL**: програми, що вміють працювати з ними, працюють і з Aurora майже без змін. За даними AWS, вона до 5 разів швидша за звичайний MySQL і до 3 разів — за PostgreSQL. Образ з простого курсу — **«турбо-версія» MySQL і PostgreSQL**.

Головна відмінність від звичайного RDS — будова. У RDS кожен сервер бази має **свій** диск. В Aurora **сервери** (обчислення) і **сховище** розділені: усі сервери бази працюють з **одним спільним розумним сховищем**.

Сховище на шість копій



6 копій: витримує втрату 2 копій без зупинки запису, 3 — без зупинки читання

Aurora: сервери бази зверху, спільне розумне сховище знизу — 6 копій даних у 3 AZ

- Дані зберігаються в **6 копіях у 3 AZ** — по дві в кожній.
- Сховище **росте само** — до **256 TiB** (з 2025 року; раніше 128 TiB).
- Воно **самовідновлюється**: зіпсована копія автоматично відновлюється з інших.
- Навіть якщо зникнуть **дві** копії, запис продовжується, а якщо **три** — читання продовжується.

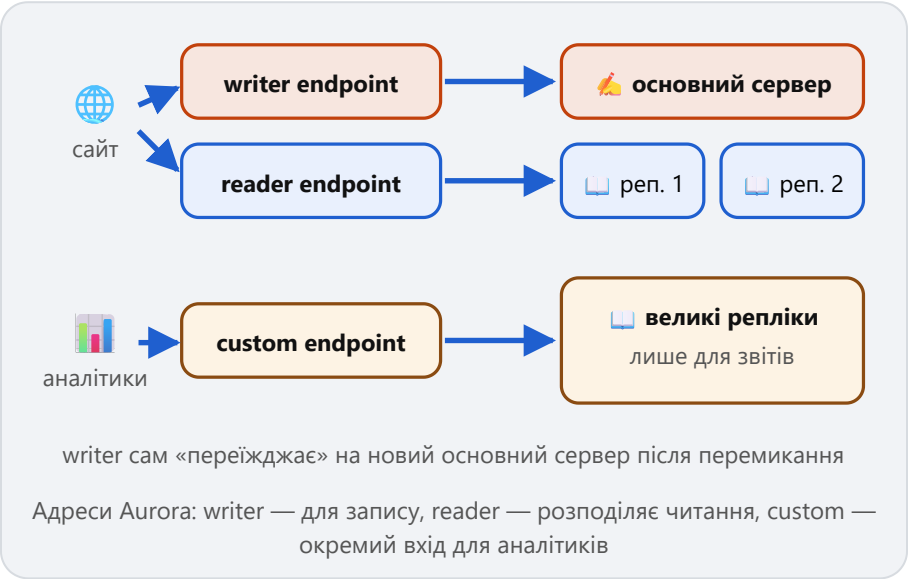
Мнемоніка з простого курсу: **Aurora — «6 копій у 3 AZ»**.

Репліки на спільному сховищі

Оскільки сховище спільне, репліці не треба копіювати дані — вона **читає те саме сховище**. Тому:

- реплік може бути до **15**, а їхнє відставання зазвичай **менше 100 мілісекунд**;
- при аварії основного сервера одна з реплік стає основною **приблизно за 30 секунд** (порядок задають «рівні пріоритету» — failover tiers);
- **Aurora Auto Scaling** сама додає й прибирає репліки за навантаженням.

Чотири адреси Aurora



writer сам «переїжджає» на новий основний сервер після перемикання

Адреси Aurora: writer — для запису, reader — розподіляє читання, custom — окремий вхід для аналітиків

- **Writer (cluster) endpoint** — адреса для **запису**. Завжди веде на поточний основний сервер, навіть після перемикання.
- **Reader endpoint** — адреса для **читання**: сама розподіляє запити між репліками.
- **Custom endpoints** — **власні** адреси для груп реплік: наприклад, «великі репліки — лише для звітів аналітиків», щоб вони не заважали сайту.
- **Instance endpoint** — адреса конкретного сервера (рідко потрібна).

⚠️ Застосунок має **писати** через writer endpoint, а **читати** — через reader чи custom.

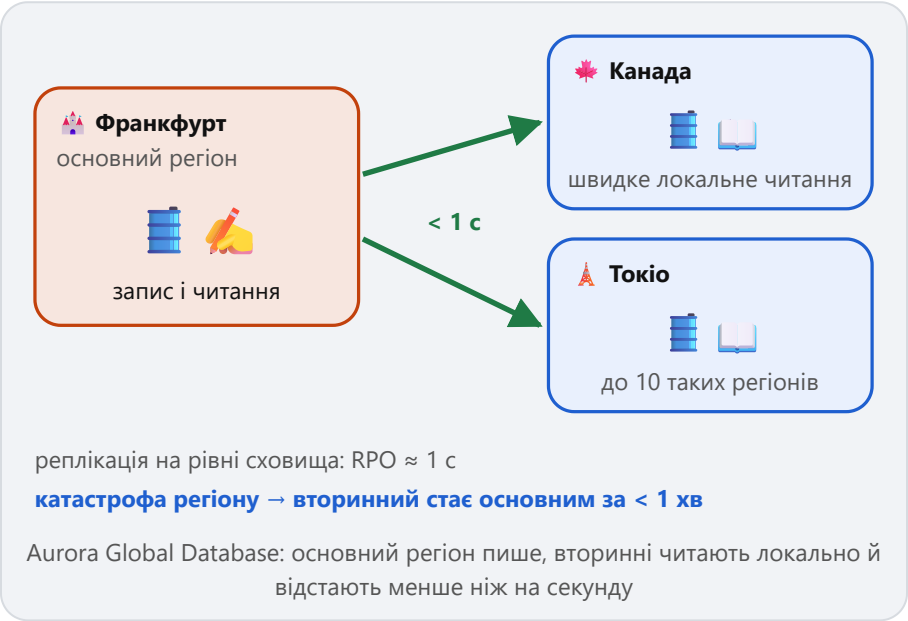
Global Database — одна база на кілька регіонів

Aurora Global Database — основна база в одному регіоні і **до 10 вторинних регіонів** (з травня 2025; раніше — 5):

- реплікація відбувається **на рівні сховища** — дуже швидко, відставання зазвичай **менше секунди**;
- у кожному регіоні — **швидке локальне читання** для місцевих клієнтів (Канада читає з Канади);

- при катастрофі регіону вторинний стає основним **менш ніж за хвилину**: **RPO ≈ 1 секунда, RTO < 1 хвилини**;
- можна навіть надсилати записи з вторинного регіону — вони переадресовуються в основний (write forwarding).

Для порівняння: звичайна **cross-Region read replica** Aurora MySQL реплікується «логічно» (через журнал змін), з відставанням від секунд до хвилин.



Aurora Global Database: основний регіон пише, вторинні читають локально й відстають менше ніж на секунду

🎯 «Реляційна база має пережити падіння регіону з RPO близько секунди й RTO менше хвилини» → **Aurora Global Database**.

Aurora Serverless v2 — авто з автоматичною коробкою

Звичайна база — як авто з механічною коробкою: обираєш потужність (клас інстансу) заздалегідь. **Aurora Serverless v2** — **автоматична коробка**: потужність змінюється **сама й миттєво** під навантаження.

- Потужність міряють в **ACU** (Aurora capacity unit): **1 ACU ≈ 2 GiB пам'яті** плюс процесор і мережа.
- З кінця 2024 року база може опускатися **до 0 ACU** — «**засинати**», коли з нею ніхто не працює. Тоді платиш лише за сховище, а пробудження займає до ~15 секунд.
- Ідеально для **непередбачуваного** чи **переривчастого** навантаження і для тестових баз.
- В одному кластері можна поєднувати звичайні (provisioned) і serverless-сервери.

Особливі вміння Aurora

- **Backtrack** («бектрек») — **перемотка** кластера назад у часі (до 72 годин) **без відновлення з бекапу**, за хвилини. Лише **Aurora MySQL**. Для PostgreSQL — відновлення на момент часу або клон.
- **Cloning** — **миттєва копія** бази для тестів: нова база ділить сховище з оригіналом і зберігає лише свої зміни (copy-on-write). Робиться за хвилини навіть для терабайтів.
- **Babelfish** («бебелфіш») — Aurora PostgreSQL **розуміє мову SQL Server** (T-SQL): застосунки з Microsoft SQL Server переходять майже без змін коду.
- **Aurora Machine Learning** — прогнози машинного навчання **прямо в SQL-запитах** (виклики SageMaker AI, Comprehend чи Bedrock) без знань ML.
- **I/O-Optimized** — режим без окремої плати за операції вводу-виводу: вигідний, коли вони складають понад ~25% витрат на Aurora.
- **Zero-ETL з Redshift** — дані з Aurora майже одразу доступні в сховищі даних Redshift, без окремих конвеєрів.
- **Aurora DSQL** (з 2025) — розподілена serverless-база, сумісна з PostgreSQL, яка працює «активно-активно» в кількох регіонах.

RDS чи Aurora?

	RDS	Aurora
Рушії	6 різних	MySQL і PostgreSQL (сумісні)
Сховище	окремий диск у кожного сервера	спільне: 6 копій у 3 AZ, до 256 TiB
Репліки	до 15, асинхронні, власні копії	до 15, на спільному сховищі, відставання < 100 мс
Перемикання	60–120 с (Multi-AZ)	~30 с
Автододавання реплік	ні	так (Auto Scaling)
Кілька регіонів	cross-Region replica	Global Database (< 1 с, до 10 регіонів)
Serverless	ні	Serverless v2 (до 0 ACU)

🎯 «Максимальна доступність і продуктивність реляційної бази з мінімумом адміністрування» → **Aurora**.

🥐 Базу мережі «Хмаринка» перевели на Aurora PostgreSQL з Global Database: Канада читає з канадського регіону. Тестова база працює на Serverless v2 і вночі засинає до нуля. А для тестів нової системи знижок розробники щоранку роблять клон робочої бази за дві хвилини — і більше ніхто не тренується на справжніх даних.

📖 **Словничок**

Aurora («аврора») — реляційна база AWS, сумісна з MySQL і PostgreSQL, з окремим спільним сховищем.

6 копій у 3 AZ — будова сховища Aurora; росте до 256 TiB, самовідновлюється.

Aurora Replica — репліка на спільному сховищі (до 15, відставання < 100 мс).

Failover tiers — порядок, у якому репліки стають основними.

Writer / reader / custom / instance endpoint — запис / розподілене читання / власні групи реплік / конкретний сервер.

Global Database — до 10 вторинних регіонів, реплікація сховища < 1 с, RPO ≈ 1 с, RTO < 1 хв.

Serverless v2 / ACU — автоматична потужність / одиниця ≈ 2 GiB пам'яті; може «засинати» до 0 ACU.

Backtrack — перемотка назад до 72 год без відновлення (лише Aurora MySQL).

Cloning — миттєва копія бази (copy-on-write).

Babelfish — Aurora PostgreSQL розуміє T-SQL (SQL Server).

I/O-Optimized / Zero-ETL / DSQL — без плати за I/O / дані одразу в Redshift / розподілена багаторегіональна база.

🧩 Міні-тест: перевір себе

? Скільки копій даних тримає сховище Aurora і в скількох AZ?

- **А** 2 копії в 2 AZ
- **Б** 3 копії в 1 AZ
- **В** 6 копій у 3 AZ
- **Г** 15 копій у 15 регіонах

Відповідь: В. «6 копій у 3 AZ» — по дві в кожній зоні; сховище самовідновлюється.

? Запити аналітиків і вебтрафік треба розвести на різні групи реплік. Що використати?

- **А** Writer endpoint
- **Б** Reader endpoint для всіх
- **В** Custom endpoints
- **Г** Instance endpoint основного сервера

Відповідь: В. Custom endpoint — окремий вхід для обраних реплік, наприклад потужніших для звітів.

? Реляційна база має пережити падіння регіону з RPO ≈ 1 с і RTO < 1 хв. Що обрати?

- **А** RDS Multi-AZ
- **Б** Щоденні знімки в інший регіон
- **В** Aurora Global Database
- **Г** Aurora Serverless v2

Відповідь: В. Global Database реплікує на рівні сховища з відставанням менше секунди.

? База простоює вночі й на вихідних, а вдень навантаження непередбачуване. Що обрати?

- **А** Aurora Serverless v2
- **Б** Найбільший клас інстансу назавжди
- **В** Read replica
- **Г** DynamoDB Standard-IA

Відповідь: А. Serverless v2 сама змінює потужність і може «засинати» до 0 ACU.

? Розробник випадково видалив дані годину тому. Потрібно швидко повернути стан без відновлення з бекапу. База — Aurora MySQL. Що обрати?

А Point-in-time restore у нову базу

Б Aurora Backtrack

В Read replica

Г Babelfish

Відповідь: Б. Backtrack «перемотує» кластер назад за хвилини. Працює лише в Aurora MySQL.

? Для тестів треба швидко отримати копію робочої бази на кілька терабайтів. Що обрати?

- **А** Повний знімок і відновлення
- **Б** Aurora Cloning
- **В** Global Database
- **Г** Multi-AZ

Відповідь: Б. Клон ділить сховище з оригіналом і зберігає лише власні зміни — тому з'являється за хвилини.

? Компанія переїжджає з SQL Server і не хоче переписувати код на T-SQL. Що обрати?

- **А** Aurora MySQL
- **Б** Babelfish for Aurora PostgreSQL
- **В** DynamoDB
- **Г** RDS for MariaDB

Відповідь: Б. Babelfish дає Aurora PostgreSQL розуміти мову й протокол SQL Server.

✅ **Головне**

Aurora — «турбо» MySQL/PostgreSQL з окремим спільним сховищем: **6 копій у 3 AZ**, до **256 TiB**, самовідновлення.

До **15 реплік** на спільному сховищі (< 100 мс), перемикання **~30 с**, **Auto Scaling** реплік.

Адреси: **writer** (запис), **reader** (розподілене читання), **custom** (окремі групи), instance.

Global Database — до **10** регіонів, < 1 с, **RPO ≈ 1 с, RTO < 1 хв**, локальне читання.

Serverless v2 (ACU, до **0**), **Backtrack** (лише MySQL, до 72 год), **Cloning**, **Babelfish** (T-SQL), **Aurora ML**, **I/O-Optimized**, **Zero-ETL** з Redshift.

🗃️ DynamoDB: база без серверів на мільйони запитів

👴 Простими словами

DynamoDB — величезна картотека, де кожну картку знаходять миттєво за номером. Серверів не видно, а витримує вона мільйони запитів на секунду.

🎯 Що ти дізнаєшся

- що таке DynamoDB і чому вона не «ламається» під навантаженням
- таблиці, елементи й ключі — partition key і sort key на прикладі кошиків
- що таке «гарячий» ключ і як його уникнути
- режими оплати й розрахунок RCU/WCU — з прикладом
- DAX, Global Tables, Streams, TTL і бекапи
- коли DynamoDB — не найкращий вибір

🥐 У Чорну п'ятницю мобільний застосунок «Хмаринки» мав мільйони операцій з кошиками: додати круасан, прибрати, змінити кількість. Реляційна база ледве встигала. «Кошику не потрібні складні зв'язки й звіти, — пояснює Тарас. — Йому потрібно одне: миттєво знайти кошик за номером клієнта, навіть коли клієнтів мільйон. Це робота для DynamoDB».

Що таке DynamoDB

Amazon DynamoDB («дайнамо-ді-бі») — **serverless NoSQL**-база (ключ-значення й документи):

- відповідає за **одноцифрові мілісекунди** на будь-якому масштабі — хоч сто запитів за секунду, хоч мільйони;
- серверів **немає**: нічого не треба встановлювати, латати й масштабувати;
- дані **автоматично** зберігаються в кількох AZ.

Образ з простого курсу — **гігантська картотека**: знаєш номер картки (ключ) — отримуєш її за мілісекунди, навіть коли запитів мільйони.

Таблиці, елементи, атрибути

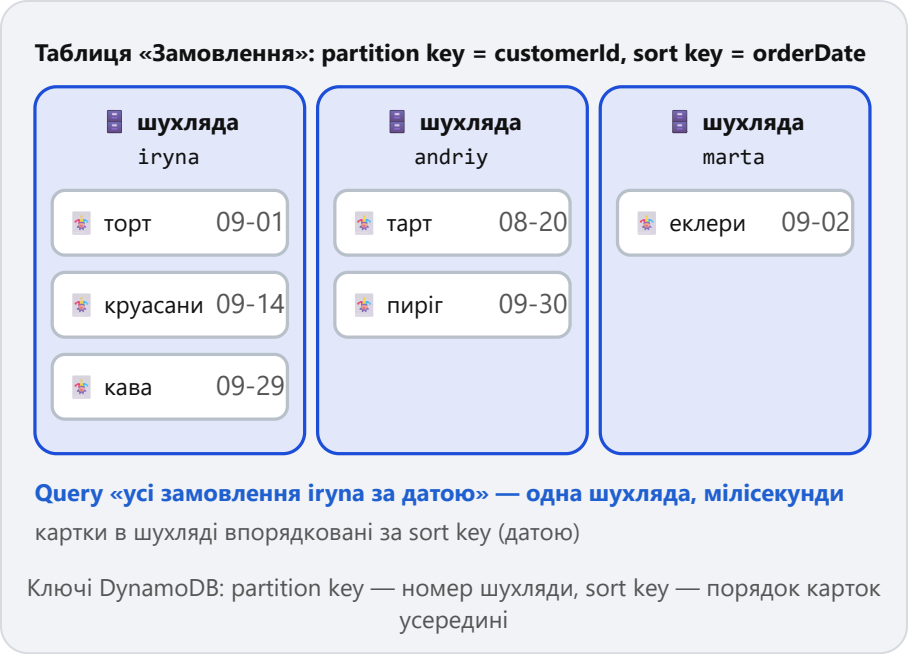
- **Таблиця** — ящик картотеки: «Кошики», «Замовлення».
- **Елемент** (item) — одна **картка**. Розмір — до **400 КВ**.
- **Атрибути** — поля на картці. Схема **гнучка**: у кожної картки можуть бути свої поля, обов'язковий лише **ключ**.

> 💡 Великі файли (фото, PDF) в DynamoDB не кладуть: їх зберігають у S3, а в елементі — лише посилання.

Ключі: номер шухляди й порядок усередині

Кожен елемент має **первинний ключ** (primary key). Буває двох видів:

- лише **partition key** («партишн кі», ключ розділу) — **номер шухляди**. Наприклад, таблиця «Кошики»: ключ — `userId`. Один клієнт — один кошик.
- **partition key + sort key** («сорт кі», ключ сортування) — **шухляда і порядок карток у ній**. Таблиця «Замовлення»: partition key — `customerId`, sort key — `orderDate`. Тоді легко отримати «всі замовлення пані Ірини, відсортовані за датою».



Гарячий ключ: черга до однієї каси

Усередині DynamoDB дані розкладено на багато **розділів** (partitions) — окремих «кас». У яку касу потрапить елемент, вирішує його partition key.

Якщо всі запити йдуть з **одним і тим самим** ключем — наприклад, ключем стала сьогоднішня дата — вся черга стоїть до **однієї каси**, а решта простоюють. Ця каса «перегрівається», і база починає відмовляти (**throttling**).



Правило: обирай partition key з **великою кількістю різних значень** (високою кардинальністю) — `userId`, `orderId`. Якщо ключ «гарячий» від природи, додають до нього випадковий суфікс (**write sharding**).

Як шукати: Query, Scan та індекси

- **Query** — пошук **за ключем**: швидко й дешево. Саме так і треба працювати з DynamoDB.
- **Scan** — переглянути **всю таблицю**: повільно й дорого, для рідкісних задач.

Щоб шукати ще й за іншими полями, створюють **індекси**:

- **GSI** («джі-ес-ай», global secondary index) — з **іншим** partition key; можна додати **будь-коли**. Наприклад, шукати замовлення за статусом.
- **LSI** («ел-ес-ай», local secondary index) — з **тим самим** partition key, але іншим sort key; створюється **лише разом з таблицею**.

Режими оплати

- **On-demand** («он-діманд») — платиш **за кожен запит**, нічого не плануючи. Для **непередбачуваного** чи нового трафіку.
- **Provisioned** («провіжнд») — заздалегідь задаєш кількість одиниць читання й запису (**RCU** і **WCU**), а **auto scaling** підкручує їх за навантаженням. Для **стабільного** трафіку — дешевше (можна ще й купити reserved capacity).

Розрахунок RCU і WCU — простіше, ніж здається

- **1 RCU** («ар-сі-ю») = **1 строго узгоджене читання** за секунду елемента **до 4 KB** (або **2** звичайні, eventually consistent, читання).
- **1 WCU** («дабл-ю-сі-ю») = **1 запис** за секунду елемента **до 1 KB**.

- Транзакційні читання й записи коштують **удвічі** більше.
- Розмір **округлюють угору**: елемент 10 KB — це 3 блоки по 4 KB для читання і 10 блоків по 1 KB для запису.



Розрахунок: елемент 10 KB — три «читальні» блоки по 4 KB і десять «записних» по 1 KB

Приклад:

- 100 строго узгоджених читань за секунду по 10 KB → 100 × 3 = **300 RCU**;
- ті самі читання, але eventually consistent → удвічі менше, **150 RCU**;
- 100 записів за секунду по 10 KB → 100 × 10 = **1 000 WCU**.

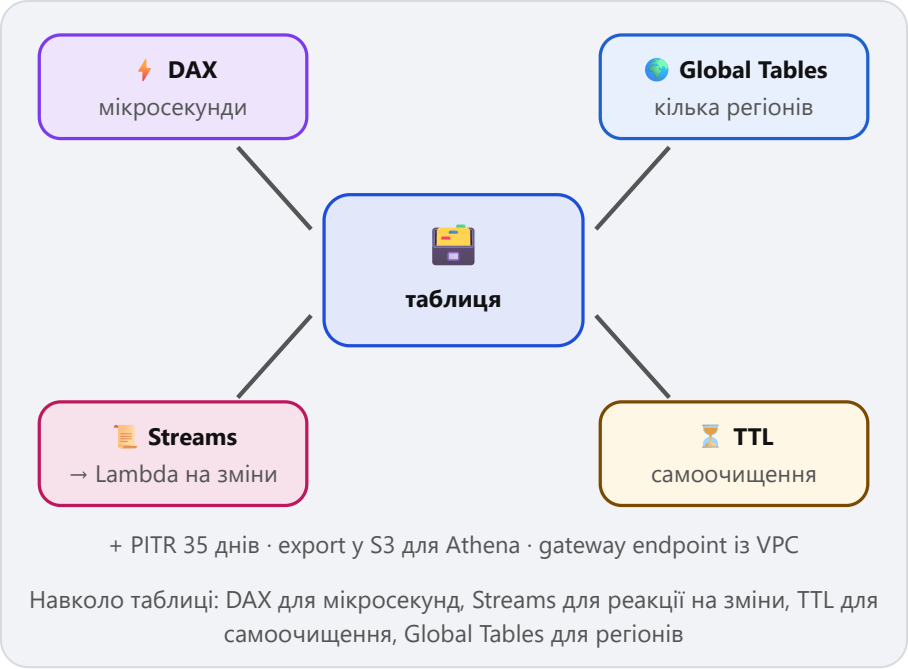
Узгодженість і транзакції

- **Eventually consistent** («евенчуелі консістент», зрештою узгоджене) читання — за замовчуванням, дешевше; щойно записане може показатися із затримкою в частку секунди.
- **Strongly consistent** (строго узгоджене) — завжди найсвіжіше, коштує вдвічі більше.
- **Transactions** — «все або нічого» для кількох елементів (як ACID у реляційних базах).

Корисні вміння DynamoDB

- **DAX** («декс», DynamoDB Accelerator) — **кеш у пам'яті** перед таблицею: відповіді за **мікросекунди**, той самий інтерфейс, код **майже не змінюється**. Працює **лише для DynamoDB**. Образ з простого курсу — **кишенька з найчастішими картками**.
- **Global Tables** — таблиця **в кількох регіонах** одночасно, записувати можна **в будь-якому** (multi-active). Узгодження між регіонами — зрештою узгоджене або (з 2025 року) **строге** з RPO = 0.

- **DynamoDB Streams** — **стрічка змін**: кожна зміна в таблиці потрапляє в потік на 24 години, і функція Lambda може на неї реагувати («нове замовлення → надіслати лист»). Для довшого зберігання й більшої кількості споживачів — **Kinesis Data Streams for DynamoDB**.
- **TTL** («ті-ті-ел», time to live) — елементи **самі зникають** після заданого часу, безкоштовно. Ідеально для сесій і тимчасових даних.
- **Бекапи**: **PITR** — відновлення на будь-яку секунду за **35 днів**; бекапи на вимогу; AWS Backup.
- **Export to S3** — вивантажити таблицю в S3 без навантаження на неї, щоб аналізувати SQL-запитами (Athena).
- **Standard-IA table class** — дешевше зберігання для таблиць, які рідко читають.
- Шифрування — за замовчуванням; доступ з VPC — через безкоштовний **gateway endpoint**.



Коли DynamoDB — не найкращий вибір

- Потрібні **складні JOIN**, довільні SQL-звіти, суворі зв'язки між таблицями → **RDS / Aurora**.
- Важка аналітика по всій історії → сховище даних (частина про аналітику).
- Великі файли → **S3** (у DynamoDB — посилання).

🥐 Кошики й профілі клієнтів «Хмаринки» переїхали в DynamoDB у режимі on-demand з ключем `userId`. Старі кошики самі зникають через 7 днів завдяки TTL, а кожне нове замовлення через Streams запускає функцію, що надсилає кухні сповіщення. У наступну Чорну п'ятницю база навіть не помітила натовпу.

📖 Словничок

DynamoDB («дайнамо-ді-бі») — serverless NoSQL-база: мілісекунди на будь-якому масштабі.

Таблиця / елемент / атрибут — ящик / картка (до 400 KB) / поле.

Partition key / sort key — номер шухляди / порядок карток у ній.

Hot partition, throttling — «гаряча» каса / відмови через перевантаження; ліки — ключ з високою кардинальністю, write sharding.

Query / Scan — пошук за ключем / перегляд усієї таблиці.

GSI / LSI — індекс з іншим partition key (будь-коли) / з тим самим ключем і іншим sort key (лише при створенні).

On-demand / provisioned — оплата за запити / заплановані RCU і WCU з auto scaling.

RCU / WCU — 1 строге читання до 4 KB за секунду (або 2 звичайні) / 1 запис до 1 KB за секунду.

Eventually / strongly consistent — зрештою узгоджене (дешевше) / строго узгоджене читання.

DAX («декс») — мікросекундний кеш лише для DynamoDB, майже без змін коду.

Global Tables — таблиця в кількох регіонах із записом у будь-якому.

Streams / TTL / PITR — стрічка змін (24 год) / самовидалення / відновлення на секунду за 35 днів.

🧩 Міні-тест: перевір себе

? Кошики, сесії й профілі гравців на мільйонах запитів, без серверів. Що обрати?

- **А** DynamoDB
- **Б** RDS for MySQL на найбільшому інстансі
- **В** Redshift
- **Г** S3 Glacier

Відповідь: А. Прості запити за ключем на величезному масштабі — сильна сторона DynamoDB.

? Який найбільший розмір одного елемента DynamoDB?

- **А** 4 KB
- **Б** 400 KB
- **В** 5 GB
- **Г** 50 TB

Відповідь: Б. 400 KB. Великі файли — в S3, а в елементі — посилання.

? Усі записи йдуть з ключем «сьогоднішня дата», і таблиця дає throttling. Що зробити?

- **А** Перейти на Scan
- **Б** Обрати partition key з багатьма значеннями (userId) або додати до ключа суфікс (write sharding)
- **В** Вимкнути auto scaling
- **Г** Збільшити розмір елемента

Відповідь: Б. Один ключ — одна «каса». Потрібен ключ, що розподіляє навантаження між розділами.

? Потрібне мікросекундне читання з DynamoDB без переписування застосунку. Що обрати?

- **А** ElastiCache з новою логікою кешу
- **Б** DAX
- **В** Read replica
- **Г** Glacier

Відповідь: Б. DAX — кеш саме для DynamoDB з тим самим інтерфейсом.

? Глобальний застосунок має записувати й читати з низькою затримкою в кількох регіонах. Що обрати?

- **А** DynamoDB Global Tables
- **Б** Одна таблиця в us-east-1
- **В** DAX
- **Г** TTL

Відповідь: А. Global Tables — multi-active: запис у будь-якому регіоні.

? На кожну зміну в таблиці треба реагувати — надсилати лист. Що використати?

- А TTL
- Б DynamoDB Streams + Lambda
- В Scan щохвилини
- Г LSI

Відповідь: Б. Streams — стрічка змін, на яку підписується функція.

? Сесії мають автоматично видалятися через 24 години. Що налаштувати?

- А TTL
- Б PITR
- В Global Tables
- Г GSI

Відповідь: А. TTL сам безкоштовно видаляє прострочені елементи.

? Потрібно 100 строго узгоджених читань за секунду елементів по 10 КВ. Скільки RCU?

А 100

Б 250

В 300

Г 1 000

Відповідь: В. 10 КВ округлюємо до 12 КВ — це 3 блоки по 4 КВ. 100 × 3 = 300 RCU.

? Трафік новий і непередбачуваний, хочеться платити лише за фактичні запити. Який режим?

А Provisioned без auto scaling

Б On-demand

В Reserved capacity на 3 роки

Г Standard-IA

Відповідь: Б. On-demand — оплата за кожен запит без планування.

✅ **Головне**

DynamoDB — serverless NoSQL: **мілісекунди** на будь-якому масштабі, кілька AZ автоматично; елемент до **400 KB**, схема гнучка.

Ключ: **partition key** (шухляда) + за потреби **sort key** (порядок); ключ — з **високою кардинальністю**, інакше **гарячий розділ** і throttling.

Шукай через **Query**, а не Scan; **GSI** — будь-коли, **LSI** — лише при створенні таблиці.

On-demand — непередбачувано; **provisioned** + auto scaling — стабільно. **1 RCU** = 1 строге читання до **4 KB** (або 2 звичайні); **1 WCU** = 1 запис до **1 KB**; округлюй угору.

DAX — мікросекунди лише для DynamoDB; **Global Tables** — кілька регіонів; **Streams** + Lambda — реакція на зміни; **TTL** — самоочищення; **PITR** — 35 днів. JOIN і звіти → RDS/Aurora.

🧠 Кеш і спеціальні бази: ElastiCache, DocumentDB, Neptune та інші

👴 Простими словами

ElastiCache — кишеня для найчастіших відповідей, щоб не питати базу щоразу. А для особливих даних є особливі бази: для документів, для зв'язків «хто з ким знайомий», для часу тощо.

🎯 Що ти дізнаєшся

- навіщо кеш, якщо є база, і які є стратегії кешування
- чим Redis/Valkey відрізняється від Memcached
- де тримати сесії stateless-серверів
- які спеціальні бази є в AWS — документи, графи, часові ряди, пошук — і коли кожна з них
- як обрати базу під задачу за одну хвилину

🥐 Сторінку меню «Хмаринки» відкривають сто тисяч разів на день — і щоразу база шукає ті самі 40 позицій. До конкурсу пекарів Олі потрібна жива таблиця лідерів. Маркетолог хоче рекомендації «з цим тортом часто беруть…», а нові розумні печі щосекунди надсилають температуру. «Чотири задачі — чотири різні інструменти, — каже Тарас. — Почнімо з записника на столі».

Навіщо кеш

Пам'ятаєш **кеш** з розділу про бази даних з нуля — **записник на столі** з відповідями на часті питання? У житті діє правило «80/20»: кілька популярних речей отримують більшість запитів. Меню, ціни, популярні торти — їх питають тисячі разів, а змінюються вони рідко.

Кеш тримає такі відповіді **в пам'яті** — тому відповідає за мікросекунди-мілісекунди, а база відпочиває. Замість купувати більший сервер бази часто досить поставити перед нею кеш.

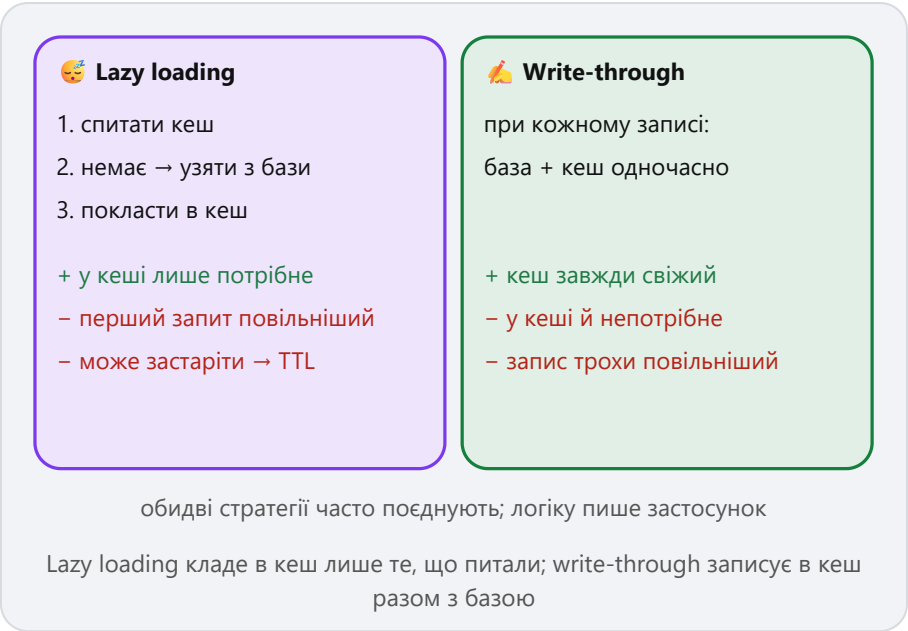
ElastiCache: Redis/Valkey і Memcached

Amazon ElastiCache («еластік кеш») — керований кеш у пам'яті. Два «двигуни»:

- **Redis OSS / Valkey** («редіс», «велкі») — багатий на вміння:
  - **реплікація** і **Multi-AZ** з автоматичним перемиканням — кеш переживе збій вузла;
  - **збереження на диск** і бекапи;
  - складні структури даних: наприклад, **sorted sets** («відсортовані набори») — готова **таблиця лідерів**;
  - повідомлення pub/sub; є й serverless-варіант.
- **Memcached** («мемкешд») — **простий** і багатопотоковий, дані діляться між вузлами, але **без реплікації** і **без збереження на диск**: вузол упав — його частина кешу зникла.

⚠️ Кеш вимагає **змін у коді**: застосунок сам має спершу зазирнути в записник і лише потім — у базу. Якщо в питанні сказано «**без змін коду**», дивись на read replicas (для RDS) або DAX (для DynamoDB).

Дві стратегії кешування



- **Lazy loading** («лейзі лоудінг», або cache-aside — «лінивий») — спершу питаємо кеш; **немає** — беремо з бази й **кладемо в кеш**. У кеші лише те, що справді питають. Мінус: дані можуть **застаріти** — тому записам дають **TTL**.
- **Write-through** («райт-сру», наскрізний запис) — при кожному **записі** в базу одразу оновлюємо й кеш. Кеш **завжди свіжий**, але в ньому лежить і те, що ніхто не читає.

Сесії для stateless-серверів

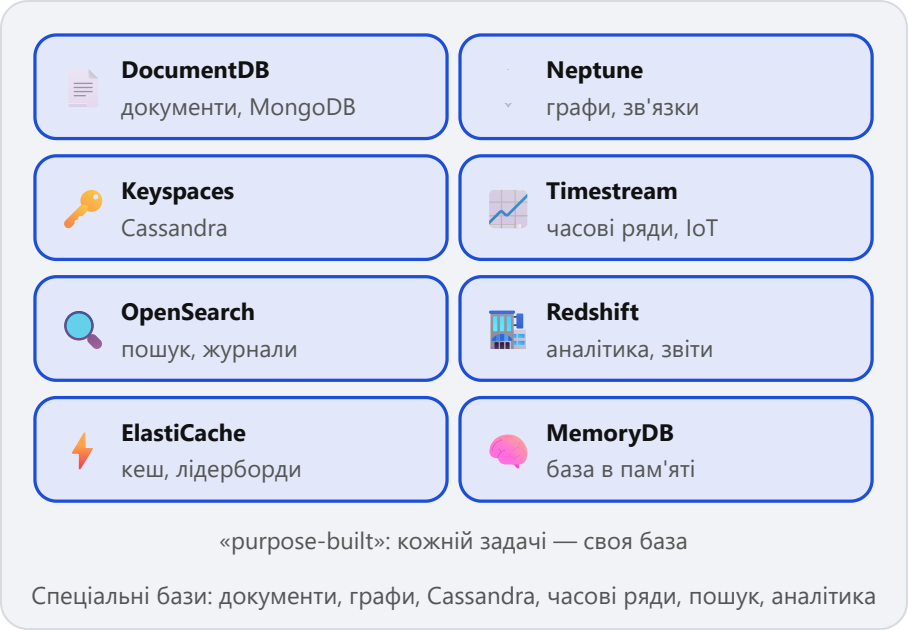
Пам'ятаєш **дошку замовлень** з розділу про застосунки? Щоб сервери були взаємозамінними, сесії (вхід, кошик) тримають не на них, а в спільному швидкому сховищі. Найчастіше — **ElastiCache (Redis/Valkey)** або **DynamoDB**.

MemoryDB — не кеш, а база в пам'яті

Amazon MemoryDB — сумісна з Redis/Valkey, але це **основна база**, а не кеш: дані надійно зберігаються в кількох AZ. Для випадків, коли потрібна швидкість пам'яті **і** надійність бази.

Спеціальні бази: кожній задачі — своя

AWS дотримується принципу «**purpose-built**» — **кожній задачі своя база**:



- **Amazon DocumentDB** — **документна** база, сумісна з **MongoDB**. Перенести застосунок на MongoDB в керований сервіс.
- **Amazon Neptune** («нептун») — **графова** база, **карта знайомств**: соцмережі («друзі друзів»), рекомендації («з цим тортом беруть...»), пошук шахрайських ланцюжків, бази знань.
- **Amazon Keyspaces** — сумісна з **Apache Cassandra**, serverless.
- **Amazon Timestream** — **часові ряди**: показник + мітка часу. Датчики печей, метрики, телеметрія IoT.

- **Amazon OpenSearch Service** — **пошуковик** по власних даних і аналіз журналів (про нього ще в частині про аналітику).
- **Amazon Redshift** — **сховище даних** для важкої аналітики (наступна частина).
- **Amazon QLDB** (журнал-реєстр) — **закритий 31 липня 2025 року**, не обирай його.

Як обрати базу за хвилину

Вимога	Сервіс
Реляційна, SQL, JOIN, транзакції	RDS / Aurora
Максимальна доступність і швидкість реляційної бази	Aurora
Ключ-значення, мілісекунди, будь-який масштаб, serverless	DynamoDB
Кеш, сесії, таблиця лідерів	ElastiCache (Redis / Valkey)
Мікросекунди для DynamoDB без змін коду	DAX
Аналітика, звіти по всій історії, петабайти	Redshift
Сумісність з MongoDB	DocumentDB
Графи, зв'язки	Neptune
Cassandra	Keyspaces
Часові ряди, IoT	Timestream
Повнотекстовий пошук, журнали	OpenSearch

Не плутай

- **DAX чи ElastiCache?** DAX — лише для DynamoDB і майже без змін коду. ElastiCache — для будь-якої бази, але логіку кешу пише застосунок.
- **Redis/Valkey чи Memcached?** Redis — реплікація, Multi-AZ, збереження, складні структури. Memcached — простий, без реплікації й без збереження.

> 🥐 Тарас поставив ElastiCache (Valkey) перед базою: меню тепер віддається за мілісекунди, а база отримує в десятки разів менше запитів. Таблиця лідерів конкурсу — sorted set у тому самому кеші. Рекомендації будує Neptune, а температура печей щосекунди летить у Timestream — і технолог бачить графік кожної печі.

📖 **Словничок**

ElastiCache («еластік кеш») — керований кеш у пам'яті: Redis OSS / Valkey і Memcached.

Redis / Valkey — кеш з реплікацією, Multi-AZ, збереженням і структурами (sorted sets — лідерборди).

Memcached — простий багатопотоковий кеш без реплікації й без збереження.

Lazy loading (cache-aside) — у кеш потрапляє запитуване; можливі застарілі дані → TTL.

Write-through — запис одночасно в базу й кеш; кеш завжди свіжий.

MemoryDB — Redis-сумісна основна база в пам'яті з надійністю.

Purpose-built — «кожній задачі своя база».

DocumentDB / Neptune / Keyspaces / Timestream — MongoDB-сумісна / графова / Cassandra-сумісна / часові ряди.

OpenSearch / Redshift — пошук і журнали / сховище даних для аналітики.

QLDB — закритий у 2025 році, не обирати.

🧩 **Міні-тест: перевір себе**

? Потрібна таблиця лідерів конкурсу в реальному часі. Що обрати?

- **А** ElastiCache Redis/Valkey (sorted sets)
- **Б** Memcached
- **В** Glacier
- **Г** Neptune

Відповідь: А. Sorted sets у Redis/Valkey — готова таблиця лідерів.

? Сторінка меню щоразу запитує ті самі дані з RDS, і база перевантажена. Що допоможе найдешевше?

- **А** Більший клас інстансу бази
- **Б** Кеш ElastiCache з lazy loading
- **В** Read replica в іншому регіоні
- **Г** Перейти на Glacier

Відповідь: Б. Часті однакові запити — ідеальна робота для кешу.

? Потрібен простий багатопотоковий кеш, реплікація не потрібна. Що обрати?

- **А** Redis з Multi-AZ
- **Б** Memcached
- **В** MemoryDB
- **Г** DAX

Відповідь: Б. Memcached — простий і швидкий, без реплікації й збереження.

? Застосунок на MongoDB треба перенести в керований сервіс AWS. Що обрати?

- **А** Amazon DocumentDB
- **Б** Amazon Keyspaces
- **В** Amazon Neptune
- **Г** Amazon Timestream

Відповідь: А. DocumentDB сумісна з MongoDB.

? Соцмережа: «друзі друзів» і пошук шахрайських ланцюжків. Що обрати?

- **А** DynamoDB
- **Б** Amazon Neptune
- **В** Redshift
- **Г** ElastiCache

Відповідь: Б. Neptune — графова база, «карта знайомств».

? Датчики печей щосекунди надсилають температуру з міткою часу. Що обрати?

- **А** Neptune
- **Б** DocumentDB
- **В** Amazon Timestream
- **Г** QLDB

Відповідь: В. Timestream — база часових рядів для IoT і метрик.

? Кеш має пережити збій вузла без втрати даних. Що обрати?

- **А** ElastiCache Redis/Valkey з реплікацією і Multi-AZ
- **Б** Memcached
- **В** Кеш на одному сервері EC2
- **Г** Instance store

Відповідь: А. Redis/Valkey вміє репліки й автоматичне перемикання; Memcached — ні.

? У питанні сказано: прискорити читання з DynamoDB «без змін коду». Що обрати?

А ElastiCache

Б DAX

В Neptune

Г Read replica

Відповідь: Б. ElastiCache потребує логіки кешу в коді. DAX — майже без змін.

✅ **Головне**

Кеш — «записник на столі»: часті відповіді з пам'яті, база відпочиває. **Потребує змін у коді** (крім DAX).

ElastiCache: **Redis/Valkey** — реплікація, Multi-AZ, збереження, **sorted sets** (лідерборди); **Memcached** — простий, без реплікації.

Стратегії: **lazy loading** (лише запитуване, TTL проти застарілого) і **write-through** (завжди свіже). Сесії stateless-серверів — у **ElastiCache** чи **DynamoDB**.

MemoryDB — основна база в пам'яті. Спеціальні: **DocumentDB** (MongoDB), **Neptune** (графи), **Keyspaces** (Cassandra), **Timestream** (часові ряди), **OpenSearch** (пошук), **Redshift** (аналітика); QLDB закритий.

«Без змін коду» для DynamoDB → **DAX**; для RDS → **read replicas**.

🌊 Потоки даних: Kinesis, Firehose, Flink і MSK

👴 Простими словами

Потік даних — як конвеєр: дані їдуть безперервно (кліки, датчики, логи), і їх обробляють прямо на ходу. Kinesis — сам конвеєр, Firehose — конвеєр, що сам усе розвантажує на склад.

🎯 Що ти дізнаєшся

- чим пакетна обробка відрізняється від потокової
- чим потік відрізняється від черги — і чому це важливо на іспиті
- як влаштований Kinesis Data Streams — shards, ключі, перемотка
- що робить Data Firehose — «трубопровід без коду»
- навіщо Managed Service for Apache Flink, Kinesis Video Streams і MSK

🥐 Мобільний застосунок «Хмаринки» щосекунди надсилає тисячі подій: хто який торт розглядає, що додає в кошик. Маркетолог хоче бачити **наживо**, що стає популярним («чізкейк раптом у тренді!»), розумні печі шлють температуру, а все разом має складатися в сховище для майбутніх звітів. «Це вже не краплі, а річка, — каже Тарас. — Для річок у AWS свої інструменти».

Відро раз на день чи річка

- **Пакетна обробка** (batch, «батч») — **відро води раз на день**: дані збирають, а потім обробляють великою порцією. Звіт за вчорашній день — типовий приклад.
- **Потокова обробка** (streaming, «стрімінг») — **річка**: дані течуть безперервно, і їх обробляють одразу, щойно вони з'являються.

Щодо швидкості кажуть так: **real-time** — реальний час (мілісекунди–секунди), **near real-time** — майже реальний час (секунди–хвилини), **batch** — години.

Черга чи потік

Ти вже знаєш чергу SQS. Потік схожий, але принципово інший:



Черга: кожне повідомлення бере один обробник і видаляє. Потік: усі читають ту саму річку і можуть перемотати назад

- **Черга (SQS)** — кожне повідомлення **бере один** обробник і після обробки **видаляє**. Як талончик у поліклініці.
- **Потік (Kinesis)** — дані **лишаються** на заданий час, і їх **незалежно** читають **багато** споживачів; кожен може **перемотати** назад і перечитати. Образ з простого курсу — **річка з пам'яттю**: кілька заводів беруть воду з тієї самої річки й можуть «перемотати» її назад.

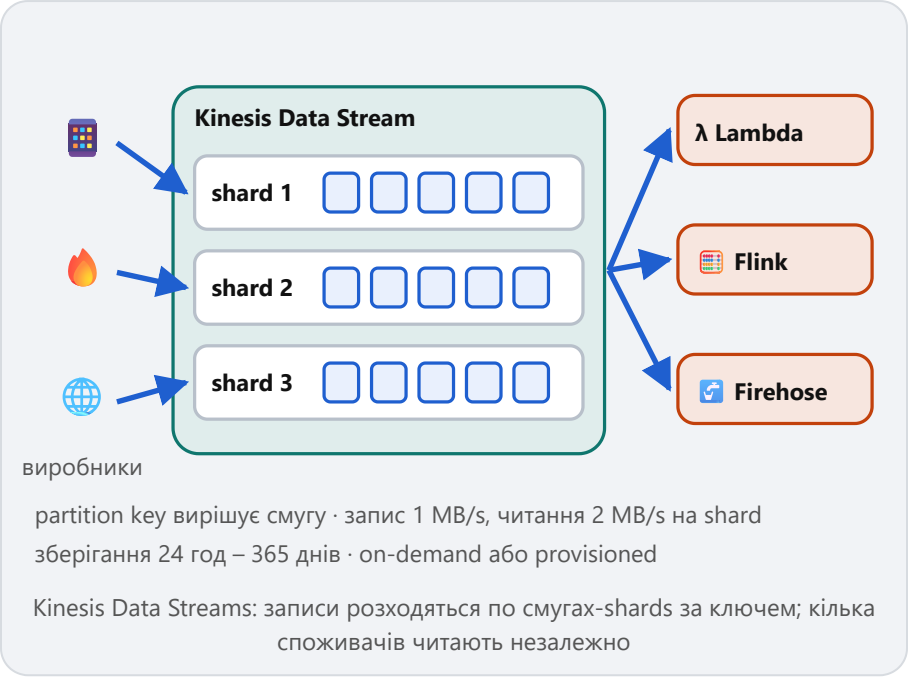
🎯 «Кілька застосунків мають читати ті самі дані в реальному часі й мати змогу перечитати» → **потік (Kinesis Data Streams)**, а не черга.

Kinesis Data Streams — річка зі смугами

Amazon Kinesis Data Streams («кінезис», KDS) — потік даних у **реальному часі** (затримка — десятки–сотні мілісекунд).

- **Виробники** (producers) — застосунки, датчики — пишуть у потік **записи** (records).
- Потік складається з **shards** («шардз») — **смуг** річки. Кожна смуга: запис до **1 MB/s** або 1 000 записів за секунду, читання до **2 MB/s**.
- **Partition key** кожного запису вирішує, у яку смугу він потрапить; усередині смуги порядок зберігається.
- Дані зберігаються **24 години** за замовчуванням і **до 365 днів** — тому можна **перемотати** (replay).
- **Споживачі**: функції Lambda, власні застосунки, Flink, Firehose. **Enhanced fan-out** дає кожному споживачу власні 2 MB/s.
- Режими: **on-demand** (смуги додаються самі) або **provisioned** (кількість смуг задаєш вручну).

- Запис — до 1 MiB (з жовтня 2025 можна налаштувати до 10 MiB).



⚠️ Якщо один shard перевантажений (помилка ProvisionedThroughputExceeded), — «гаряча смуга», як «гарячий ключ» у DynamoDB. Ліки: **кращий partition key**, **розділити shard** (split) або **on-demand** режим.

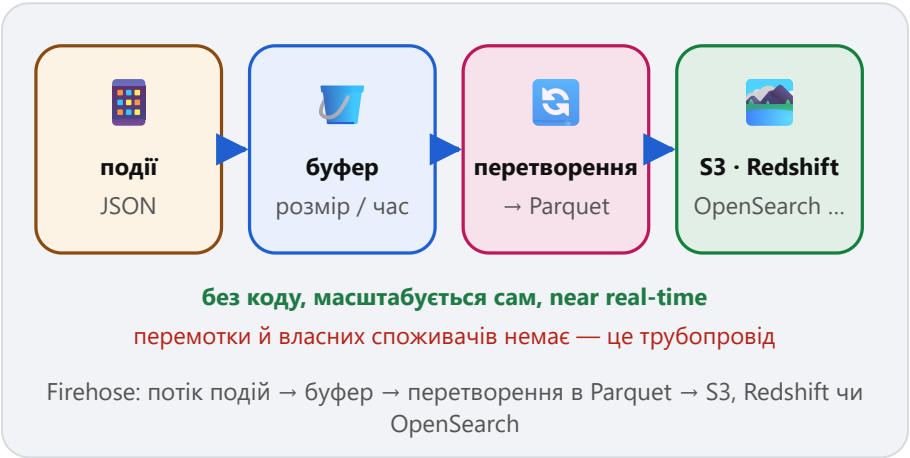
Data Firehose — трубопровід без коду

Amazon Data Firehose («фаєрхоуз», колишня назва — Kinesis Data Firehose) — **трубопровід**, що **сам доставляє**

потік у сховища:

- куди: **S3**, **Redshift**, **OpenSearch**, Splunk, будь-які HTTP-адреси, Snowflake, таблиці Iceberg;
- **near real-time**: дані накопичуються в буфері (за розміром чи часом) і вивантажуються порціями;
- повністю керований і масштабується сам — **коду писати не треба**;
- по дорозі вміє **перетворювати** (через Lambda), **конвертувати JSON у Parquet** чи ORC і **стискати**.

А от **перемотки й власних споживачів** у Firehose **немає** — це саме трубопровід, а не річка з пам'яттю.



🎯 «Складати потік у S3 (чи Redshift, OpenSearch) без коду, майже в реальному часі» → **Data Firehose**. «Перетворити JSON у Parquet перед збереженням» → **Firehose**.

Flink — аналіз «на льоту»

Amazon Managed Service for Apache Flink («флінк», колишня назва — Kinesis Data Analytics) — аналізує потік **просто під час руху**:

- **вікна часу** — «скільки замовлень за останні 5 хвилин у кожному місті»;
- **аномалії** — «замовлень чізкейка раптом утричі більше, ніж зазвичай»;
- писати можна на SQL, Java чи Python.

Відео і Kafka

- **Kinesis Video Streams** — потокове **відео з камер** для аналітики й машинного навчання (наприклад, розпізнавання в Rekognition Video).
- **Amazon MSK** («ем-ес-кей») — керований **Apache Kafka** — найпопулярнішої у світі платформи потоків. Обирають, коли застосунки **вже працюють з Kafka**. Є MSK Serverless і MSK Connect.

Що обрати

Потреба	Сервіс
Потік у реальному часі, кілька своїх споживачів, перемотка	Kinesis Data Streams
Просто доставити потік у S3 / Redshift / OpenSearch без коду	Data Firehose
Аналіз потоку у вікнах часу, аномалії	Managed Service for Apache Flink
Відео з камер	Kinesis Video Streams
Наявні застосунки на Kafka	Amazon MSK
Завдання, які має обробити один обробник, і видалити	SQS (не потік)

🥐 Тепер події застосунку «Хмаринки» летять у Kinesis Data Streams. Flink рахує популярність тортів за останні 10 хвилин і сповіщає маркетолога про тренди, а Firehose складає всі події в S3 у форматі Parquet — для звітів, про які наступний розділ.

📖 Словничок

Batch / streaming — обробка порціями / безперервно.

Real-time / near real-time — мілісекунди–секунди / секунди–хвилини.

Kinesis Data Streams (KDS) — потік у реальному часі зі смугами (shards), зберігання до 365 днів, перемотка.

Shard («шард») — «смуга» потоку: запис 1 MB/s, читання 2 MB/s.

Partition key — визначає смугу запису й порядок усередині неї.

Replay — перечитати дані з минулого.

Enhanced fan-out — власні 2 MB/s для кожного споживача.

Data Firehose («фаєрхоуз») — доставка потоку в сховища без коду, з перетворенням у Parquet; без перемотки.

Managed Service for Apache Flink («флінк») — аналіз потоку у вікнах часу.

Kinesis Video Streams — потокове відео з камер.

MSK («ем-ес-кей») — керований Apache Kafka.

🧩 **Міні-тест: перевір себе**

? Кілька застосунків мають незалежно читати той самий потік кліків у реальному часі й за потреби перечитувати його. Що обрати?

- **А** Kinesis Data Streams
- **Б** SQS Standard
- **В** Data Firehose
- **Г** SNS

Відповідь: А. Потік з пам'яттю: багато споживачів, перемотка до 365 днів.

? Потрібно складати потік подій у S3 у форматі Parquet, не пишучи коду. Що обрати?

А Kinesis Data Streams з власним застосунком

Б Amazon Data Firehose

В Amazon MSK

Г SQS FIFO

Відповідь: Б. Firehose — трубопровід, що сам доставляє дані й конвертує їх у Parquet.

? Чим потік відрізняється від черги?

А У потоці дані лишаються на час зберігання, їх читають багато споживачів і можна перемотати; повідомлення з черги бере один обробник і видаляє

Б Нічим

В Черга швидша за будь-який потік

Г Потік не зберігає дані зовсім

Відповідь: А. Черга — талончики; потік — річка з пам'яттю.

? Один shard Kinesis перевантажений і видає ProvisionedThroughputExceeded. Що допоможе?

А Вимкнути потік

Б Кращий partition key, розділити shard або перейти в on-demand

В Перейти на Glacier

Г Збільшити тайм-аут

Відповідь: Б. «Гаряча смуга» — як гарячий ключ: розподілити навантаження або додати смуг.

? Потрібно в реальному часі рахувати замовлення за останні 5 хвилин і ловити аномалії. Що обрати?

А Athena

Б Managed Service for Apache Flink

В Glue

Г Data Exchange

Відповідь: Б. Flink аналізує потік у вікнах часу, просто під час руху.

? Компанія вже має застосунки на Apache Kafka і хоче перенести їх у керований сервіс. Що обрати?

А Kinesis Data Streams

Б Amazon MSK

В SQS

Г EventBridge

Відповідь: Б. MSK — керований Kafka; код переписувати не треба.

? Скільки найдовше можуть зберігатися дані в Kinesis Data Streams?

А 24 години

Б 14 днів

В 365 днів

Г Вічно

Відповідь: В. За замовчуванням 24 години, максимум — 365 днів.

✅ **Головне**

Batch — відро раз на день; **streaming** — річка. **Черга** — один обробник і видалення; **потік** — багато споживачів і **перемотка**.

Kinesis Data Streams — реальний час: **shards** (1 MB/s запис, 2 MB/s читання), **partition key**, зберігання **24 год – 365 днів**, replay; гаряча смуга → кращий ключ, split, on-demand.

Data Firehose — доставка в **S3, Redshift, OpenSearch** без коду, near real-time, **конвертація в Parquet**; без перемотки.

Flink — аналіз у вікнах часу й аномалії; **Kinesis Video Streams** — відео з камер; **MSK** — Kafka.

📊 Аналітика: озеро даних, Glue, Athena, Redshift і дашборди

👴 Простими словами

Аналітика — це коли з гори даних витягують відповіді. Озеро даних (S3) — усе в одному місці, Glue прибирає й описує, Athena ставить питання, Redshift рахує великі звіти, QuickSight малює графіки.

🎯 Що ти дізнаєшся

- чим озеро даних відрізняється від сховища даних
- чому формат Parquet і партиції роблять аналітику в десятки разів дешевшою
- що роблять Glue, Athena, Redshift, EMR, OpenSearch і Amazon Quick
- як Lake Formation охороняє дані до окремих колонок
- як виглядає повний конвеєр аналітики від кліку до дашборда

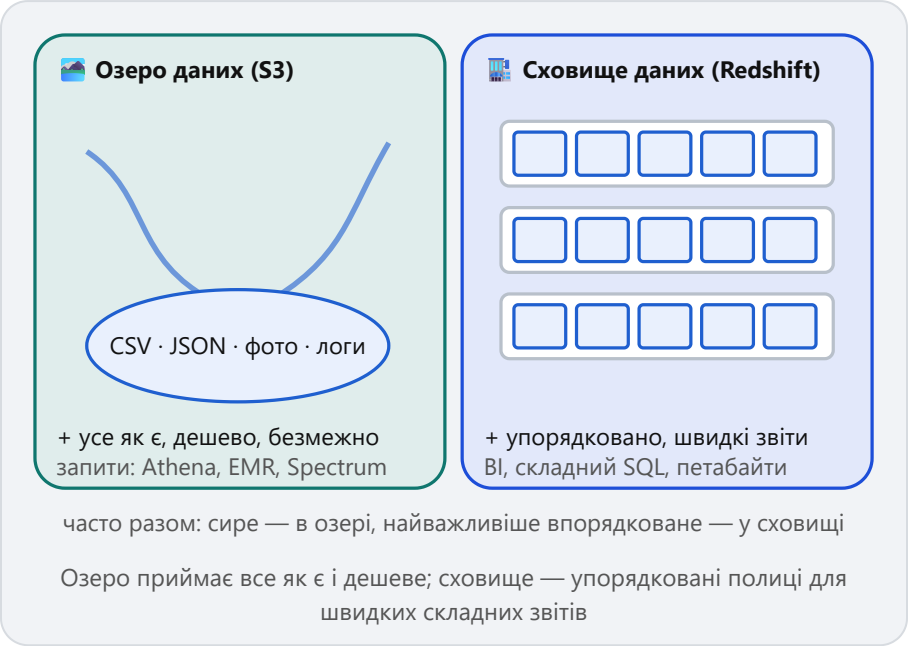
🥐 Оля хоче знати, які торти продаються найкраще — за місяцями, містами й днями тижня, за три роки. Але дані розкидані: замовлення в базі, кліки з застосунку — у S3, вивантаження з кас — у файлах CSV. «Якщо запускати такі звіти на робочій базі, сайт ляже, — попереджає Тарас. — Для аналітики — окрема кухня. Її звуть озером даних».

Озеро даних і сховище даних

Пам'ятаєш «касира» (OLTP) і «бухгалтера» (OLAP) з розділу про бази з нуля? Для бухгалтерської роботи — важких звітів по всій історії — є два підходи:

- **Data lake** («дата лейк», **озеро даних**) — **озеро, куди стікають усі річки даних у сирому вигляді**: будь-які формати, будь-які обсяги. В AWS озеро — це **S3**: дешево, безмежно, гнучко.
- **Data warehouse** («дата веєрхаус», **сховище даних**) — **склад з упорядкованими полицями**: дані заздалегідь очищені й структуровані, щоб складні звіти працювали миттєво. В AWS — **Redshift**.

Часто використовують обидва: сирі дані — в озері, найважливіше й упорядковане — у сховищі.

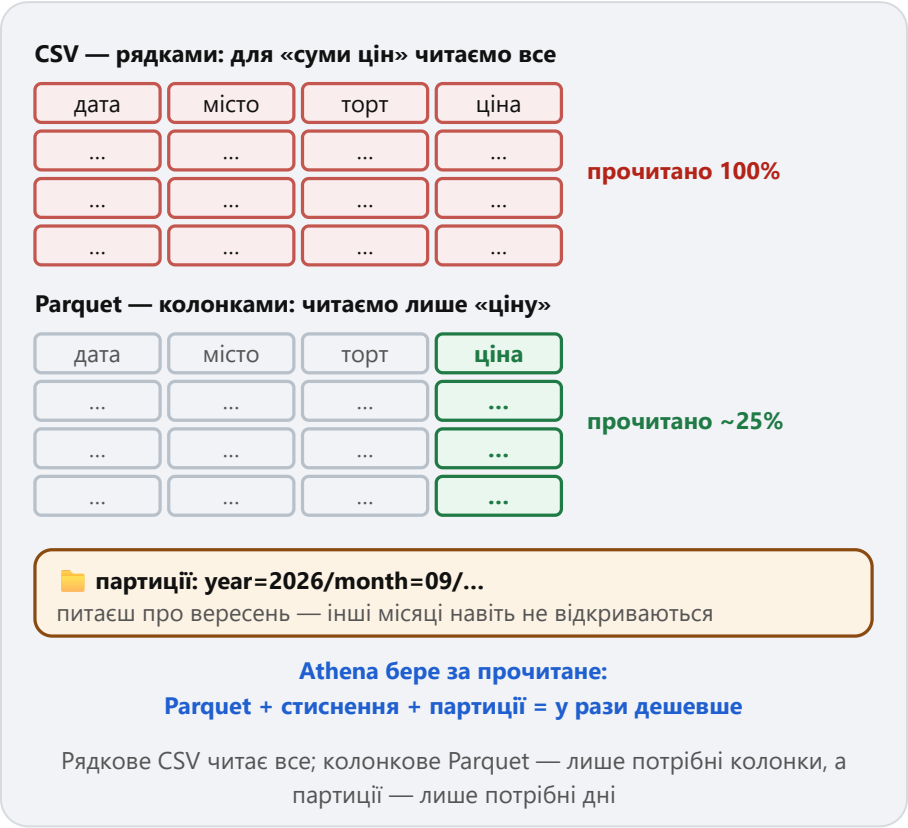


Формати файлів: читати всю книжку чи одразу потрібну главу

Від формату файлів залежить, скільки даних доведеться прочитати для відповіді — а отже, скільки часу й грошей піде.

- **CSV** — текст, де дані записано **рядками**. Щоб порахувати суму однієї колонки «ціна», доводиться прочитати **всі рядки повністю**. Образ з простого курсу — **щоб знайти одне слово, читаєш усю книжку**.

- **Parquet** («паркет») і **ORC** — **колонкові** стиснені формати: кожна колонка зберігається окремо. Потрібна лише «ціна» — читаєш лише її. Образ — **одразу відкриваєш потрібну главу**.
- **Партиції** (partitions) — дані розкладають по «папках» за датою: `year=2026/month=09/day=30/`. Питаєш про вересень — читаються лише вересневі папки.



Наскільки це важливо? Сервіс Athena бере плату **за прочитані дані** — приблизно 5 доларів за терабайт. Запит по 1 ТБ сирих CSV — близько 5 доларів. Той самий запит по стиснених Parquet з партиціями може прочитати лише кілька відсотків — і коштувати центи. Саме тому на іспиті «зменшити вартість запитів» майже завжди означає **Parquet + стиснення + партиції**.

AWS Glue — бібліотекар

AWS Glue («глу», клей) — **бібліотекар** озера:

- **Crawlers** («кролери», обхідники) — самі обходять файли в S3 і **визначають їхню структуру** (схему).
- **Data Catalog** — **каталог**: опис усіх таблиць озера. Ним користуються Athena, Redshift Spectrum, EMR.
- **ETL jobs** (extract, transform, load — витягти, перетворити, завантажити) — **serverless**-перетворення даних на Spark: наприклад, **CSV** → **Parquet** з партиціями.
- **Job bookmarks** («закладки») — обробляти лише **нові** файли, а не все щоразу заново.
- **Glue DataBrew** — візуальна підготовка даних без коду.

Athena — питай SQL прямо в озера

Amazon Athena («афіна») — **serverless SQL прямо по файлах у S3**: даних нікуди не завантажуєш, серверів не маєш.

- Платиш **за прочитані дані** (тому Parquet і партиції!).
- Ідеальна для **разових запитів** і **аналізу журналів**: CloudTrail, балансувальників, VPC Flow Logs, CloudFront.
- **Federated Query** — SQL ще й до RDS, DynamoDB та інших джерел через конектори.
- **CTAS** (CREATE TABLE AS SELECT) — перетворити CSV у Parquet одним запитом.

⚠️ Athena — не сховище: дані лишаються в S3, Athena лише ставить до них питання.

Redshift — склад з упорядкованими полицями

Amazon Redshift («редшифт») — **колонкове сховище даних** для важкої аналітики:

- **петабайти** даних і складні SQL-звіти; багато вузлів працюють паралельно (MPP);
- **Redshift Serverless** — без керування кластером;
- **Redshift Spectrum** — **зі складу зазирнути в озеро**: запити до даних у S3 без завантаження;

- **Concurrency Scaling** — додаткова потужність у пікові години звітів;
- **Zero-ETL** — дані з Aurora, RDS і DynamoDB з'являються в Redshift майже одразу, без конвеєрів;
- найшвидше завантаження з S3 — команда **COPY** (паралельно з багатьох файлів);
- аварійне відновлення — копіювання знімків в інший регіон.

🎯 Разові SQL-запити по S3 → **Athena**. Регулярна важка BI-аналітика на петабайтах → **Redshift**.

EMR — важкий завод

Amazon EMR («і-ем-ар») — керовані кластери **Hadoop, Spark, Hive, Presto** для великих даних. Обирають, коли потрібен **контроль над фреймворками** або вже є готовий код на Spark чи Hadoop. Кластер складається з головного, основних і **задачних (task) вузлів** — останні можна брати на **Spot**, бо їхня втрата не губить даних. Є EMR Serverless і EMR on EKS.

OpenSearch і Amazon Quick

- **Amazon OpenSearch Service** («опенсьорч») — **пошуковик** по власних даних (каталог тортів з пошуком «шоколад без цукру») і **аналіз журналів**, з дашбордами.
- **Amazon Quick** (колишня назва — QuickSight) — **вітрина з дашбордами** для бізнесу: serverless, власний швидкий рушій у пам'яті (SPICE), ML-підказки, вбудовування дашбордів у застосунки, доступ «кожен бачить лише свої рядки».

Lake Formation — охорона озера

У озері лежить усе: і продажі, і персональні дані клієнтів. **AWS Lake Formation** — **охорона озера**: централізовані дозволи **аж до окремих колонок, рядків і клітинок** для Athena, Redshift Spectrum, EMR і Glue. Маркетолог бачить продажі, але не бачить телефонів клієнтів. Ще Lake Formation допомагає швидко збудувати озеро й ділитися даними між акаунтами.

Повний конвеєр: від кліку до дашборда



Конвеєр аналітики «Хмаринки»: джерела → приймання → сире озеро → Glue → впорядковане озеро → запити → дашборди, під охороною Lake Formation

1. **Джерела**: застосунок, каси, бази, SaaS-сервіси.
2. **Приймання**: потоки — Kinesis і Firehose; бази — DMS; SaaS — AppFlow.
3. **Сире озеро**: S3, як є.
4. **Glue**: визначає схеми й перетворює в Parquet з партиціями → **впорядковане озеро**.
5. **Запити**: Athena (разові), Redshift (регулярні звіти), EMR (важкі обчислення).

6. **Дашборди**: Amazon Quick.
7. Над усім — **Lake Formation** з дозволами.

Ще один сервіс для повноти: **AWS Data Exchange** — підписка на **сторонні набори даних** (погода, фінанси) з доставкою просто в S3.

Що обрати

Потреба	Сервіс
Разові SQL-запити по файлах у S3, аналіз журналів	Athena
Регулярна важка BI-аналітика на петабайтах	Redshift
Запити з Redshift до даних у S3 без завантаження	Redshift Spectrum
Serverless ETL, каталог даних, CSV → Parquet	Glue
Spark/Hadoop з контролем над кластером	EMR
Пошук і аналіз журналів	OpenSearch
Дашборди для бізнесу	Amazon Quick
Дозволи до колонок і рядків в озері	Lake Formation
Сторонні набори даних	Data Exchange

> 🥐 Через тиждень Оля побачила дашборд: найкраще в Кракові продаються макаронс, у Львові — сирники, а в Торонто по неділях злітає попит на медівник. Звіт за три роки будується за кілька секунд і коштує центи, а маркетолог бачить продажі, але не бачить жодного телефону клієнта.

📖 **Словничок**

Data lake / data warehouse («дата лейк / дата веєрхаус») — озеро сирих даних (S3) / сховище впорядкованих даних (Redshift).

CSV / Parquet / ORC — рядковий текст / колонкові стиснені формати.

Партиції — розкладання даних по «папках», наприклад за датою.

Glue («глу») — crawlers, Data Catalog, serverless ETL, job bookmarks, DataBrew.

ETL — витягти, перетворити, завантажити.

Athena («афіна») — serverless SQL по S3, оплата за прочитані дані; Federated Query, CTAS.

Redshift («редшифт») — колонкове сховище даних: Serverless, Spectrum, Concurrency Scaling, Zero-ETL, COPY.

EMR («і-ем-ар») — кластери Spark/Hadoop; task-вузли на Spot.

OpenSearch — пошук і аналіз журналів. **Amazon Quick** (QuickSight) — дашборди.

Lake Formation — дозволи до колонок, рядків і клітинок в озері.

Data Exchange — сторонні набори даних.

🧩 **Міні-тест: перевір себе**

? Потрібні SQL-запити по журналах у S3 без жодних серверів. Що обрати?

- **А** Amazon Athena
- **Б** Amazon Redshift з великим кластером
- **В** Amazon EMR
- **Г** DynamoDB

Відповідь: А. Athena ставить SQL-питання просто до файлів у S3; платиш за прочитані дані.

? Як найсильніше зменшити вартість запитів Athena?

А. Зберігати дані як один великий CSV

Б. Перетворити дані в Parquet зі стисненням і партиціями

В. Перейти в інший регіон

Г. Робити запити частіше

Відповідь: Б. Колонковий формат і партиції зменшують обсяг прочитаного — отже, і рахунок.

? Потрібен serverless ETL: визначити схеми файлів у S3, перетворити CSV у Parquet і вести каталог таблиць. Що обрати?

- **А** Athena
- **Б** AWS Glue
- **В** Amazon Quick
- **Г** Redshift Spectrum

Відповідь: Б. Glue — бібліотекар: crawlers, Data Catalog і ETL-задачі.

? Регулярна важка BI-аналітика на петабайтах структурованих даних. Що обрати?

- **А** Athena
- **Б** Amazon Redshift
- **В** DynamoDB
- **Г** ElastiCache

Відповідь: Б. Redshift — сховище даних для постійних складних звітів.

? Потрібно запитувати дані в S3 прямо з Redshift, не завантажуючи їх. Що обрати?

- **А** Redshift Spectrum
- **Б** Redshift COPY
- **В** Glue DataBrew
- **Г** Kinesis

Відповідь: А. Spectrum — «зі складу зазирнути в озеро».

? Маркетолог має бачити продажі, але не колонку з телефонами клієнтів. Що налаштувати?

- **А** S3 Block Public Access
- **Б** AWS Lake Formation з дозволами на рівні колонок
- **В** Amazon Quick без паролів
- **Г** CloudFront

Відповідь: Б. Lake Formation дає дозволи аж до колонок, рядків і клітинок.

? Керівникам потрібні інтерактивні дашборди продажів. Що обрати?

- **А** CloudWatch
- **Б** Amazon Quick (колишній QuickSight)
- **В** Glue
- **Г** OpenSearch Serverless для бухгалтерії

Відповідь: Б. Amazon Quick — BI-вітрина з дашбордами для бізнесу.

? Великі задачі Spark, потрібен контроль над кластером і дешево. Що обрати?

- **А** Athena
- **Б** Amazon EMR з task-вузлами на Spot
- **В** Redshift Serverless
- **Г** Lambda

Відповідь: Б. EMR — керований Spark/Hadoop; задачні вузли на Spot суттєво здешевлюють.

✅ **Головне**

Data lake (S3) — озеро сирих даних; **data warehouse** (Redshift) — упорядковані полиці для швидких звітів.

Parquet/ORC + стиснення + партиції — читаєш лише потрібне: у рази швидше й дешевше (Athena бере за прочитане).

Glue — crawlers, **Data Catalog**, serverless ETL, job bookmarks. **Athena** — serverless SQL по S3 (журнали, разові запити).

Redshift — важка регулярна аналітика: Serverless, **Spectrum** (до S3), Zero-ETL, COPY. **EMR** — Spark/Hadoop (task-вузли на Spot). **OpenSearch** — пошук і журнали. **Amazon Quick** — дашборди.

Lake Formation — дозволи до колонок і рядків; конвеєр: джерела → Kinesis/Firehose/DMS/AppFlow → S3 → Glue → Athena/Redshift/EMR → Quick.

🤖 Штучний інтелект без магії: готові AI-сервіси, SageMaker і Bedrock

👴 Простими словами

AI-сервіси — готові «розумні помічники»: розпізнати фото, перекласти, перетворити голос на текст. SageMaker — майстерня для власних моделей, Bedrock — доступ до великих мовних моделей.

🎯 Що ти дізнаєшся

- що таке машинне навчання простими словами — модель, навчання й відповідь
- які «органи чуття» AWS можна підключити до застосунку без жодних знань ML
- як поєднувати їх у конвеєри
- коли потрібна власна модель у SageMaker AI, а коли — генеративний AI в Bedrock
- що роблять Amplify і Device Farm

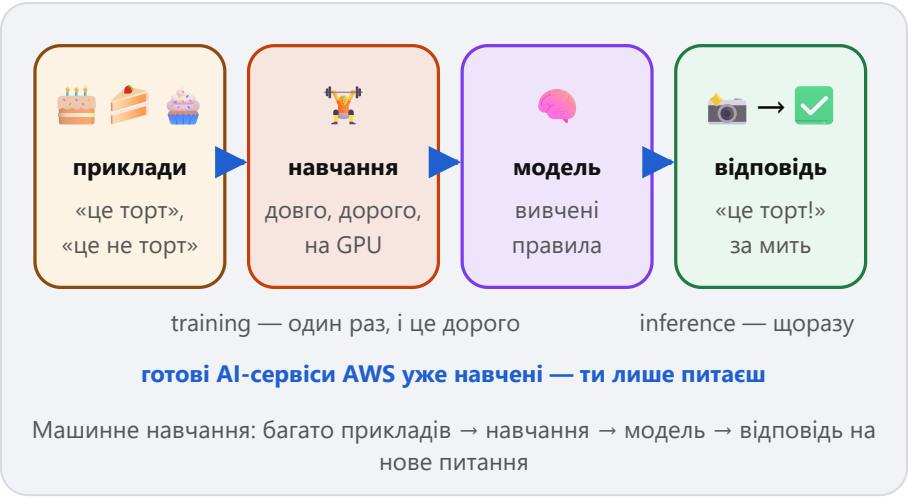
🥐 У «Хмаринки» назбиралися задачі, з якими не впорається жодна таблиця. Клієнти завантажують фото тортів у відгуки — треба відсіювати непристойні. Постачальники надсилають рахунки в PDF — їх хочуть читати автоматично. Сотні відгуків щотижня — а Олі цікаво, чи люди задоволені. Відео-рецепти без субтитрів, сайт лише українською, і клієнти хочуть чатбот для замовлень. «Невже треба наймати вчених?» — лякається Оля. «Ні, — сміється Тарас. — Більшість цього в AWS уже вміють готові "органи чуття"».

Машинне навчання — без магії

Машинне навчання (machine learning, ML) — коли комп'ютер не отримує готових правил, а **вчиться на прикладах**. Як дитина вчиться впізнавати котів: їй не пояснюють «чотири лапи, вуса, хвіст», а показують сотні котів — і згодом вона впізнає навіть незнайомого.

- **Модель** — «вивчені правила», результат навчання.
- **Навчання** (training) — довгий і дорогий процес: тисячі прикладів, потужні GPU.

- **Відповідь моделі** (inference, «інференс») — коли готова модель відповідає на нове питання: «на фото — торт». Це швидко й дешево за один запит.



Машинне навчання: багато прикладів → навчання → модель → відповідь на нове питання

А **генеративний AI** — моделі, що не лише впізнають, а й **створюють**: тексти, зображення, відповіді в чаті. Їх навчають на величезних обсягах даних, і називають **базовими моделями** (foundation models).

Для іспиту головна думка проста: **стандартну задачу** (розпізнати, перекласти, озвучити) вирішують **готові сервіси**, де не треба знати ML — просто викликаєш API. **Власна, унікальна задача** зі своїми даними — **SageMaker AI**.

Органи чуття для застосунку

Образ з простого курсу — **набір готових органів чуття**:



- **Amazon Rekognition** («рекогнішн») — **очі**: об'єкти й обличчя на фото і відео, текст на зображеннях, **модерація** неприйнятного вмісту.
- **Amazon Textract** («текстракт») — **бухгалтер, що читає документи**: текст, **форми й таблиці** зі сканів і PDF. Більше, ніж звичайне розпізнавання тексту (OCR): розуміє, де «Сума», а де «Дата».

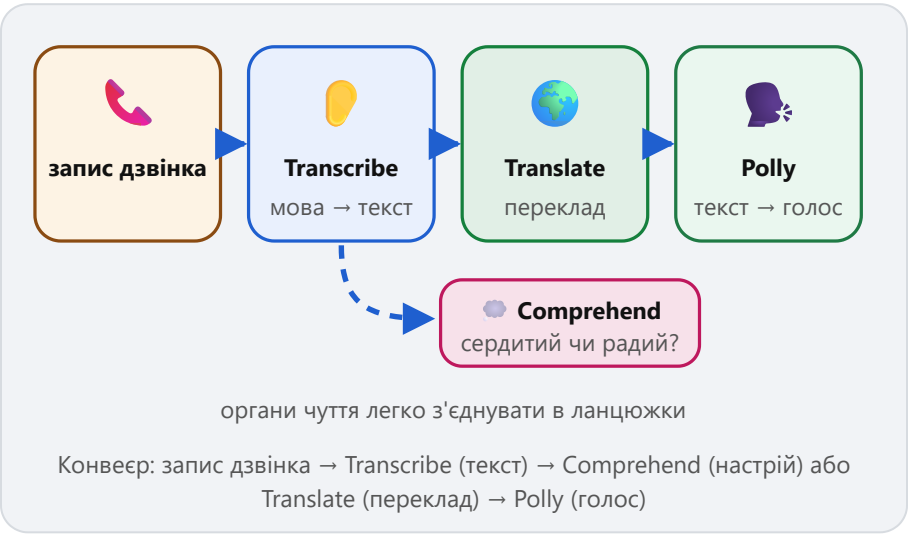
- **Amazon Comprehend** («компріхенд») — **читач, що розуміє**: **тональність** (задоволений клієнт чи ні), ключові фрази, мова, імена й місця, **персональні дані** в тексті.
- **Amazon Transcribe** («транскрайб») — **вуха**: **мова → текст** (субтитри, розшифровки дзвінків), уміє приховувати персональні дані.
- **Amazon Polly** («поллі») — **голос**: **текст → мова** (озвучення). Мнемоніка: **папуга Поллі говорить**.
- **Amazon Translate** — **перекладач**: машинний переклад текстів.
- **Amazon Lex** («лекс») — **співрозмовник**: **чатботи** голосом і текстом.
- **Amazon Kendra** («кендра») — **бібліотекар документів компанії**: розумний пошук по внутрішніх документах питаннями звичайною мовою.

⚠️ Не плутай: **Transcribe** — мова → текст, **Polly** — текст → мова. **Textract** — документи й форми, **Rekognition** — фото й відео.

💡 На іспиті можуть згадуватися й **Forecast** (прогнози за історією) та **Fraud Detector** (підозрілі транзакції) — знай, для чого вони. Але частину старих ML-сервісів AWS уже закрив для нових клієнтів, тож у нових проєктах таке зазвичай роблять через SageMaker AI.

Конвеєри: органи чуття разом

Сервіси легко з'єднувати в ланцюжки. Улюблений на іспиті:



органи чуття легко з'єднувати в ланцюжки

Конвеєр: запис дзвінка → Transcribe (текст) → Comprehend (настрій) або Translate (переклад) → Polly (голос)

- запис дзвінка клієнта → **Transcribe** (у текст) → **Comprehend** (сердитий чи задоволений?);
- текст українською → **Translate** (польською) → **Polly** (озвучити польською).

SageMaker AI — лабораторія для власного мозку

Коли готового «органу» для задачі немає — наприклад, треба **передбачати попит на круасани** в кожній пекарні за погодою, днем тижня й святами, — потрібна **власна модель**. **Amazon SageMaker AI** («сейджмейкер», колишня назва — SageMaker) — **лабораторія, де вирощують власний «мозок»**:

- підготувати дані;
- **навчити** модель і підібрати налаштування;
- **розгорнути** її як точку, куди застосунок надсилає питання;
- стежити, щоб модель не «старіла».

Bedrock — готові генеративні моделі

Amazon Bedrock («бедрок») — доступ через API до **готових генеративних моделей** від різних компаній: писати тексти, відповідати в чаті, підсумовувати документи, створювати зображення. Моделі не треба навчати — достатньо дати інструкції, а за потреби підключити власні документи як базу знань.

- Стандартне «бачити, чути, читати» → **готові AI-сервіси**.
- Власна модель на власних даних → **SageMaker AI**.

- Генерувати текст і вести розмову на основі готових великих моделей → **Bedrock**.

> 💉 **Спробуй у справжньому AWS: Поговори з генеративною моделлю в Bedrock (5 хвилин, за бажанням)**
>
> Вправа необов'язкова. На безкоштовному плані знайомство з Bedrock ще й додає кредити, але доступні моделі залежать від регіону, тож якщо якоїсь немає — просто обери іншу.
>
> 1. У консолі відкрий **Amazon Bedrock** → **Playgrounds** → **Chat** (або Text).
> 2. Обери будь-яку доступну модель.
> 3. Напиши: «Придумай три назви для нового торта з вишнею і шоколадом для пекарні "Хмаринка"».
> 4. Подивися на відповідь і на кількість **токенів** — шматочків тексту, приблизно частин слів. Саме за токени платять за генеративні моделі: і за ті, що надіслано, і за ті, що модель написала у відповідь.
> 5. Нічого видаляти не потрібно: Playground не створює ресурсів, які коштували б гроші.

Для фронтенду

Ще два сервіси з того самого розділу іспиту:

- **AWS Amplify** («ампліфай») — хостинг і готовий бекенд для **веб- і мобільних застосунків**, з автоматичним розгортанням при кожній зміні коду.
- **AWS Device Farm** — тестування застосунку на **сотнях справжніх телефонів і планшетів** у хмарі.

> 🥐 Тепер фото у відгуках «Хмаринки» перевіряє Rekognition, рахунки постачальників читає Textract, Comprehend щопонеділка показує Олі, скільки відгуків були радісними, відео-рецепти мають субтитри від Transcribe, сайт перекладено польською через Translate, а замовлення приймає чатбот на Lex. Попит на круасани прогнозує власна модель у SageMaker AI — і пекарні більше не викидають зайвого.

📖 **Словничок**

Машинне навчання (ML) — навчання комп'ютера на прикладах.

Модель / навчання / inference — вивчені правила / процес навчання / відповідь готової моделі.

Генеративний AI / foundation model — моделі, що створюють текст і зображення / великі базові моделі.

Rekognition — фото й відео, обличчя, модерація. **Textract** — текст, форми й таблиці з документів.

Comprehend — тональність, сутності, персональні дані в тексті.

Transcribe — мова → текст. **Polly** — текст → мова. **Translate** — переклад.

Lex — чатботи. **Kendra** — розумний пошук по документах компанії.

SageMaker AI — побудова, навчання й розгортання власних моделей.

Bedrock («бедрок») — готові генеративні моделі через API.

Amplify / Device Farm — веб- і мобільні застосунки під ключ / тестування на справжніх пристроях.

🧩 **Міні-тест: перевір себе**

? Потрібно автоматично відсіювати неприйнятні фото у відгуках. Що обрати?

- **А** Amazon Rekognition
- **Б** Amazon Textract
- **В** Amazon Polly
- **Г** Amazon Translate

Відповідь: А. Rekognition — «очі»: розпізнавання й модерація зображень і відео.

? Треба автоматично витягати суми й дати з рахунків постачальників у PDF. Що обрати?

- **А** Rekognition
- **Б** Amazon Textract
- **В** Amazon Lex
- **Г** Amazon Kendra

Відповідь: Б. Textract розуміє форми й таблиці в документах, а не лише окремі літери.

? Оля хоче знати, чи задоволені клієнти, — за текстом сотень відгуків. Що обрати?

- **А** Transcribe
- **Б** Amazon Comprehend
- **В** Polly
- **Г** SageMaker AI з нуля

Відповідь: Б. Comprehend визначає тональність тексту без знань ML.

? Потрібно зробити субтитри до відео-рецептів. Що обрати?

- **А** Amazon Transcribe
- **Б** Amazon Polly
- **В** Amazon Rekognition
- **Г** Amazon Translate

Відповідь: А. Transcribe перетворює мову на текст.

? Потрібно озвучити тексти рецептів. Що обрати?

А Transcribe

Б Amazon Polly

В Comprehend

Г Lex

Відповідь: Б. Папуга Поллі говорить: текст → мова.

? Потрібен чатбот, який приймає замовлення текстом і голосом. Що обрати?

А Amazon Kendra

Б Amazon Lex

В Amazon Textract

Г Amazon Polly

Відповідь: Б. Lex — співрозмовник: чатботи голосом і текстом.

? Потрібна власна модель, що прогнозує попит на круасани за погодою й святами на даних пекарні. Що обрати?

- **А** Rekognition
- **Б** Translate
- **В** Amazon SageMaker AI
- **Г** Amplify

Відповідь: В. Унікальна задача на власних даних — власна модель у SageMaker AI.

? Записи дзвінків треба перетворити на текст, перекласти й озвучити іншою мовою. Який ланцюжок?

- **А** Transcribe → Translate → Polly
- **Б** Polly → Translate → Transcribe
- **В** Textract → Rekognition → Lex
- **Г** Comprehend → Kendra → Amplify

Відповідь: А. Мова → текст (Transcribe), переклад (Translate), текст → мова (Polly).

✅ **Головне**

ML — навчання на прикладах: **модель**, **навчання** (довго й дорого), **inference** (швидка відповідь). Генеративний AI — створює текст і зображення.

Стандартні задачі — **готові сервіси без знань ML**: **Rekognition** (очі), **Textract** (документи), **Comprehend** (розуміння тексту), **Transcribe** (вуха), **Polly** (голос), **Translate**, **Lex** (чатботи), **Kendra** (пошук по документах).

Конвеєр іспиту: **Transcribe** → **Comprehend / Translate** → **Polly**.

Власна модель на власних даних → **SageMaker AI**; готові генеративні моделі → **Bedrock**.

Фронтенд: **Amplify** (застосунки під ключ), **Device Farm** (тести на справжніх телефонах).

📈 Приладова панель, відеореєстратор і пульт: моніторинг і автоматизація

👴 Простими словами

CloudWatch — приладова панель автомобіля: показує стан і бібикає, коли щось не так. CloudTrail — відеореєстратор: записує, хто що робив. Config — фотоархів того, як усе було налаштовано.

🎯 Що ти дізнаєшся

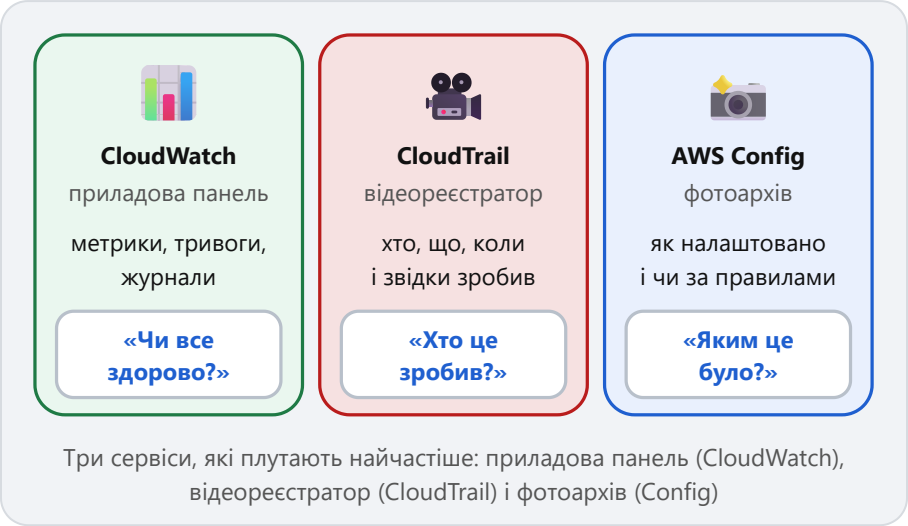
- як CloudWatch стежить за здоров'ям системи — метрики, тривоги, журнали і чому пам'ять видно лише з агентом
- чим CloudTrail відрізняється від CloudWatch і Config — трійця, яку плутають найчастіше
- як X-Ray знаходить, де гальмує запит, а EventBridge миттєво реагує на події
- як Systems Manager керує сотнями серверів без SSH
- що таке інфраструктура як код — CloudFormation, стеки, change sets, drift і StackSets

- які порадники підказують, де заощадити й що виправити

🥐 «Хмаринка» тепер — це десятки серверів у кількох акаунтах, бази, черги й функції. Одного суботнього ранку сайт почав відповідати по п'ять секунд — і Тарас дізнався про це не від системи, а з гнівних коментарів в Instagram. Того ж тижня хтось видалив правило в Security Group, і ніхто не міг сказати, хто й коли. А ще сорок серверів треба було оновити, і Тарас дві ночі заходив на кожен по черзі. «Ми їдемо вночі без фар і з заклеєною приладовою панеллю», — зітхає він.

Три питання чергового

Уяви, що вся інфраструктура «Хмаринки» — це автомобіль. Щоб їхати безпечно, водієві потрібні три речі, і в AWS для кожної є свій сервіс:



Три сервіси, які плутають найчастіше: приладова панель (CloudWatch), відеореєстратор (CloudTrail) і фотоархів (Config)

- **«Чи все здорово просто зараз?» — приладова панель**: спідометр, датчики, лампочки. Це **Amazon CloudWatch**.
- **«Хто це зробив?» — відеореєстратор**, що записує кожну дію водія. Це **AWS CloudTrail**.
- **«Яким це було вчора і чи відповідає правилам?» — фотоархів** і техогляд. Це **AWS Config**.

У частині про безпеку ця трійця вже коротко траплялася. Тепер розберемо кожен сервіс детально — і додамо до них помічників.

CloudWatch: метрики — стрілки на приладах

Amazon CloudWatch («клауд вотч», дослівно «спостерігати за хмарою») — головний сервіс

моніторингу AWS.

Метрика — число, яке вимірюють знову і знову: завантаження процесора у відсотках, кількість запитів, помилок, переданих байтів. Як спідометр, що показує швидкість щомиті, — тільки CloudWatch ще й пам'ятає всю історію і малює графіки.

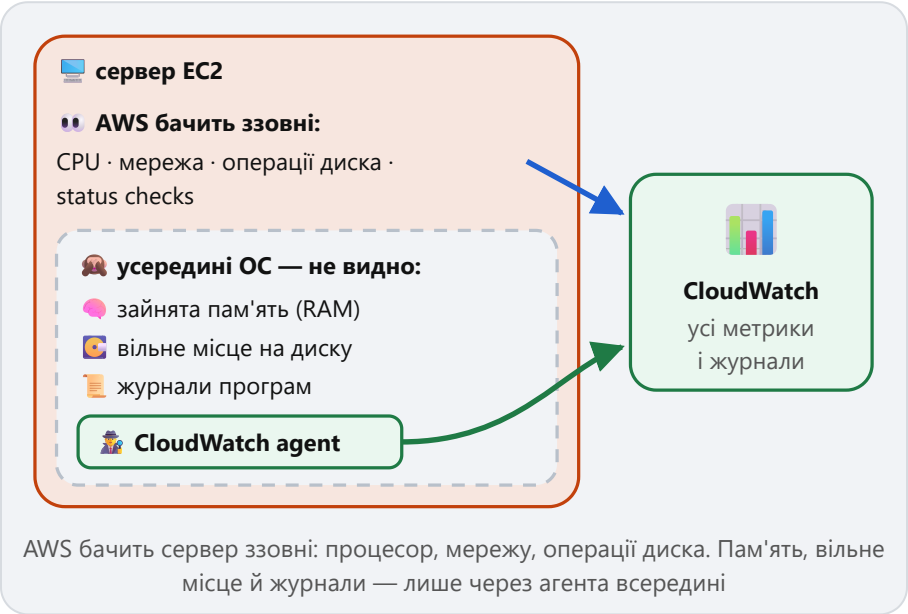
- Для **EC2** метрики є одразу, без налаштувань: процесор, мережа, операції з диском, **перевірки стану** (status checks). За замовчуванням — **раз на 5 хвилин**; **detailed monitoring** («детальний моніторинг») — **щохвилини**, за доплату.
- Майже кожен сервіс AWS сам надсилає свої метрики: балансувальник — кількість запитів і помилок, RDS — з'єднання з базою, SQS — довжину черги, Lambda — тривалість і помилки.
- **Власні метрики** (custom metrics) — будь-яке число твого застосунку: «замовлень за хвилину», «тортів у кошиках».
- **Дашборди** (dashboards) — екрани з графіками. Їх можна вивісити на телевізор у пекарні.

Чому CloudWatch не бачить пам'ять

Ось одна з найулюбленіших пасток іспиту. Пам'ятаєш віртуалізацію? Твій сервер EC2 — віртуальна машина, а

AWS дивиться на неї **ззовні**, з боку гіпервізора. Звідти видно, скільки часу процесора вона забирає і скільки даних передає мережею. Але як операційна система **всередині** розпоряджається своєю пам'яттю — що зайнято, що вільно, — знає лише вона сама. Так само і з вільним місцем на диску.

Образ: механік бачить машину ззовні — скільки вона їздить і як швидко. А скільки пального в баку, покаже лише датчик, встановлений усередині. Таким датчиком є **CloudWatch agent** («агент») — невелика програма, яку встановлюють на сервер. Вона надсилає в CloudWatch пам'ять, вільне місце на диску й журнали програм.



AWS бачить сервер ззовні: процесор, мережу, операції диска. Пам'ять, вільне місце й журнали — лише через агента всередині

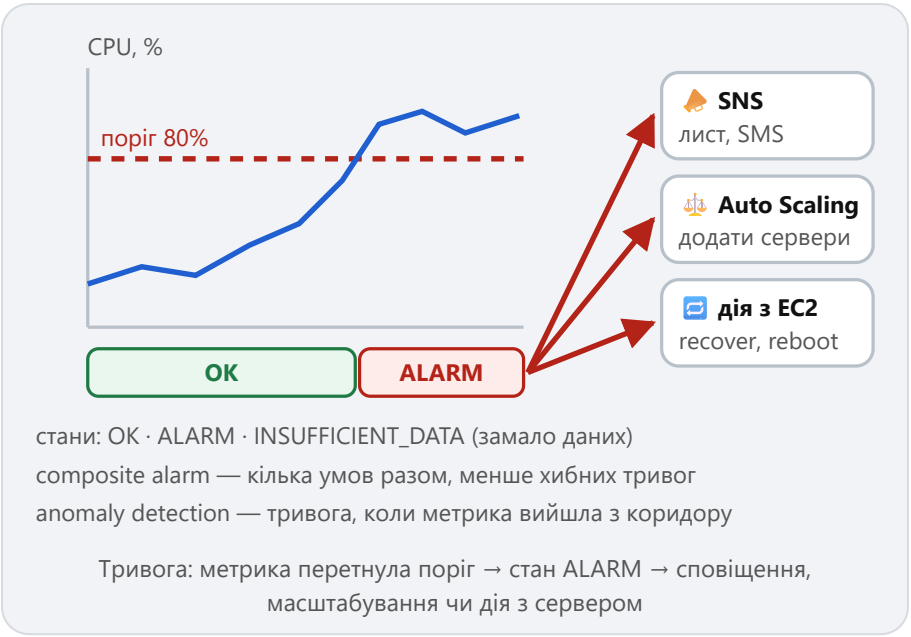
🎯 «Моніторити використання пам'яті (RAM) чи вільне місце на диску EC2» → **CloudWatch agent**. Пастка: «увімкнути detailed monitoring» не допоможе — він лише частіше надсилає ті самі метрики.

Тривоги — лампочки, що загоряються самі

Дивитися на графіки цілодобово ніхто не буде. Тому в CloudWatch є **тривоги** (alarms, «аларми»). Тривога стежить за однією метрикою і порогом: «якщо процесор понад 80% довше 5 хвилин…». У неї три стани: **OK**, **ALARM** (тривога) і **INSUFFICIENT_DATA** (замало даних, щоб вирішити).

Коли тривога спрацьовує, вона може:

- **сповістити людей** через SNS — гучномовець: лист, SMS;
- **запустити масштабування** — додати сервери в групу Auto Scaling (пам'ятаєш термостат?);
- **зробити дію з EC2** — зупинити, видалити, перезавантажити чи **відновити** (recover): перенести сервер на справне обладнання, зберігши його налаштування й адреси;
- запустити автоматизацію Systems Manager (про неї — нижче).



стани: OK · ALARM · INSUFFICIENT_DATA (замало даних)
composite alarm — кілька умов разом, менше хибних тривог
anomaly detection — тривога, коли метрика вийшла з коридору

Тривога: метрика перетнула поріг → стан ALARM → сповіщення, масштабування чи дія з сервером

Ще два корисні види тривог:

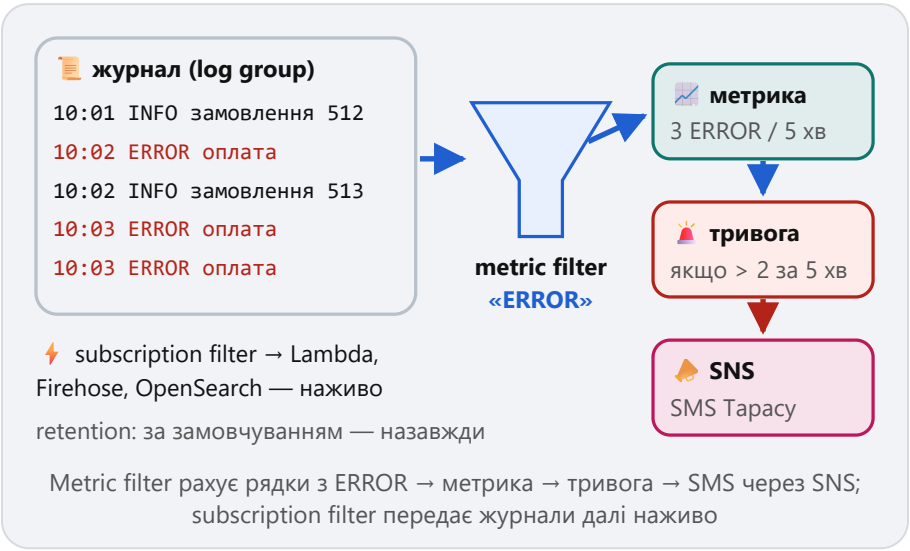
- **Composite alarm** («композитна») — поєднує кілька умов: «тривога, лише якщо високий процесор **і** багато помилок». Менше хибних нічних дзвінків.
- **Anomaly detection** («виявлення аномалій») — CloudWatch сам вивчає звичний «коридор» метрики. Ранковий пік замовлень у пекарні — норма, а такий самий пік о третій ночі — підозріло. Ручний поріг не потрібен.

🎯 «Сервер EC2 має сам відновитися, якщо зламалося обладнання AWS» → **тривога CloudWatch на StatusCheckFailed_System з дією recover**.

Журнали — бортовий щоденник

Програми постійно пишуть рядки про свою роботу: «10:01 замовлення 512 прийнято», «10:02 ERROR: оплата не пройшла». Це **журнали** (logs, «логи»). **CloudWatch Logs** збирає їх в одному місці: з агента на серверах, з Lambda (автоматично), з API Gateway, з VPC Flow Logs, з CloudTrail.

- Журнали складаються в **групи** (log groups). **Строк зберігання** (retention) налаштовують для кожної групи. За замовчуванням — **назавжди**, тож старі журнали тихо накопичуються в рахунку.
- **Metric filter** («фільтр метрик») — «рахуй рядки зі словом ERROR». Кількість стає метрикою, на метрику ставиться тривога, тривога шле SMS. Класика іспиту.
- **Logs Insights** — запити по журналах: «покажи 20 найповільніших запитів за останню годину».
- **Subscription filter** («фільтр підписки») — передавати журнали далі **в реальному часі**: у Lambda, Firehose, Kinesis, OpenSearch. А для архіву журнали можна експортувати в S3.



🎯 «Сповіщати, коли в журналах слово ERROR з'являється частіше за N разів» → **metric filter + тривога CloudWatch → SNS**.

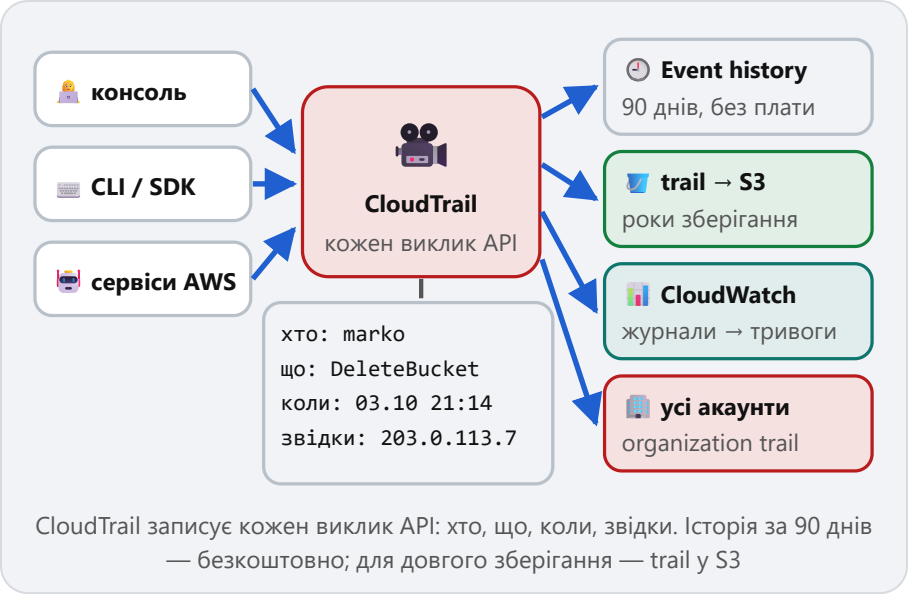
🔍 Для допитливих: Ще кілька приладів CloudWatch

- **CloudWatch Synthetics** — «таємний покупець». Робот-скрипт (canary, «канарка») кожні кілька хвилин відкриває сайт і проходить шлях клієнта: меню → кошик → оплата. Щось зламалося — ти дізнаєшся раніше за клієнтів. Назва — від канарок, яких шахтарі брали в шахту: пташка першою помічала небезпечний газ.

- **Container Insights** — метрики й журнали контейнерів ECS і EKS.
- **Amazon Managed Service for Prometheus** і **Amazon Managed Grafana** — популярні відкриті інструменти моніторингу (Prometheus збирає метрики, Grafana малює дашборди), але без власних серверів. Їх часто обирають для контейнерів у EKS.

CloudTrail — відеореєстратор кожної дії

Пригадай: будь-яка дія в AWS — клацання в консолі, команда в CLI, запит з програми чи дія самого сервісу — це **виклик API**. **AWS CloudTrail** («клауд трейл», «хмарний слід») записує кожен такий виклик: **хто** (який користувач чи роль), **що** (наприклад, `DeleteBucket` — видалити сховище), **коли**, **звідки** (IP-адреса) і чи вдалося.



CloudTrail записує кожен виклик API: хто, що, коли, звідки. Історія за 90 днів — безкоштовно; для довгого зберігання — trail у S3

- **Event history** («історія подій») — останні **90 днів** дій з ресурсами. Безкоштовно і вже ввімкнено.
- Щоб зберігати довше, створюють **trail** («трейл», «слід») — він пише журнал у **S3** (на роки), а за бажанням — і в **CloudWatch Logs**, щоб ставити тривоги на підозрілі дії.
- **Organization trail** — один trail одразу для **всіх акаунтів** організації. Разом з SCP «заборонити вимикати CloudTrail» — відповідальність, від якої не сховатися.
- **Management events** («події керування») — дії **з ресурсами**: створити сервер, змінити Security Group, видалити сховище. Записуються за замовчуванням.

- **Data events** («події даних») — дії **з даними всередині**: хто прочитав який файл у S3, хто викликав функцію Lambda. Їх мільйони, тому вони **вимкнені за замовчуванням** і платні. Образ: management events — хто зайшов у кабінет і переставив меблі; data events — хто відкрив яку папку в шафі.
- **CloudTrail Insights** — помічає незвичну активність: раптом сотні видалень за годину.
- **Log file integrity validation** — доказ, що журнали ніхто не підправив. Працює на хешах — «відбитках пальців» даних із розділу про шифрування.
- **CloudTrail Lake** — запити мовою SQL по подіях.

🎯 «Хто видалив ресурс і коли?» → **CloudTrail**. «Хто читає об'єкти в S3?» → **CloudTrail data events** (або S3 server access logs). «Довести, що журнали ніхто не змінював» → **log file integrity validation**.

Config — фотоархів і техогляд

AWS Config («конфіг», від configuration — налаштування) відповідає на питання «яким це було» і «чи за правилами»:

- **Інвентар** — які ресурси є і як вони налаштовані.

- **Історія конфігурацій** — щоразу, коли щось змінюється, Config ніби робить фото. «Яким було правило Security Group учора о 15:00?» — Config покаже зміну й підкаже подію CloudTrail, яка її спричинила.
- **Правила** (Config rules) — техогляд: «усі диски EBS зашифровані», «жодне сховище S3 не публічне», «SSH не відкритий для всього інтернету». Є готові правила і власні (на Lambda).
- **Автовиправлення** (remediation, «ремедіейшн») через **SSM Automation**: знайшов публічне сховище — сам його закрив.
- **Conformance packs** — готові набори правил (наприклад, під певний стандарт). **Aggregator** — одна картина з усіх акаунтів і регіонів.

⚠️ Config — **детектив, а не охоронець**: він помічає порушення, коли воно вже сталося, і може його виправити. **Заборонити** дію заздалегідь — справа IAM і SCP.

Трійця, яку плутають найчастіше

Питання	Сервіс	Образ
Сервер перевантажений? Скільки помилок? Скільки вільної пам'яті?	**CloudWatch**	приладова панель

Питання	Сервіс	Образ
Хто видалив сховище? Хто змінив Security Group і з якої адреси?	**CloudTrail**	відеореєстратор
Яким був ресурс тиждень тому? Чи всі диски зашифровані?	**Config**	фотоархів і техогляд

Підказка в самих назвах: **Watch** — «дивитися» на здоров'я; **Trail** — «слід», сліди дій; **Config** — налаштування.

X-Ray — рентген одного запиту

У мікросервісах одне натискання «Замовити торт» проходить через десяток сервісів: API Gateway, Lambda, базу, платіжну систему, чергу. Клієнт чекає 3 секунди — але де саме губиться час? **AWS X-Ray** («ікс-рей», рентген) **просвічує шлях кожного запиту**: скільки тривав кожен крок і де сталася помилка. А ще малює **мапу сервісів** (service map) — хто кого викликає.



X-Ray показує кожен крок одного запиту в часі: одразу видно, що гальмує платіжний сервіс

🎯 «Знайти, де виникає затримка чи помилка між мікросервісами» → **X-Ray**.

EventBridge — миттєва реакція на подію

EventBridge — диспетчерську подій — ти вже знаєш з розділу про повідомлення. В експлуатації вона працює як нервова система:

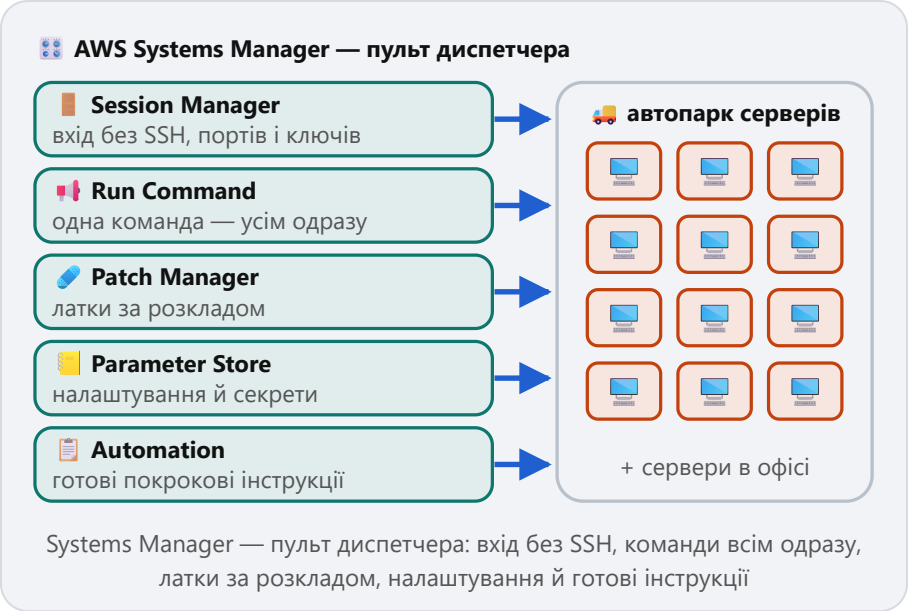
- CloudTrail записав «хтось змінив Security Group» → **правило EventBridge** → SMS Тарасу через SNS або функція Lambda, що повертає правило назад. За лічені секунди.

- AWS Health повідомив «плануємо обслуговування хоста, на якому працює твій сервер» → EventBridge → сповіщення команді.
- **EventBridge Scheduler** — дії за розкладом: «щодня о 19:00 зупиняти тестові сервери».

> 🎯 «Миттєве сповіщення, щойно хтось змінює Security Group» → **правило EventBridge на подію CloudTrail → SNS чи Lambda**.

Systems Manager — пульт керування автопарком

Уяви, що сорок серверів «Хмаринки» — це сорок фургонів доставки. Можна об'їжджати кожен з ключами, а можна посадити диспетчера з пультом. **AWS Systems Manager** («системс менеджер», скорочено SSM) — такий пульт:



Systems Manager — пульт диспетчера: вхід без SSH, команди всім одразу, латки за розкладом, налаштування й готові інструкції

- **Session Manager** («сешн менеджер») — вхід у консоль сервера просто з браузера: **без SSH**, без відкритого порту 22, без бастіону й ключів; кожну сесію можна записати. Він уже траплявся в частині про мережу.
- **Run Command** («ран команд») — **одна команда для всіх** серверів одразу: «онови програму каси на всіх сорока».
- **Patch Manager** («патч менеджер») + **Maintenance Windows** («вікна обслуговування») — **латки** (patches) операційної системи **за розкладом**: «щонеділі о третій ночі, по десять серверів за раз».
- **Parameter Store** — записник налаштувань і секретів, знайомий з розділу про шифрування.

- **Automation** («автомейшн») — готові покрокові інструкції (runbooks): «зробити знімок диска → оновити → перевірити».
- Ще є **Inventory** (що встановлено на кожному сервері) і **State Manager** (тримати потрібний стан).
- Працює і з серверами **в офісі чи власному дата-центрі** (hybrid activations).

Щоб сервер «чув» пульт, на ньому має працювати **SSM Agent** (в образах Amazon Linux і Windows від AWS він уже є), а сам сервер має отримати права на роботу з Systems Manager — зазвичай через **роль IAM**. Немає прав — сервера не видно в пульті.

🎯 «Виконати скрипт на сотнях серверів без SSH» → **Run Command**. «Патчити сервери за розкладом» → **Patch Manager + Maintenance Windows**. «Доступ до сервера без відкритих портів і бастіону» → **Session Manager**. Пам'ятаєш з частини про безпеку: **Inspector знаходить** вразливості, **Patch Manager латає**.

CloudFormation — креслення замість клацання

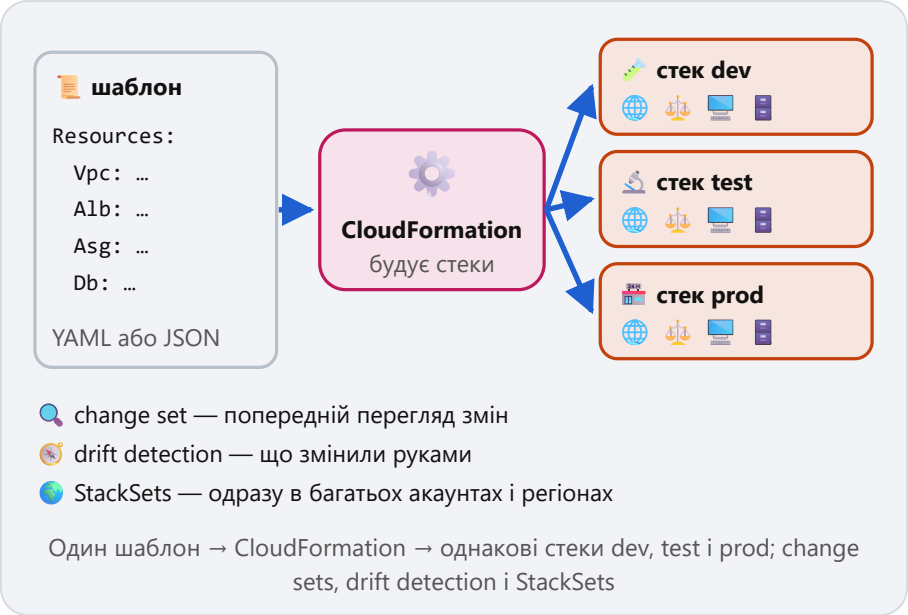
Тарас два дні клацав у консолі, щоб зібрати тестове середовище, — і воно однаково вийшло не таким, як робоче: одне правило забулося. Вихід —

інфраструктура як код (Infrastructure as Code, IaC): уся інфраструктура описується в текстовому файлі.

Образ з простого курсу — **креслення будинку**. Одне креслення — і будівельники зведуть однакові будинки будь-де.

AWS CloudFormation («клауд формейшн»):

- **Шаблон** (template) — текстовий файл у форматі **YAML** («ямл») чи JSON: «потрібна VPC з такими підмережами, балансувальник, група автомасштабування й база RDS». JSON — той самий «бланк замовлення» з розділу про API, а YAML — його спрощений родич: замість дужок і лапок вкладеність показують відступами, тож такий файл легко читає і людина, і програма.
- CloudFormation читає шаблон і будує **стек** (stack) — набір ресурсів, створених за цим шаблоном. Правильний порядок він знає сам: спершу мережа, потім підмережі, потім сервери.
- Видалив стек — зникло все, що він створив. Нічого не забудеться і не коштуватиме грошей «у куточку».
- Один шаблон — однакові середовища **dev, test і prod**. І копія в **іншому регіоні** за лічені хвилини — це знадобиться в наступному розділі, про аварії.
- Шаблон зберігають як код: видно історію змін, можна перевірити перед застосуванням і відкотитися.



- **Change set** («ченж сет») — **попередній перегляд**: що саме зміниться, якщо застосувати нову версію шаблону. Як кошторис перед ремонтом: видно, яку стіну зноситимуть.
- **Drift detection** («дрифт», дрейф) — виявляє **ручні зміни**, через які реальність «відпливла» від креслення.
- **StackSets** («стек сетс») — один шаблон одразу в **багатьох акаунтах і регіонах**.
- **DeletionPolicy** — що робити з ресурсом при видаленні стека: `Retain` (залишити, наприклад базу) чи `Snapshot` (спершу зробити знімок).

💡 Не плутай з **Elastic Beanstalk** з розділу про контейнери: Beanstalk — «ресторан під ключ» для одного застосунку, CloudFormation — креслення **будь-якої** інфраструктури.

🔍 **Для допитливих: А якщо писати не YAML, а звичайною мовою програмування?**

Є **AWS CDK** (Cloud Development Kit): ту саму інфраструктуру описують мовами програмування — TypeScript, Python, Java, — а CDK перетворює опис на шаблон CloudFormation. Для іспиту головне — розуміти саму ідею інфраструктури як коду і що вміє CloudFormation.

Service Catalog — вітрина затверджених рішень

AWS Service Catalog — каталог **затверджених** продуктів, тобто шаблонів CloudFormation. Розробник з обмеженими правами обирає з вітрини «тестове середовище» чи «сховище для фото» — і отримує саме те, що затвердили, без права створювати будь-що на власний розсуд. Як меню в їдальні: обирай будь-яку страву, але з меню.

🎯 «Користувачі мають самі запускати лише затверджені шаблони» → **Service Catalog**.

Порадники

- **AWS Trusted Advisor** («трастед едвайзер») — досвідчений механік, що оглядає машину: де переплачуєш (сервери без діла), що з безпекою (відкриті порти, root без MFA), продуктивністю, стійкістю, і чи не наближаєшся до **лімітів сервісів**. Повний набір перевірок — з планами підтримки **Business і вище**.
- **Service Quotas** — ті самі ліміти з розділу про акаунт: тут їх переглядають і просять підняти.
- **AWS Compute Optimizer** — рекомендації **правильного розміру** на основі машинного навчання для EC2, груп автомасштабування, дисків EBS, Lambda, ECS на Fargate і RDS: «сервер завантажений на 5% — візьми вдвічі менший».
- **AWS Health Dashboard** — новини AWS **саме про твої ресурси**: збій у твоїй зоні, планове обслуговування хоста твого сервера. Події можна передавати в EventBridge.
- **AWS License Manager** — облік ліцензій (Microsoft, Oracle, SAP), щоб не перевищити куплену кількість.

- **AWS Well-Architected Tool** — самоперевірка архітектури за шістьма стовпами — тим самим храмом з частини про хмару.

🎯 «Інстанси завеликі — потрібні рекомендації щодо розміру» → **Compute Optimizer**. «Чи наближаємося до лімітів сервісів?» → **Trusted Advisor / Service Quotas**. «Сповіщення про планове обслуговування AWS, що зачепить наші сервери» → **Health Dashboard + EventBridge**.

🧪 **Спробуй у справжньому AWS: Подивися запис «відеореєстратора» свого акаунта (5 хвилин, безкоштовно)**

Ця вправа нічого не створює і нічого не коштує — лише перегляд.

1. У консолі відкрий **CloudTrail** → **Event history**.
2. Перед тобою всі дії в акаунті за останні 90 днів — зокрема твої власні з попередніх вправ.
3. У фільтрі обери **Event name** і введи `RunInstances` (запуск сервера з вправи про EC2) або `CreateBucket` (сховище з вправи про S3). Якщо цих вправ не було, — `ConsoleLogin` (вхід у консоль).

4. Відкрий будь-яку подію: подивися, **хто** (User name), **коли** (Event time), **звідки** (Source IP address) і повний запис у форматі JSON.
5. Тепер відкрий **CloudWatch → Metrics → All metrics** і подивися, які сервіси вже надсилають метрики про твій акаунт.

🥐 Тепер Тарас дізнається про проблеми раніше за клієнтів. «Таємний покупець» Synthetics щохвилини проходить шлях замовлення круасана, тривоги шлють SMS, а дашборд висить на стіні в пекарні. Organization trail записує кожну дію в захищене сховище, а Config щоночі перевіряє, чи всі диски зашифровані. Сорок серверів самі отримують латки в неділю о третій ночі через Patch Manager, а нове середовище з'являється за 15 хвилин з одного шаблону CloudFormation. «Нарешті в нашої машини є фари й приладова панель», — усміхається Оля.

📖 **Словничок**

CloudWatch («клауд вотч») — моніторинг: метрики, тривоги, журнали, дашборди.

Метрика — число, яке вимірюють знову і знову (процесор, запити, помилки).

Detailed monitoring — метрики EC2 щохвилини замість раз на 5 хвилин.

CloudWatch agent — програма на сервері, що надсилає пам'ять, місце на диску й журнали.

Тривога (alarm) — стежить за метрикою; стани OK / ALARM / INSUFFICIENT_DATA; дії: SNS, Auto Scaling, EC2 (recover та інші).

Composite alarm / anomaly detection — кілька умов разом / тривога за «коридором» без ручного порогу.

CloudWatch Logs — журнали; **metric filter** — рахує рядки в журналах; **subscription filter** — передає журнали наживо; **Logs Insights** — запити.

CloudTrail («клауд трейл») — журнал викликів API: хто, що, коли, звідки. Event history — 90 днів; **trail** — у S3 надовго; **organization trail** — усі акаунти.

Management / data events — дії з ресурсами (за замовчуванням) / дії з даними (вмикають окремо, платні).

Config («конфіг») — інвентар, історія налаштувань, правила й автовиправлення.

X-Ray («ікс-рей») — трасування запитів через мікросервіси й мапа сервісів.

Systems Manager (SSM) — пульт серверів: Session Manager, Run Command, Patch Manager, Parameter Store, Automation.

Інфраструктура як код (IaC) — інфраструктура, описана в текстовому файлі.

CloudFormation — шаблон → **стек**; **change set** — перегляд змін; **drift** — ручні зміни; **StackSets** — багато акаунтів і регіонів.

Service Catalog — вітрина затверджених шаблонів.

Trusted Advisor / Compute Optimizer / Health Dashboard — поради (вартість, безпека, ліміти) / правильний розмір / події AWS про твої ресурси.

🧩 **Міні-тест: перевір себе**

? Потрібно стежити, скільки оперативної пам'яті використовують сервери EC2. Що зробити?

- **А** Встановити CloudWatch agent
- **Б** Увімкнути detailed monitoring
- **В** Увімкнути CloudTrail
- **Г** Налаштувати правило AWS Config

Відповідь: А. AWS бачить сервер ззовні й не знає, як ОС використовує пам'ять. Пам'ять і вільне місце на диску надсилає лише агент.

? Учора хтось видалив сховище S3. Як дізнатися, хто і коли?

А CloudWatch Logs Insights

Б AWS CloudTrail

В AWS X-Ray

Г Compute Optimizer

Відповідь: Б. CloudTrail — відеореєстратор: записує кожен виклик API — хто, що, коли й звідки.

? Усі диски EBS мають бути зашифровані, а порушення — виправлятися автоматично. Що обрати?

- **А** CloudTrail Insights
- **Б** Правило AWS Config з автовиправленням через SSM Automation
- **В** Тривога CloudWatch з anomaly detection
- **Г** Trusted Advisor

Відповідь: Б. Config перевіряє налаштування за правилами й уміє виправляти через SSM Automation.

? Треба отримувати SMS, коли в журналах застосунку слово ERROR з'являється понад 10 разів за 5 хвилин. Що налаштувати?

А CloudTrail data events

Б Правило AWS Config

В Metric filter у CloudWatch Logs → тривога → SNS

Г AWS X-Ray

Відповідь: В. Metric filter рахує рядки в журналі, з кількості виходить метрика, тривога на ній шле SMS через SNS.

? Запит проходить через вісім мікросервісів і інколи відповідає 3 секунди. Як знайти, де гальмує?

А CloudTrail

Б AWS Config

В AWS X-Ray

Г Trusted Advisor

Відповідь: В. X-Ray трасує кожен запит через усі сервіси й показує, скільки часу забрав кожен крок.

? Потрібно виконати скрипт на 300 серверах, не відкриваючи SSH. Що обрати?

- **А** Зайти на кожен через Session Manager
- **Б** SSM Run Command
- **В** CloudFormation drift detection
- **Г** Сервер-бастіон

Відповідь: Б. Run Command виконує команду на всіх потрібних серверах одразу, без SSH і відкритих портів.

? Однакову інфраструктуру треба розгорнути у 20 акаунтах і 3 регіонах. Що обрати?

- **А** CloudFormation change sets
- **Б** CloudFormation StackSets
- **В** AWS Service Catalog
- **Г** Elastic Beanstalk

Відповідь: Б. StackSets розгортає один шаблон одразу в багатьох акаунтах і регіонах.

? Сервер EC2 має сам відновитися, якщо зламалося обладнання AWS під ним. Що налаштувати?

- **А** Тривогу CloudWatch на StatusCheckFailed_System з дією recover
- **Б** Правило AWS Config
- **В** Подію CloudTrail Insights
- **Г** Detailed monitoring

Відповідь: А. Дія recover переносить сервер на справне обладнання, зберігаючи його налаштування й адреси.

? Команда безпеки хоче миттєве сповіщення, щойно хтось змінює будь-яку Security Group. Що обрати?

А Щоденний звіт Trusted Advisor

Б Правило EventBridge на подію CloudTrail → SNS

В Ручний перегляд Event history

Г Detailed monitoring

Відповідь: Б. CloudTrail фіксує виклик API, EventBridge за секунди реагує на нього й надсилає сповіщення.

✅ **Головне**

CloudWatch — приладова панель: **метрики** (EC2 — раз на 5 хв, detailed — щохвилини), **тривоги** (SNS, Auto Scaling, recover), **журнали** (metric filter, subscription filter, retention — за замовчуванням назавжди).

Пам'ять і місце на диску EC2 → лише через **CloudWatch agent**.

CloudTrail — хто що зробив (API): 90 днів безкоштовно, trail → S3, **organization trail**; data events вмикають окремо; integrity validation — доказ незмінності.

Config — яким був ресурс і чи за правилами; **автовиправлення через SSM Automation**; помічає, а не забороняє.

X-Ray — де гальмує запит; **EventBridge** — миттєва реакція на події й розклад.

Systems Manager: **Session Manager** (без SSH), **Run Command** (усім одразу), **Patch Manager** (латки за розкладом), Parameter Store, Automation.

CloudFormation — інфраструктура як код: шаблон → стек; **change sets**, **drift detection**, **StackSets**. **Service Catalog** — лише затверджені шаблони.

Порадники: **Trusted Advisor** (вартість, безпека, ліміти), **Compute Optimizer** (правильний розмір), **Health Dashboard** (події AWS про твої ресурси).

🛟 Коли все ламається: висока доступність і план на випадок аварії

👴 Простими словами

Усе колись ламається, тож план має бути наперед. Чим швидше треба відновитися і чим менше даних можна втратити — тим дорожчий запасний план: від резервних копій до другого працюючого регіону.

🎯 Що ти дізнаєшся

- що таке єдина точка відмови і як її знайти
- як будують високу доступність в одному регіоні — і як зробити надійним старий застосунок без змін коду
- як повтори з паузами й ідемпотентність рятують від хвилі помилок
- чотири стратегії аварійного відновлення — від бекапів до двох активних регіонів — і як обрати за RPO та RTO
- з яких цеглинок збирають резервний регіон
- як AWS Backup і Vault Lock захищають копії навіть від адміністратора

🥐 Оля читає новини: у Європі згорів дата-центр одного хмарного провайдера, і частина його клієнтів утратила дані назавжди — копії лежали в тій самій будівлі. «А якщо таке станеться з нашим Франкфуртом?» — питає вона Тараса. «Від пожежі в одній зоні нас уже захищають дві інші, — відповідає він. — Але якщо зникне цілий регіон, потрібен окремий план. І спершу треба вирішити, скільки замовлень ми готові втратити й скільки часу можемо стояти. Від цього залежить і план, і його ціна».

Усе колись ламається

Технічний директор Amazon Вернер Фогельс любить повторювати: «Everything fails, all the time» — усе ламається, і постійно. Це не песимізм, а інженерна чесність: диски зношуються, дата-центри втрачають живлення, люди помиляються. Хороша архітектура **заздалегідь припускає поломку** — і переживає її. Пригадай три поняття з розділу про стовпи:

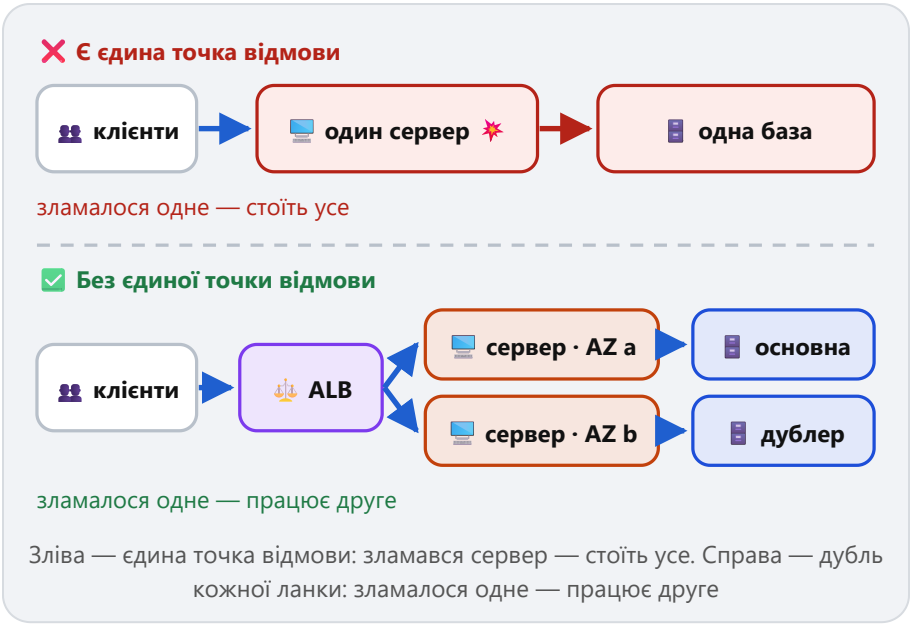
- **висока доступність** (HA) — **запасне колесо**: зламалося — коротка зупинка, і їдемо далі;

- **відмовостійкість** (fault tolerance) — **двомоторний літак**: один мотор відмовив, а політ триває без перерви;
- **аварійне відновлення** (disaster recovery, DR) — **план на випадок пожежі в цілому місті**, тобто регіоні.

І два числа: **RPO** — скільки **даних** можна втратити, **RTO** — скільки **часу** можна простояти.

Єдина точка відмови

Єдина точка відмови (single point of failure, SPOF) — деталь, з поломкою якої зупиняється все. У пекарні це одна піч, один ключ від дверей, одна касирка, яку нікому замінити. В інфраструктурі — один сервер, одна база, одна зона, один класичний NAT Gateway на всю мережу, один регіон, а ще одна людина, яка єдина знає пароль.



Зліва — єдина точка відмови: зламався сервер — стоїть усе. Справа — дубль кожної ланки: зламалося одне — працює друге

Головна вправа архітектора: подумки «ламати» кожен елемент по черзі й питати: **«Що буде, якщо зламається саме це?»** Якщо відповідь «стане все» — це єдина точка відмови, і її треба продублювати.

Що зламалося	Що рятує
Один сервер	Кілька серверів за балансувальником + Auto Scaling
Одна зона (AZ)	Усе щонайменше у двох AZ: ALB, ASG, Multi-AZ база, NAT у кожній AZ
Увесь регіон	Резервний регіон — одна зі стратегій DR (нижче)
Хтось видалив чи зашифрував дані	Бекапи з історією, в окремому акаунті, під Vault Lock
Збій одного компонента валить інші	Черги SQS між компонентами — слабке зв'язування

Висока доступність в одному регіоні

Збери докупи цеглинки з попередніх розділів — це саме те, що іспит називає «highly available architecture»:

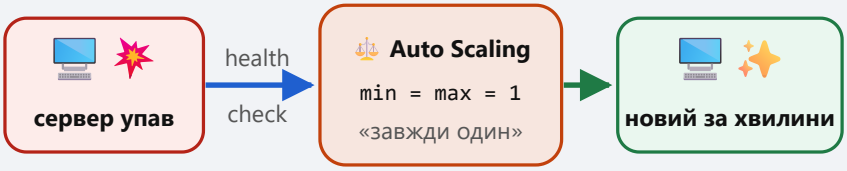
- **ALB + Auto Scaling щонайменше у двох AZ** — сервери в кількох районах міста, а балансувальник веде клієнтів лише до здорових.
- **Multi-AZ база** (RDS чи Aurora) — дублер в іншій зоні, перемикання за хвилину-дві.
- **Сервери без стану** (stateless): кошики й сесії лежать в ElastiCache чи DynamoDB, тож будь-який сервер може зникнути без втрат.
- **Статика в S3 + CloudFront** — фото й сторінки не залежать від серверів.
- **NAT Gateway у кожній AZ**. Класичний NAT Gateway живе в одній зоні: одна така NAT на всю мережу дешевша, але це єдина точка відмови — годиться хіба для тестів. (Новий Regional NAT Gateway сам розширюється на потрібні зони.)
- **Черги SQS між шарами**: якщо кухня тимчасово «впала», замовлення чекають у черзі, а не губляться.

⚠️ Multi-AZ рятує від збою **зони**, але не від збою **регіону**. Для регіону потрібна копія в іншому регіоні.

Старий застосунок, який не можна переписати

Часта ситуація на іспиті: стара програма працює на одному сервері, а змінювати її код не можна. Надійність додають «ззовні»:

- **Auto Scaling group з min = max = 1** — як кадрова агенція, яка гарантує, що на кухні завжди рівно один кухар. Зламався сервер — health check це помітить, і група за кілька хвилин створить новий. Щоб клієнти знаходили новий сервер, його ставлять за балансувальник.
- **EFS** для спільних файлів — новий сервер побачить ті самі файли.
- **Multi-AZ RDS** для бази, а **RDS Proxy** — якщо застосунок відкриває забагато з'єднань.
- **Route 53 health checks** — перевіряти, чи живий сайт, і перемикати адресу.
- Або тривога CloudWatch з дією **recover** з попереднього розділу — якщо зламалося обладнання під сервером.



старий застосунок без змін коду — сам «воскресає» після збою

Auto Scaling group з min = max = 1: сервер упав — health check помітив — група створила новий

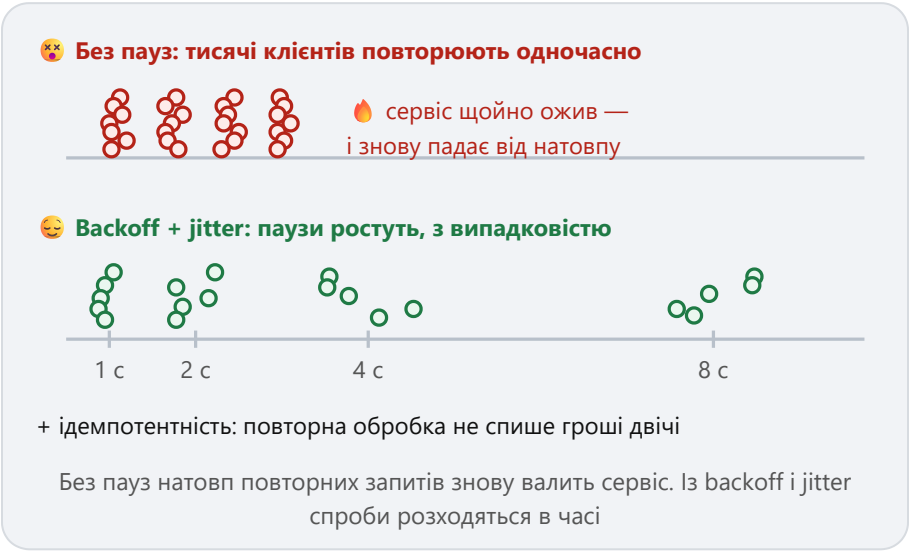
🎯 «Застосунок на одному EC2 має сам відновлюватися після збою, без змін коду» → **Auto Scaling group з min = max = 1** (+ health check балансувальника).

Стійкий код: повтори з паузами й ідемпотентність

Навіть у хорошій архітектурі трапляються короткі збої: сервіс перезапускається, мережа на мить «кашляє». Код має бути до цього готовий:

- **Повтори** (retries) — не вдалося, спробуй ще раз.
- **Exponential backoff** («експоненційна затримка») — паузи між спробами ростуть: 1 с, 2 с, 4 с, 8 с… Як дзвонити на зайняту лінію: не набирати щосекунди, а щоразу чекати довше.

- **Jitter** («джитер», дослівно «тремтіння») — до паузи додають трохи випадковості, щоб тисячі клієнтів не повторювали в ту саму мить. Як магазин, куди покупці приходять не всі рівно о 9:00:00, а розтягнуто.



- **Ідемпотентність** (idempotency, «ідемпотентність») — повторна обробка дає той самий результат, що й перша. Оплату за замовлення № 512 не можна списати двічі, навіть якщо повідомлення про неї прийшло двічі. А звичайна черга SQS справді **може** доставити повідомлення повторно. Рецепт простий: запам'ятовувати номери вже оброблених замовлень і пропускати дублікати.

Повтори з backoff уже вміють бібліотеки AWS (SDK), а от ідемпотентність — завдання коду застосунку.

Аварійне відновлення: чотири стратегії

Тепер — пожежа в цілому місті. Питання не в тому, чи потрібен план, а в тому, **скільки ми готові платити за швидкість відновлення**. Стратегій чотири — від дешевої й повільної до дорогої й миттєвої:



Чотири стратегії DR: що є в резервному регіоні, які RPO і RTO — і скільки це коштує

1. **Backup & Restore** («бекап енд рістор») — **фото документів у сейфі в іншому місті**. У резервному регіоні лежать лише копії: знімки дисків і баз, копії в S3, бекапи AWS Backup. Після аварії інфраструктуру будують заново з шаблонів CloudFormation і відновлюють дані. **RPO і RTO — години**. Найдешевше: платиш лише за зберігання копій.
2. **Pilot Light** («пайлот лайт») — **запальник газового котла**: маленький вогник горить завжди, тому котел розпалюється швидко. У резервному регіоні постійно працює лише **ядро — база даних, що безперервно реплікується**. Серверів немає або вони вимкнені, але готові образи AMI лежать поруч. Під час аварії запускають сервери, збільшують їх кількість і перемикають DNS. **Десятки хвилин**.
3. **Warm Standby** («ворм стендбай») — **прогріта запасна кухня**: усе обладнання ввімкнене, і працюють один-два кухарі. У резервному регіоні постійно працює **зменшена, але повна копія** всього: база, сервери, балансувальник. Під час аварії її **збільшують** до повного розміру й перемикають туди клієнтів. **Хвилини**.

4. **Multi-site Active-Active** («мультисайт актив-актив») — **два ресторани працюють одночасно**: обидва регіони весь час обслуговують клієнтів. Зник один — другий просто приймає всіх. **RPO і RTO — майже нуль**. Найдорожче: платиш за дві повні системи.

Найчастіше плутають другу й третю. У **Pilot Light** працює **лише база**, а сервери вимкнені. У **Warm Standby** працює **все**, просто в зменшеному розмірі.

💡 Правило вибору на іспиті: **найдешевша стратегія, що вкладається в RPO і RTO з питання**. «RTO — 4 години, мінімальна вартість» → Backup & Restore. «RTO — 30 хвилин, база має реплікуватися» → Pilot Light. «RTO — кілька хвилин» → Warm Standby. «Без простою, трафік в обох регіонах» → Active-Active.

Цеглинки резервного регіону



Резервний регіон збирають з цеглинок: перемикач трафіку, копії баз, файлів, образів і шаблон інфраструктури

- **Перемкнути клієнтів**: **Route 53** з політикою failover — переадресує, щойно health check основного регіону не проходить, — або **Global Accelerator**, що перемикає за секунди і не залежить від кешу DNS.
- **Реляційна база**: **Aurora Global Database** — RPO ≈ 1 с, RTO < 1 хв. Дешевший варіант — **read replica RDS в іншому регіоні**, але під час аварії її треба вручну «підвищити» до основної (promote), і останні зміни можуть загубитися: копіювання асинхронне.

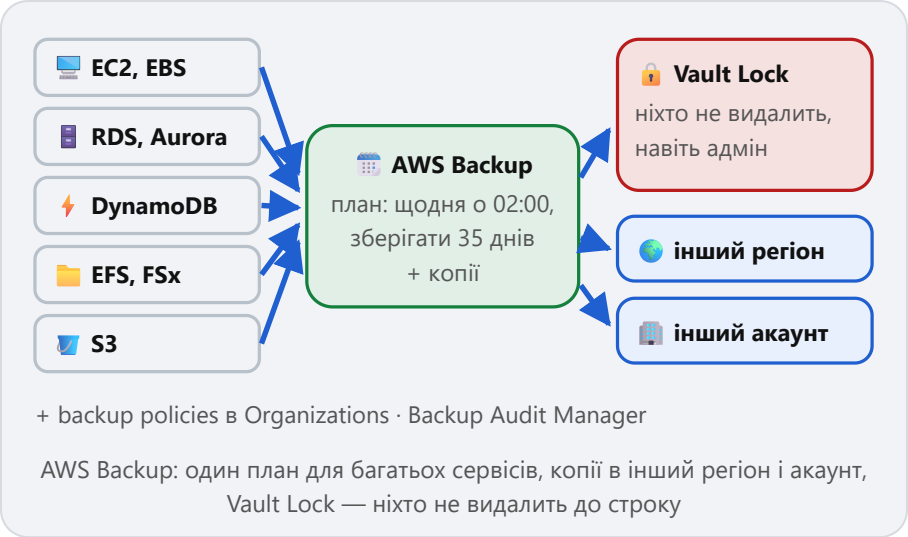
- **NoSQL**: **DynamoDB Global Tables** — записувати можна в будь-якому регіоні.
- **Файли**: **S3 Cross-Region Replication** (CRR); з **Replication Time Control** — 99,99% об'єктів за 15 хвилин.
- **Сервери**: копії образів **AMI** і знімків **EBS** у резервному регіоні.
- **Інфраструктура**: шаблони **CloudFormation** — щоб збудувати все за хвилини.
- **Квоти**: **Service Quotas** у резервному регіоні треба підняти **заздалегідь**. Інакше під час аварії AWS просто не дасть запустити потрібну кількість серверів.

⚠️ **Реплікація — не бекап.** Якщо хтось помилково видалить чи зіпсує дані, репліка слухняно повторить і це. Від помилок людей і вірусів-вимагачів рятують лише **бекапи з історією**.

AWS Backup — сейф для всіх копій

AWS Backup — одне місце для бекапів майже всього: EC2 і EBS, RDS і Aurora, DynamoDB, EFS, FSx, S3, Storage Gateway, DocumentDB, Neptune та інших сервісів.

- **План бекапів** (backup plan): коли робити копію (щодня о 02:00), скільки зберігати (35 днів) і куди копіювати.
- **Копії в інший регіон і в інший акаунт**. Навіть якщо зловмисник захопить основний акаунт, копії в іншому вціліють.
- **Backup Vault Lock** («волт лок») — сейф з часовим замком, режим WORM («записав один раз — читай скільки завгодно»): до кінця строку бекап **не видалить ніхто**, навіть адміністратор чи root. Це головний захист від **вірусів-вимагачів** (ransomware), які першим ділом намагаються знищити копії.
- **Backup policies** в Organizations — однакові правила бекапів для всіх акаунтів; **Backup Audit Manager** — звіти, чи всі ресурси справді захищені.



Ще один помічник — **AWS Elastic Disaster Recovery** (DRS) з розділу про переїзд: він безперервно копіює цілі сервери з офісу чи іншого регіону в AWS. RPO — секунди, RTO — хвилини.

🎯 «Централізовані бекапи багатьох сервісів з копією в інший акаунт і регіон» → **AWS Backup**. «Бекапи не може видалити навіть адміністратор» → **AWS Backup Vault Lock**.

Незмінна інфраструктура

Immutable infrastructure («ім'ютабл», незмінна інфраструктура): сервери не латають «наживо», а **замінюють** новими, зібраними з нового образу AMI. Як

у пекарні: зіпсований торт не латають кремом — печуть новий за рецептом. Інструменти — instance refresh в Auto Scaling і blue/green з розділу про контейнери.

Переваги: кожен сервер точно відповідає кресленню, без «ручних» сюрпризів, а відкотитися легко — достатньо повернути старий образ.

🔍 **Для допитливих: Навчальна пожежна тривога**

План, який ніколи не перевіряли, — це лише надія. Тому команди влаштовують «дні гри» (game days): навмисно вимикають сервер, зону чи частину системи й дивляться, чи спрацює захист. В AWS для цього є **AWS Fault Injection Service** — керовані «аварії» за сценарієм: зупинити частину серверів, додати затримку в мережу, перевантажити процесор. Звучить страшно, але краще знайти слабке місце під час навчальної тривоги, ніж під час справжньої пожежі.

🥐 Оля й Тарас домовилися: втратити можна щонайбільше хвилину замовлень (RPO), стояти — щонайбільше 15 хвилин (RTO). Тому обрали **Warm Standby** в Ірландії: зменшена копія сайту працює там постійно, Aurora Global Database тримає базу з відставанням менше за секунду, DynamoDB Global Tables — кошики, а S3 реплікує фото тортів. Щойно Франкфурт перестане відповідати, Route 53 перемкне клієнтів, а автомасштабування швидко збільшить копію до повного розміру. AWS Backup щоночі кладе копії в окремий акаунт, у сейф під Vault Lock. А раз на квартал команда влаштовує навчальну пожежну тривогу — і перевіряє, що план справді працює.

📖 **Словничок**

Єдина точка відмови (SPOF) — деталь, з поломкою якої зупиняється все.

HA / fault tolerance / DR — запасне колесо / двомоторний літак / план на пожежу в регіоні.

RPO / RTO — скільки даних можна втратити / скільки часу можна простояти.

Backup & Restore — у резервному регіоні лише бекапи; години.

Pilot Light — працює лише реплікована база, сервери вимкнені; десятки хвилин.

Warm Standby — зменшена повна копія працює постійно; хвилини.

Multi-site Active-Active — обидва регіони обслуговують клієнтів; майже нуль.

Aurora Global Database — RPO ≈ 1 с, RTO < 1 хв між регіонами.

DynamoDB Global Tables / S3 CRR + RTC — NoSQL у кількох регіонах / копія файлів в інший регіон (99,99% за 15 хв).

AWS Backup — централізовані бекапи з копіями між регіонами й акаунтами.

Vault Lock — бекап не видалить ніхто до кінця строку (захист від ransomware).

Elastic Disaster Recovery (DRS) — безперервні копії серверів, RPO секунди.

Exponential backoff / jitter — паузи між повторами ростуть / з додаванням випадковості.

Ідемпотентність — повторна обробка дає той самий результат.

Immutable infrastructure — сервери не латають, а замінюють новими з образу.

🧩 **Міні-тест: перевір себе**

? RPO і RTO — кілька годин, головне — мінімальна вартість. Яку стратегію DR обрати?

- **А** Backup & Restore
- **Б** Pilot Light
- **В** Warm Standby
- **Г** Multi-site Active-Active

Відповідь: А. Найдешевша стратегія, що вкладається в години, — лише бекапи в резервному регіоні.

? RTO — близько 30 хвилин. База має постійно реплікуватися в резервний регіон, а сервери можна запускати вже під час аварії. Що це?

А Backup & Restore

Б Pilot Light

В Warm Standby

Г Multi-site Active-Active

Відповідь: Б. Pilot Light: постійно «горить» лише ядро — база, — а сервери розпалюють під час аварії.

? RTO — кілька хвилин. У резервному регіоні постійно працює зменшена копія всього застосунку. Що це?

А Backup & Restore

Б Pilot Light

В Warm Standby

Г Multi-site Active-Active

Відповідь: В. Warm Standby: прогріта зменшена копія, яку під час аварії лише збільшують.

? Компанія має Multi-AZ базу RDS і вважає, що вже захищена від збою цілого регіону. Чи так?

А Так, Multi-AZ захищає і від збою регіону

Б Ні: Multi-AZ працює в межах одного регіону, для регіону потрібна копія в іншому регіоні

В Так, якщо ввімкнути detailed monitoring

Г Ні, бо Multi-AZ не захищає навіть від збою зони

Відповідь: Б. Multi-AZ рятує від падіння зони. Від падіння регіону — лише резервний регіон.

? Після збою тисячі клієнтів одночасно повторюють запити й знову валять сервіс, що щойно ожив. Що змінити в коді клієнтів?

А Повторювати якомога частіше

Б Повтори з exponential backoff і jitter

В Повністю прибрати повтори

Г Збільшити тайм-аут до години

Відповідь: Б. Паузи, що ростуть, і випадковість розносять повторні спроби в часі.

? Старий застосунок працює на одному EC2. Змінювати код не можна, але після збою сервер має відновлюватися автоматично. Що обрати?

А Більший тип інстансу

Б Auto Scaling group з min = max = 1 і health check

В Read replica

Г CloudTrail

Відповідь: Б. Група завжди тримає рівно один здоровий сервер: зламався — створює новий.

? Бекапи мають бути захищені від вірусу-вимагача: до кінця строку їх не має видалити навіть адміністратор. Що обрати?

- **А** Versioning у S3
- **Б** AWS Backup Vault Lock
- **В** Multi-AZ
- **Г** Правило AWS Config

Відповідь: Б. Vault Lock — сейф з часовим замком (WORM): ніхто не видалить бекап до строку.

? Під час навчання з аварії резервний регіон не зміг запустити потрібну кількість серверів. Що треба було зробити заздалегідь?

- **А** Увімкнути CloudTrail
- **Б** Підняти Service Quotas у резервному регіоні
- **В** Перейти на Spot
- **Г** Видалити старі AMI

Відповідь: Б. Ліміти діють у кожному регіоні окремо. Для резервного їх піднімають наперед, а не під час пожежі.

✅ **Головне**

Усе колись ламається: шукай **єдині точки відмови** й дублюй їх. Multi-AZ рятує від збою зони, **не регіону**.

HA в регіоні: **ALB + ASG у 2+ AZ**, **Multi-AZ база**, сервери **без стану**, статика в S3 + CloudFront, NAT у кожній AZ, **SQS** між шарами.

Старий застосунок без змін коду: **ASG min = max = 1**, EFS, Multi-AZ RDS, RDS Proxy, Route 53 health checks.

Стійкий код: повтори з **exponential backoff і jitter**, **ідемпотентність**.

DR від дешевої до дорогої: **Backup & Restore** (години) → **Pilot Light** (база, десятки хвилин) → **Warm Standby** (зменшена копія, хвилини) → **Active-Active** (≈ нуль). Обирай найдешевшу, що вкладається в RPO і RTO.

Резервний регіон: Route 53 failover / Global Accelerator, **Aurora Global Database**, **DynamoDB Global Tables**, **S3 CRR**, копії AMI, CloudFormation, **квоти заздалегідь**.

Реплікація — не бекап. **AWS Backup** — копії між регіонами й акаунтами; **Vault Lock** — не видалить ніхто.

💰 Розумний гаманець: з чого складається рахунок і як платити менше

👴 Простими словами

Рахунок AWS складається з того, скільки працювали сервери, скільки зберігалося даних і скільки їх вийшло назовні. Платиш менше, якщо вимикаєш непотрібне, обираєш правильний розмір і знижки.

🎯 Що ти дізнаєшся

- з чого складається рахунок AWS і де ховаються «тихі» витрати
- карта цін на переміщення даних — що безкоштовно, а що платно
- як заощаджувати на обчисленнях, сховищі, базах і мережі
- чим Budgets відрізняється від Cost Explorer, Cost and Usage Report і Cost Anomaly Detection
- як розподілити витрати між відділами — тегами й акаунтами
- як на іспиті розуміти «найдешевше рішення»

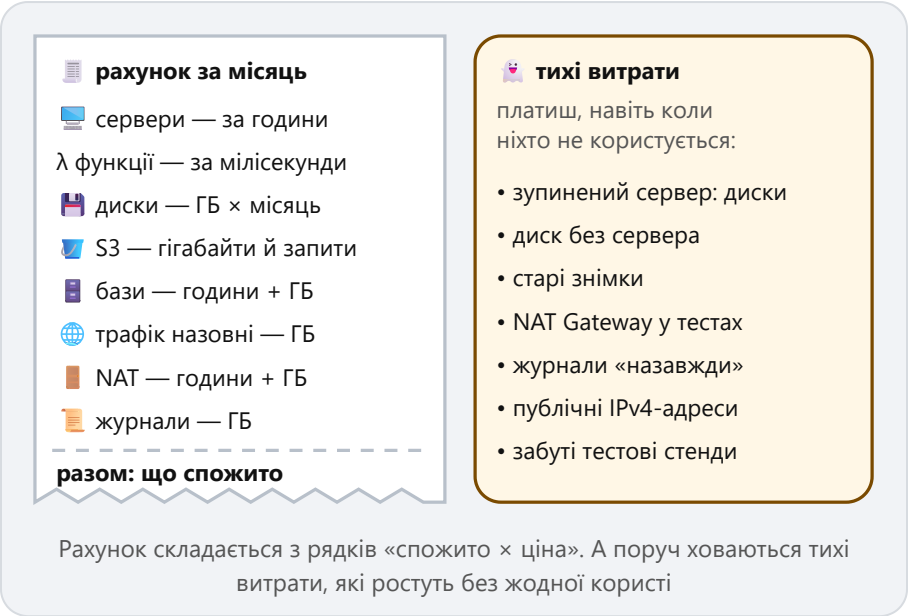
🥐 Наприкінці кварталу Соня з бухгалтерії кладе перед Олею рахунок AWS: за три місяці він виріс утричі. «Ми ж продаємо лише вдвічі більше!» Тарас відкриває Cost Explorer і знаходить винуватців: тестові сервери працюють цілодобово, хоча розробники йдуть додому о сьомій; NAT Gateway бере гроші за кожен гігабайт фото, що йде в S3; журнали CloudWatch зберігаються «назавжди»; три диски ні до чого не підключені, а сотні старих знімків ніхто не видаляє. «Хмара не дорога, — каже Тарас. — Дорога хмара без господаря».

Рахунок — як чек у супермаркеті

Ти вже знаєш головне правило хмари: **платиш за те, що спожито**. Рахунок AWS схожий на довгий чек у супермаркеті: кожен рядок — це «скільки спожито × ціна». Сервери — за години чи секунди роботи, функції — за мілісекунди, диски й сховища — за гігабайти на місяць, деякі сервіси — за кожен мільйон запитів, а мережа — за гігабайти переданих даних.

Найболючіше в рахунку — не великі рядки, які всі бачать, а **тихі витрати**: те, за що платиш, навіть коли

ніхто цим не користується.



Рахунок складається з рядків «спожито × ціна». А поруч ховаються тихі витрати, які ростуть без жодної користі

- **Зупинений сервер** не платить за процесор, але **диски EBS** при ньому далі коштують грошей.
- **Диск без сервера** — сервер видалили, а диск залишився.
- **Старі знімки** (snapshots) — копії, які роками ніхто не відкриває.
- **NAT Gateway** — бере плату за кожну годину роботи й за кожен гігабайт, навіть у тестовому середовищі.
- **Журнали «назавжди»** — CloudWatch Logs за замовчуванням нічого не видаляє.

- **Публічні IPv4-адреси** — з 2024 року платні всі, зокрема Elastic IP.
- **Забуті тестові стенди** — «я на хвилинку запущу...» і забули на півроку.

Карта цін на переміщення даних

Образ з простого курсу: **в'їзд у місто AWS безкоштовний, а виїзд — платний**. Переїзд між районами (AZ) — платний міст. **NAT Gateway** — платний КПП, що бере гроші з кожної тонни вантажу (гігабайта). **Gateway endpoint** — безкоштовна приватна дорога до S3 і DynamoDB. А **CloudFront** — доставка з місцевих складів, дешевша за доставку з центру.



- **Вхід** даних з інтернету в AWS — **безкоштовно**.
- У межах **однієї AZ** за приватними адресами — **безкоштовно**.
- **Між AZ** — **платно**, навіть усередині однієї VPC. Так, висока доступність у двох зонах коштує трохи грошей — і це нормальна ціна за надійність.
- **Між регіонами** — платно.
- **В інтернет** — найдорожче. Тому роздачу фото й сторінок ставлять за **CloudFront**: він кешує контент близько до людей і зменшує вихідний трафік з серверів.

- Через **NAT Gateway** — плата за години плюс за **кожен оброблений гігабайт**. Для S3 і DynamoDB є безкоштовна альтернатива — **gateway endpoint**.
- **Direct Connect** дає дешевший вихідний трафік для **великих постійних** обсягів.

> 🎯 «Великий рахунок за NAT Gateway, бо сервери з приватних підмереж пишуть у S3» → **gateway endpoint для S3**. «Великі витрати на вихідний трафік для фото й відео» → **CloudFront**.

Обчислення: таксі, проїзний чи горящий тур

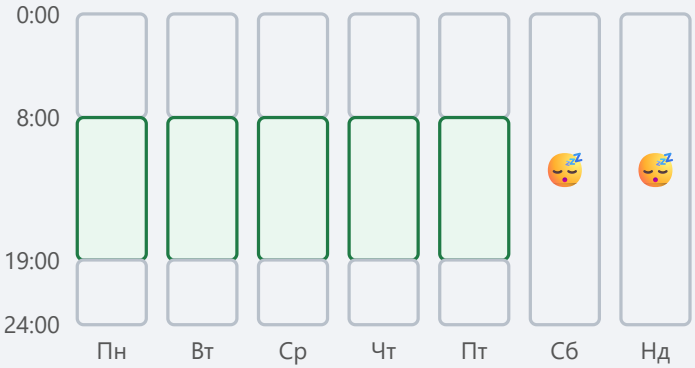
Способи оплати EC2 ти вже знаєш з розділу про ціни: On-Demand — **таксі**, Savings Plans і Reserved Instances — **проїзний**, Spot — **горящий тур**. Правило вибору:

- **Постійне** навантаження 24/7 → **Savings Plans** або **Reserved Instances**.
- **Переривне** (задачу можна перервати й продовжити) → **Spot**.
- **Невідоме**, коротке чи нове → **On-Demand** або **serverless**.

І ще чотири звички, що економлять щомісяця:

- **Правильний розмір** (rightsizing, «райтсайзинг»): сервер завантажений на 5% — візьми менший. Підкаже **Compute Optimizer**.
- **Graviton** — процесори AWS на ARM: та сама робота дешевше.
- **Auto Scaling** замість постійного запасу: сервери додаються в пік і прибираються після.
- **Вимикати тестові сервери** поза робочим часом — за розкладом через **EventBridge Scheduler** з Lambda або готовим рішенням **Instance Scheduler**.

Тестові сервери: цілодобово чи лише в робочі години?



🟩 **увімкнено лише 55 годин із 168 на тиждень**

рахунок за ці сервери — приблизно на дві третини менший

Тестові сервери, що працюють лише в будні з 8:00 до 19:00, вмикаються 55 годин із 168 — і коштують приблизно втричі менше

Окремо — вічне питання **Lambda чи EC2**. Для переривчастого чи невеликого навантаження **Lambda**

дешевша: немає запитів — немає плати. А для **постійного високого** навантаження 24/7 зазвичай дешевші **EC2 або Fargate з Savings Plans** — кухар на виклик дорогий, якщо дзвониш йому щохвилини цілодобово.

🎯 «Тестові сервери потрібні лише в робочі години» → **зупиняти за розкладом** (EventBridge Scheduler + Lambda). «Постійне високе навантаження на Lambda 24/7, рахунки ростуть» → **EC2 або Fargate + Savings Plans**.

Сховище: не тримати мотлох у дорогих шафах

- **Lifecycle у S3**: старі фото самі переїжджають у дешевші класи (Standard → Standard-IA → Glacier) і видаляються після строку.
- **Intelligent-Tiering** — якщо невідомо, як часто читатимуть файли: S3 сам перекладає їх у дешевші яруси.
- **Storage Class Analysis** і **Storage Lens** — підкажуть, які дані варто перевести в дешевші класи.
- **gp3 замість gp2** для дисків EBS — дешевше, а швидкість налаштовується окремо від розміру.
- **Прибирати** старі знімки й непідключені диски.

- **EFS Infrequent Access / Archive** — для файлів, які рідко відкривають.
- Для аналітики — **Parquet і стиснення**: Athena бере гроші за прочитане, а Parquet читає лише потрібні колонки.
- **Строк зберігання журналів** у CloudWatch Logs — не «назавжди», а скільки справді потрібно.

Бази даних

- **Нерівномірне** навантаження → **Aurora Serverless v2** або **DynamoDB on-demand**: платиш лише за фактичну роботу. **Стабільне** → звичайні (provisioned) сервери з резервуванням.
- **Кеш** (ElastiCache, DAX) замість дорожчого сервера бази; **read replicas** замість дедалі більшого сервера.
- 🆕 **Database Savings Plans** — знижка до 35% на RDS, Aurora, DynamoDB та інші бази за зобов'язання на рік чи три.
- **RDS для тестів** можна зупинити — до 7 днів, потім він стартує сам. Або перевести тестову базу на Aurora Serverless v2.

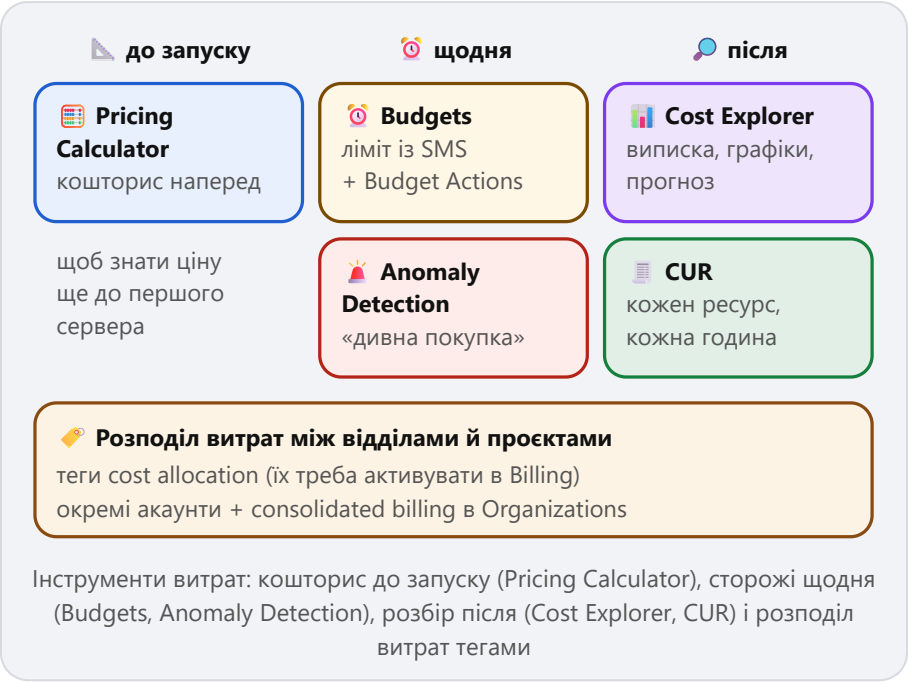
Мережа

- **Gateway endpoints** для S3 і DynamoDB замість NAT Gateway.
- **CloudFront** для фото, відео й сторінок.
- **NAT Gateway у кожній AZ** — дорожче, зате висока доступність і без плати за трафік між зонами. **Одна класична NAT** на всю мережу (вона живе в одній зоні) — дешевше, але це єдина точка відмови; годиться для тестів.
- Компоненти, що постійно «балакають» між собою, тримай **в одній AZ чи регіоні**, якщо вимоги до доступності це дозволяють.
- **Transit Gateway чи VPC Peering?** Transit Gateway бере плату за кожне підключення й за кожен гігабайт, що проходить через нього. Peering — лише за сам трафік між зонами чи регіонами. Для кількох VPC, що обмінюються великими обсягами, **peering дешевший**.
- **Direct Connect** — для великих постійних обсягів між офісом і AWS.

🎯 «Дві VPC обмінюються великими обсягами даних, потрібне найдешевше з'єднання» → **VPC Peering**, а не Transit Gateway.

Хто стежить за гаманцем

Інструменти зручно розкласти за часом: що робити **до** запуску, **щодня** і **після** — для розбору.



Інструменти витрат: кошторис до запуску (Pricing Calculator), сторожі щодня (Budgets, Anomaly Detection), розбір після (Cost Explorer, CUR) і розподіл витрат тегами

- **AWS Pricing Calculator** — **кошторис перед ремонтом**: оцінити вартість майбутньої архітектури ще до першого сервера.
- **AWS Budgets** — **ліміт на картці з SMS**: сповіщає, коли витрати **або їхній прогноз** перевищать поріг. **Budget Actions** вміють ще й діяти: наприклад, заборонити запуск нових ресурсів.

- **Cost Anomaly Detection** — **«банк помітив дивну покупку»**: машинне навчання помічає раптовий стрибок витрат і одразу пише.
- **AWS Cost Explorer** — **банківська виписка з графіками**: витрати за минулі місяці за сервісами, акаунтами й тегами, прогноз і поради, які Savings Plans чи RI варто купити.
- **Cost and Usage Report** (CUR, тепер через Data Exports) — **найдетальніший чек**: кожен ресурс, кожна година. Його складають у S3 й аналізують в Athena чи Amazon Quick.
- **Cost allocation tags** — **ярлики на чеках**: розподіл витрат між проєктами й відділами. Пастка: теги треба **активувати** в Billing, інакше в звітах їх не буде.
- **Consolidated billing** в Organizations — один рахунок на всі акаунти, **об'ємні знижки** і спільні RI та Savings Plans.

⚠️ Не плутай: **Budgets** — «попередь мене» (поріг і сповіщення). **Cost Explorer** — «покажи й поясни» (аналіз і прогноз). **CUR** — «дай усі сирі дані» (для власного аналізу).

«Найдешевше» на іспиті

Слова **MOST cost-effective** («найбільш економічно вигідне») — найчастіша вимога в питаннях. Але вона завжди означає: **найдешевше з тих рішень, що виконують усі інші вимоги**.

Якщо в питанні сказано «висока доступність», то дешевий варіант на одному сервері в одній зоні **програє**, хоч би яким дешевим був. Спершу викресли все, що не виконує вимоги, — і лише тоді порівнюй ціну.

🧪 **Спробуй у справжньому AWS: Порахуй, скільки коштуватиме сайт пекарні (10 хвилин, безкоштовно)**

Обидва інструменти безкоштовні й нічого не створюють.

1. Відкрий **calculator.aws** — це AWS Pricing Calculator. Входити в акаунт не обов'язково.
2. Натисни **Create estimate** і знайди **Amazon EC2**. Вкажи регіон Frankfurt, 2 сервери Linux будь-якого невеликого типу, роботу 24/7. Подивися місячну ціну.
3. Зміни спосіб оплати на **Savings Plans** на рік — і порівняй, наскільки стало дешевше.
4. Додай **Amazon S3**: 50 ГБ у класі Standard. Подивися підсумок оцінки.

5. Тепер у консолі AWS відкрий **Billing and Cost Management** → **Cost Explorer** і подивися свої реальні витрати за сервісами. На новому акаунті дані можуть з'явитися не одразу — Cost Explorer готує їх до доби.

🥐 За місяць Тарас навів лад. Тестові сервери вимикаються о 19:00 й вмикаються о 8:00 за розкладом EventBridge. Фото йдуть у S3 безкоштовною дорогою — gateway endpoint — замість NAT. Журнали зберігаються 30 днів, старі знімки й зайві диски видалено, а постійні сервери сайту перейшли на Savings Plans і процесори Graviton. Теги `project` показують Соні, скільки окремо коштують сайт, застосунок і аналітика, а Cost Anomaly Detection напише, якщо витрати раптом підскочать. Рахунок зменшився майже вдвічі — а замовлень стало ще більше.

📖 **Словничок**

Тихі витрати — те, за що платиш без користі: диски без серверів, старі знімки, NAT у тестах, журнали «назавжди», публічні IPv4.

Передача даних — вхід і рух в одній AZ безкоштовні; між AZ і регіонами — платно; в інтернет — найдорожче.

Gateway endpoint — безкоштовна приватна дорога до S3 і DynamoDB замість платного NAT.

Rightsizing («райтсайзинг») — підбір правильного розміру сервера.

Instance Scheduler / EventBridge Scheduler — вмикати й вимикати сервери за розкладом.

Database Savings Plans — знижка до 35% на бази за зобов'язання.

Pricing Calculator — кошторис майбутньої архітектури.

Budgets — сповіщення про перевищення бюджету чи прогнозу; **Budget Actions** — автоматичні дії.

Cost Explorer — аналіз і прогноз витрат, поради щодо Savings Plans і RI.

Cost and Usage Report (CUR) — найдетальніші дані: кожен ресурс, кожна година.

Cost Anomaly Detection — сповіщення про незвичні стрибки витрат.

Cost allocation tags — розподіл витрат за тегами (треба активувати в Billing).

Consolidated billing — один рахунок організації, об'ємні знижки.

🧩 Міні-тест: перевір себе

? Сервери в приватних підмережах щодня вивантажують сотні гігабайтів у S3, і рахунок за NAT Gateway величезний. Як зменшити витрати?

- **А** Gateway endpoint для S3
- **Б** Ще один NAT Gateway
- **В** Перевести сервери в публічну підмережу
- **Г** Interface endpoint для кожного сервера

Відповідь: А. Gateway endpoint — безкоштовна приватна дорога до S3: трафік більше не проходить через платний NAT.

? Фінансовий відділ хоче отримати лист, щойно прогноз витрат за місяць перевищить $1000. Що обрати?

А Cost Explorer

Б AWS Budgets

В Cost and Usage Report

Г Trusted Advisor

Відповідь: Б. Budgets стежить за витратами й прогнозом і сповіщає, коли перетнуто поріг.

? Треба бачити, скільки коштують окремо проєкти «сайт» і «аналітика», що живуть в одному акаунті. Що зробити?

А Увімкнути consolidated billing

Б Поставити теги project на ресурси й активувати їх як cost allocation tags

В Купити Savings Plans

Г Налаштувати Cost Anomaly Detection

Відповідь: Б. Cost allocation tags розкладають витрати за тегами, але лише після активації в Billing.

? Аналітики хочуть найдетальніші дані про витрати — кожен ресурс, кожна година — і аналізувати їх в Athena. Що обрати?

А Cost Explorer

Б AWS Budgets

В Cost and Usage Report

Г Pricing Calculator

Відповідь: В. CUR — найдетальніший «чек»: складається в S3 і читається в Athena.

? Тестові сервери потрібні лише в робочі години в будні. Як найпростіше заощадити?

А Купити Reserved Instances на 3 роки

Б Зупиняти й запускати їх за розкладом (EventBridge Scheduler + Lambda або Instance Scheduler)

В Перевести на Dedicated Hosts

Г Збільшити тип інстансу

Відповідь: Б. Вимкнений сервер не платить за процесор: 55 годин замість 168 на тиждень.

? Дві VPC в одному регіоні постійно обмінюються великими обсягами даних. Яке з'єднання найдешевше?

- **А** VPC Peering
- **Б** Transit Gateway
- **В** Site-to-Site VPN
- **Г** Обмін через NAT Gateway

Відповідь: А. Peering не бере плати за підключення й обробку трафіку — лише за сам трафік. Transit Gateway бере і за те, і за інше.

? Функція Lambda працює під постійним високим навантаженням цілодобово, і рахунок постійно росте. Що зазвичай дешевше?

А Збільшити пам'ять функції

Б Перенести на EC2 або Fargate з Savings Plans

В Додати ще одну функцію

Г Перейти на On-Demand Capacity Reservations

Відповідь: Б. Для постійного навантаження 24/7 сервери з проїзним (Savings Plans) зазвичай дешевші за оплату кожного виклику.

? У питанні потрібна висока доступність і «MOST cost-effective». Варіант А — один сервер в одній AZ, найдешевший. Варіант Б — два менші сервери у двох AZ за балансувальником. Що обрати?

А Варіант А, бо він найдешевший

Б Варіант Б, бо лише він виконує вимогу високої доступності

В Обидва однаково правильні

Г Жоден: висока доступність можлива лише в кількох регіонах

Відповідь: Б. «Найдешевше» — це найдешевше з рішень, що виконують усі вимоги. Варіант А вимогу HA не виконує.

✅ **Головне**

Платиш за спожите: обчислення, зберігання, запити, передачу даних. Шукай **тихі витрати**: диски без серверів, старі знімки, NAT у тестах, журнали «назавжди», публічні IPv4.

Трафік: **вхід** і рух **в одній AZ** — безкоштовно; **між AZ** (навіть в одній VPC) і **між регіонами** — платно; **в інтернет** — найдорожче. NAT бере за кожен ГБ → **gateway endpoint** для S3 і DynamoDB; роздача контенту → **CloudFront**.

Обчислення: постійне → **Savings Plans / RI**, переривне → **Spot**, невідоме → **On-Demand / serverless**; **rightsizing** (Compute Optimizer), **Graviton**, тести — **за розкладом**.

Сховище: **lifecycle**, **Intelligent-Tiering**, **gp3**, прибирати знімки й диски, Parquet для аналітики. Бази: **Aurora Serverless v2 / DynamoDB on-demand** для нерівного навантаження, кеш, **Database Savings Plans**.

Мережа: дві VPC з великим трафіком → **peering** дешевший за Transit Gateway; одна класична NAT — лише для тестів.

Інструменти: **Pricing Calculator** (до), **Budgets** і **Anomaly Detection** (щодня), **Cost Explorer** і **CUR** (після), **cost allocation tags** (активувати!) і **consolidated billing**.

«Найдешевше» на іспиті = найдешевше з того, що виконує **всі** вимоги.

🪜 Від одного сервера до двох регіонів: як мислить архітектор

👴 Простими словами

Архітектор не будує все одразу. Почни з одного сервера, а потім крок за кроком додавай: базу окремо, кілька серверів, кеш, другу зону, другий регіон — лише тоді, коли це справді потрібно.

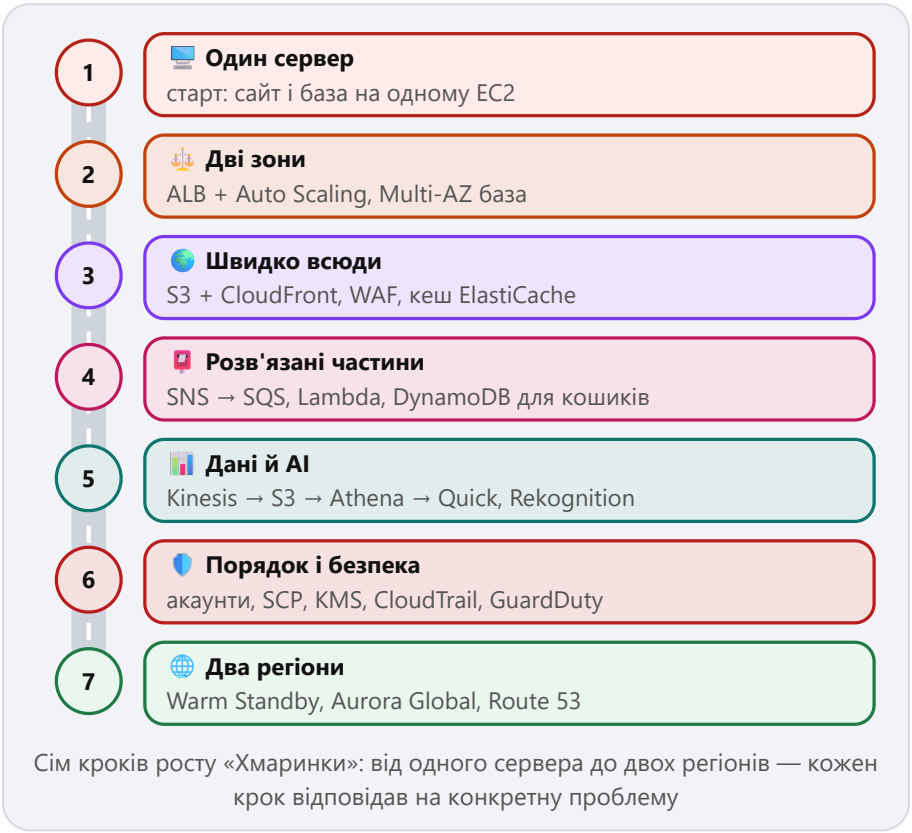
🎯 Що ти дізнаєшся

- як виросла архітектура «Хмаринки» — сім кроків від одного сервера до двох регіонів, і яку проблему розв'язував кожен
- шість питань, які архітектор ставить до першої стрілки на схемі
- чотири таблиці вибору — обчислення, зберігання, бази й інтеграція
- як перевірити будь-яку схему шістьма стовпами Well-Architected
- на які компроміси доводиться йти
- уся «Хмаринка» на одній схемі і подорож одного замовлення через неї

🥐 Оля готується до зустрічі з інвестором: «Хмаринка» відкриває пекарні ще у двох країнах, і треба показати, на чому тримається вся система. «Намалюй мені все на одному аркуші», — просить вона Тараса. Тарас усміхається: «На одному аркуші — можна. Але спершу я розповім, як ми до цього дійшли. Бо кожна стрілка на цьому аркуші з'явилася тому, що колись щось зламалося або стало тісно».

Сім кроків росту

Жодна велика система не народжується великою. Вона росте крок за кроком, і кожен крок — відповідь на конкретну проблему. Пройдімо шлях «Хмаринки» ще раз, тепер — з висоти пташиного польоту.



Сім кроків росту «Хмаринки»: від одного сервера до двох регіонів — кожен крок відповідав на конкретну проблему

1. **Один сервер.** Сайт і база жили на одному EC2 з публічною адресою. Дешево і просто, але це єдина точка відмови: у Чорну п'ятницю сервер «ліг», а коли зламався диск, пропали замовлення. *Розділи про EC2 і масштабування.*

2. **Дві зони.** База переїхала в RDS з Multi-AZ, сервери — у групу Auto Scaling за балансувальником ALB у двох AZ, у приватних підмережах. Кошики винесли з серверів, і сервери стали без стану. Тепер збій сервера чи навіть цілої зони клієнти майже не помічають. *Розділи про VPC, масштабування й RDS*.
3. **Швидко всюди.** Фото тортів — у S3 за CloudFront, а сховище закрите через OAC, тож і в Торонто сторінки відкриваються миттєво. WAF і Shield захищають від ботів і DDoS, а ElastiCache тримає меню, яке читають сто тисяч разів на день. *Розділи про S3, CDN, загрози й кеш*.
4. **Розв'язані частини.** Сайт лише кладе замовлення в SNS, а кухня, бухгалтерія й SMS мають власні черги SQS — пік більше не губить замовлень. Фото зменшує Lambda, мобільний застосунок працює через API Gateway, кошики — у DynamoDB. *Розділи про Lambda, повідомлення й DynamoDB*.
5. **Дані й AI.** Події застосунку течуть через Kinesis і Firehose в озеро даних S3. Glue впорядковує, Athena відповідає на питання, Amazon Quick малює дашборди. Rekognition перевіряє фото у відгуках, Comprehend визначає настрій. *Розділи про потоки, аналітику й AI*.

6. **Порядок і безпека.** Окремі акаунти для сайту, тестів і журналів в Organizations, SCP — «конституція» для всіх. Люди отримують лише потрібні права, програми — ролі замість ключів. Дані зашифровано ключами KMS, а CloudTrail, Config, GuardDuty і Security Hub стежать за всім. *Розділи про безпеку й моніторинг.*
7. **Два регіони.** Warm Standby в Ірландії, Aurora Global Database, DynamoDB Global Tables, реплікація S3 і перемикання через Route 53. Уся інфраструктура описана в CloudFormation, бекапи лежать в AWS Backup під Vault Lock. *Розділи про надійність і гроші.*

Бачиш закономірність? Кожен крок додавав складності лише тоді, коли попередня схема **справді** перестала справлятися. Це головна навичка архітектора: **не будувати космодром для пекарні**, але й не чекати, поки все впаде.

Шість питань до першої стрілки

Перш ніж малювати схему, архітектор розпитує. Відповіді на ці питання — і є «вимоги», з яких складається кожне питання іспиту:

1. **Хто користувачі й де вони?** Одне місто чи весь світ → один регіон чи кілька, CloudFront, Global Accelerator.

2. **Скільки їх і коли приходять?** Рівно, піками, непередбачувано → Auto Scaling за розкладом чи за метрикою, serverless, черги як буфер.
3. **Які дані?** Таблиці зі зв'язками, мільйони простих записів, файли, потоки → вибір сховища й бази.
4. **Що буде, якщо зламається?** Скільки можна простояти (RTO) і скільки даних втратити (RPO) → Multi-AZ, стратегія DR, бекапи.
5. **Хто має доступ і що треба записувати?** Персональні дані, шифрування, аудит → IAM, KMS, CloudTrail, окремі акаунти.
6. **Скільки можна витратити і хто це обслуговуватиме?** Бюджет і розмір команди → керовані й serverless-сервіси, коли людей мало.

💡 На іспиті ці відповіді вже написані в тексті питання. Твоє завдання — помітити їх: «з усього світу», «непередбачувані піки», «без змін коду», «RPO 15 хвилин», «мінімальні операційні витрати».

Чотири таблиці вибору

Більшість питань іспиту зводиться до одного з чотирьох виборів. Ось вони — як підсумок усієї книжки.

Де запускати код:

Ситуація	Вибір
Короткі задачі за подією (до 15 хвилин), без серверів	**Lambda**
Контейнери без керування серверами	**ECS або EKS на Fargate**
Команда вже працює з Kubernetes	**EKS**
Потрібен повний контроль ОС, особливі ліцензії, довгі процеси	**EC2**
Просто задеплоїти веб-застосунок, не думаючи про інфраструктуру	**Elastic Beanstalk**
Великі пакетні обчислення	**AWS Batch**

Де зберігати файли:

Ситуація	Вибір
Фото, бекапи, озеро даних, статика сайту	**S3**
Диск для одного сервера (бази — io2 з високими IOPS)	**EBS**
Спільна папка для багатьох серверів Linux	**EFS**
Спільна папка Windows (SMB) / обчислення HPC	**FSx for Windows** / **FSx for Lustre**
Архів на роки, читають майже ніколи	**S3 Glacier Deep Archive**
Офіс працює з файлами, а зберігати їх — у хмарі	**Storage Gateway**

Яку базу обрати:

Ситуація	Вибір
Таблиці зі зв'язками, транзакції, SQL	**RDS** або **Aurora**
Мільйони простих запитів за ключем, serverless	**DynamoDB**
Прискорити часті читання	**ElastiCache** (для DynamoDB — **DAX**)
Звіти й аналітика на великих обсягах	**Redshift**; разові запити по S3 — **Athena**
Графи зв'язків / документи MongoDB / часові ряди / пошук	**Neptune** / **DocumentDB** / **Timestream** / **OpenSearch**

Як з'єднати частини системи:

Ситуація	Вибір
Буфер між частинами, кожне завдання — одному обробнику	**SQS** (суворий порядок — **FIFO**)
Розіслати одне повідомлення багатьом	**SNS** (+ черги SQS — fan-out)
Маршрутизація подій за правилами, події SaaS, розклад	**EventBridge**
Потік у реальному часі, кілька споживачів, перемотка	**Kinesis Data Streams**
Доставити потік у S3 чи Redshift без коду	**Data Firehose**
Багатокроковий процес з повторами й гілками	**Step Functions**
Наявні застосунки на RabbitMQ чи ActiveMQ / на Kafka	**Amazon MQ** / **MSK**

🎯 Ці таблиці — скорочення розділу шпаргалки «X чи Y?». Перечитай його після цього розділу: там ті самі пари, але з пастками.

Перевірка шістьма стовпами

Готову схему архітектор перевіряє тими самими шістьма стовпами Well-Architected, з яких ми починали. Ось як їх пройшла «Хмаринка»:

Стовп	Питання	Що зроблено в «Хмаринці»
🛠️ Операційна досконалість	Чи легко змінювати й обслуговувати?	CloudFormation, CloudWatch, Systems Manager, автоматичні латки
🔐 Безпека	Чи захищені дані й доступи?	Окремі акаунти, SCP, ролі, KMS, WAF, CloudTrail, GuardDuty
🛟 Надійність	Чи переживе збій?	Дві AZ, Multi-AZ база, черги, Warm Standby, AWS Backup
⚡ Продуктивність	Чи швидко для всіх?	CloudFront, ElastiCache, правильні бази для кожної задачі

Стовп	Питання	Що зроблено в «Хмаринці»
💰 Оптимізація витрат	Чи не переплачуємо?	Savings Plans, Spot для нічних задач, lifecycle, endpoints, розклад
🌱 Сталий розвиток	Чи економно витрачаємо ресурси?	Graviton, правильні розміри, serverless, керовані сервіси

Компроміси

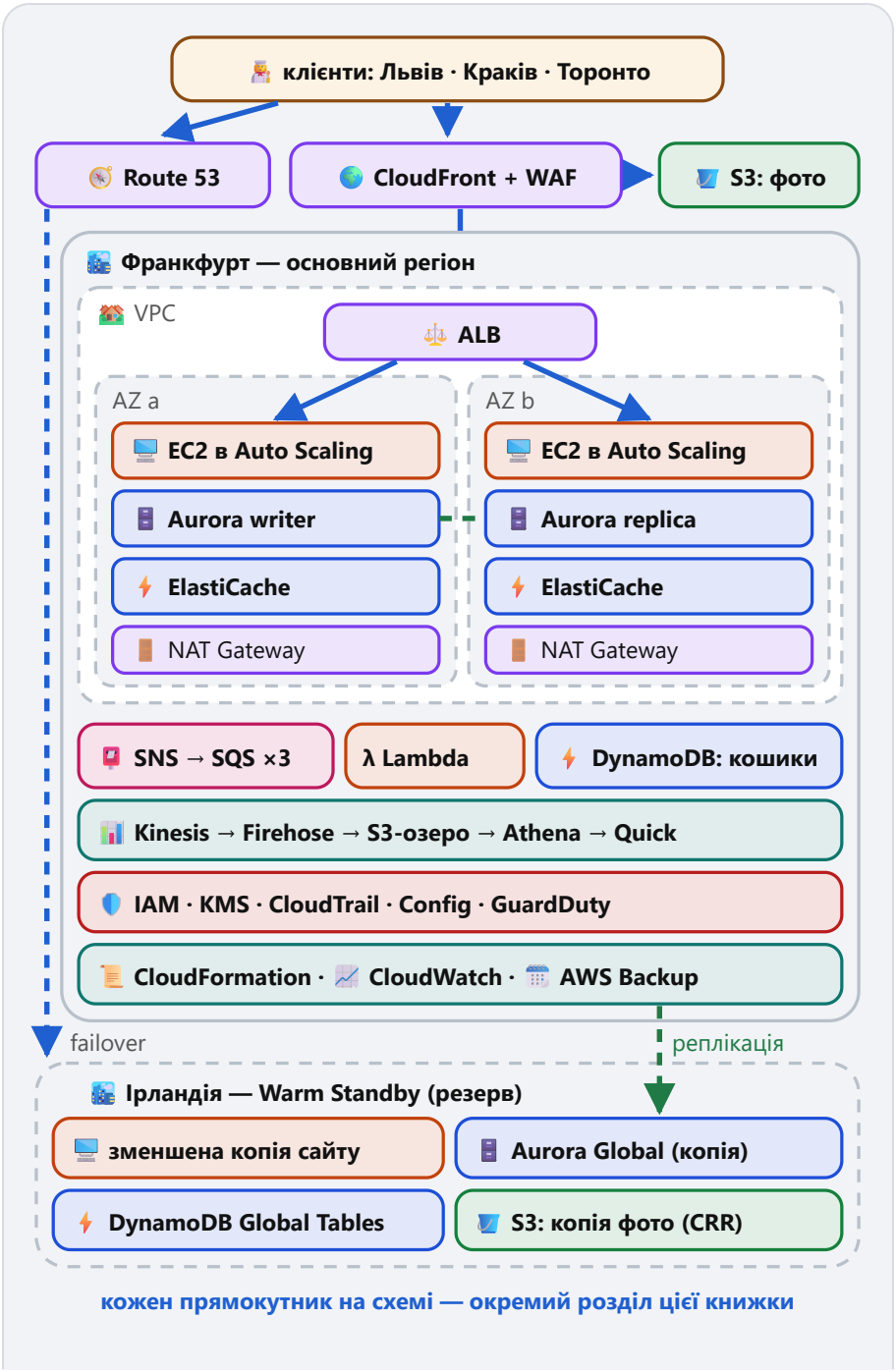
Ідеальної архітектури не існує — є **компроміси** (trade-offs, «трейд-офс»). Архітектор свідомо обирає, чим пожертвувати:

- **Надійність коштує грошей.** Дві зони — це плата за трафік між ними, резервний регіон — друга система. Пекарні вистачить двох зон і Warm Standby, а банку може знадобитися Active-Active.
- **Свіжість чи швидкість.** Кеш відповідає за мілісекунди, але може віддати трохи застарілі дані; так само й узгодженість «з часом» (eventual consistency) у реплік.
- **Контроль чи турбота.** EC2 дає повний контроль — і повну відповідальність за латки й масштабування. Lambda й Fargate знімають турботи, але дають менше налаштувань.
- **Простота чи гнучкість.** Один керований сервіс простіше обслуговувати, ніж зв'язку з п'яти самописних.

💡 На іспиті компроміс зазвичай вирішує одна фраза. «LEAST operational overhead» — обирай керований чи serverless-сервіс. «MOST cost-effective» — найдешевше з того, що виконує всі вимоги. «Without code changes» — рішення, яке працює ззовні застосунку.

Уся «Хмаринка» на одному аркуші

А ось і той самий аркуш, який Тарас приніс на зустріч з інвестором:



кожен прямокутник на схемі — окремий розділ цієї книжки

Уся архітектура «Хмаринки»: вхід через Route 53 і CloudFront, дві зони у Франкфурті, черги, дані, безпека й резервний регіон в Ірландії

Простеж на схемі **подорож одного замовлення** — це повторення всієї книжки за хвилину:

1. Клієнтка з Кракова відкриває **khmarynka.ua**. **Route 53** підказує браузеру адресу **CloudFront**.
2. CloudFront віддає фото тортів з найближчого кіоску (а бере їх із закритого сховища **S3**), а запит на замовлення пропускає через **WAF** до **ALB** у Франкфурті.
3. ALB відправляє запит на здоровий сервер **EC2** в одній із двох **AZ**. Сервер бере меню з **ElastiCache**, записує замовлення в **Aurora**, а кошик — у **DynamoDB**.
4. Сервер кладе замовлення в **SNS** і одразу відповідає «Дякуємо!». Кухня, бухгалтерія й служба SMS забирають його кожна зі своєї черги **SQS**, а **Lambda** надсилає клієнтці SMS.
5. Подія про замовлення летить у **Kinesis** — і за хвилину маркетолог бачить її на дашборді.
6. Усі дії адміністраторів записує **CloudTrail**, дані зашифровані **KMS**, **GuardDuty** шукає загрози.
7. А якщо Франкфурт раптом зникне, **Route 53** перемкне клієнтів на Ірландію, де вже працює зменшена копія з реплікованими даними.

Кожен прямокутник на схемі — це розділ, який уже позаду. Якщо якийсь здається незнайомим, повернися до нього: тепер ти точно знаєш, навіщо він потрібен.

🥐 Інвестор довго роздивлявся аркуш. «А якщо згорить Франкфурт?» — «За кілька хвилин клієнтів обслужить Ірландія». — «А якщо в Чорну п'ятницю прийде вдесятеро більше людей?» — «Сервери додадуться самі, а замовлення почекають у черзі». — «А хто має доступ до даних клієнтів?» — «Лише ті, кому треба, і кожну дію записано». Інвестор усміхнувся: «У вас не пекарня, а годинниковий механізм». А Оля подумала, що ще рік тому не знала навіть, що таке сервер.

📖 **Словничок**

Архітектура — план системи: з яких частин вона складається і як вони з'єднані.

Вимоги — що має вміти система і за яких умов: користувачі, навантаження, дані, RPO і RTO, безпека, бюджет.

Компроміс (trade-off) — свідомий вибір, чим пожертвувати заради головного.

Керований сервіс (managed) — AWS бере на себе сервери, латки й масштабування.

Слабке зв'язування (loose coupling) — частини спілкуються через черги й події і не падають разом.

Stateless — сервер не зберігає в собі стану клієнта, тож його можна будь-коли замінити.

Well-Architected — шість стовпів для перевірки будь-якої архітектури.

🧩 **Міні-тест: перевір себе**

? Сервери сайту мають вільно додаватися й зникати в Auto Scaling, не втрачаючи кошиків клієнтів. Що зробити?

А Зберігати кошики на диску EBS кожного сервера

Б Зберігати сесії й кошики в ElastiCache чи DynamoDB, а сервери зробити stateless

В Вимкнути автомасштабування

Г Увімкнути sticky sessions назавжди

Відповідь: Б. Стан виносять у спільне сховище — тоді будь-який сервер можна замінити без втрат.

? Фото тортів мають швидко відкриватися в усьому світі, а сховище — лишатися закритим. Що обрати?

- **А** Публічне сховище S3
- **Б** S3 за CloudFront з OAC
- **В** EBS з публічним IP
- **Г** EFS у кожному регіоні

Відповідь: Б. CloudFront кешує фото близько до людей, а OAC дає доступ до сховища лише йому.

? У пікові години частина замовлень губиться, бо кухонна система не встигає. Що змінити з найменшими зусиллями?

- **А** Поставити більший сервер кухні
- **Б** Поставити чергу SQS між сайтом і кухнею
- **В** Перенести сайт у другий регіон
- **Г** Увімкнути CloudTrail

Відповідь: Б. Черга — буфер: замовлення чекають, а кухня бере їх у своєму темпі.

? Потрібно зменшувати фото одразу після завантаження в S3 з найменшими операційними витратами. Що обрати?

- **А** Подія S3 → функція Lambda
- **Б** Сервер EC2, що щохвилини перевіряє сховище
- **В** Кластер EKS
- **Г** Щоденна задача AWS Batch

Відповідь: А. Подія + Lambda: без серверів, запускається лише тоді, коли з'являється фото.

? Клієнти в Канаді мають швидко читати реляційну базу, а сама база — пережити падіння регіону з RPO близько секунди. Що обрати?

- **А** RDS Multi-AZ
- **Б** Aurora Global Database
- **В** DynamoDB Global Tables
- **Г** Read replica в тій самій AZ

Відповідь: Б. Global Database дає локальне читання в іншому регіоні й перемикання з RPO ≈ 1 с.

? Дві зони, Multi-AZ база, черги, резервний регіон і бекапи — до якого стовпа Well-Architected це належить насамперед?

А Оптимізація витрат

Б Продуктивність

В Надійність

Г Сталий розвиток

Відповідь: В. Усе це про те, щоб система переживала збої, — стовп надійності.

? Питання вимагає «LEAST operational overhead» для API, що викликають кілька разів на годину. Що, найімовірніше, правильне?

- **А** Група EC2 за ALB у трьох AZ
- **Б** API Gateway + Lambda
- **В** Кластер EKS
- **Г** EC2 з власним вебсервером

Відповідь: Б. Мінімум турбот і плата лише за виклики — керований serverless.

? Архітектор додає резервний регіон для пекарні, хоча RTO у вимогах — 24 години, а бюджет мінімальний. Що не так?

- **А** Нічого, резервний регіон потрібен завжди
- **Б** Рішення надмірне: для RTO 24 години й мінімального бюджету вистачить Backup & Restore
- **В** Треба ще й Active-Active
- **Г** RTO 24 години неможливий в AWS

Відповідь: Б. Не будуй космодром для пекарні: обирай найпростіше, що виконує вимоги.

✅ **Головне**

Архітектура росте **кроками**: кожна складність з'являється, коли попередня схема перестала справлятися. Не будуй зайвого — але й не чекай, поки все впаде.

Шість питань: **хто й де, скільки й коли, які дані, що буде при збої (RPO/RTO), безпека й аудит, бюджет і команда**.

Чотири вибори — **обчислення, зберігання, бази, інтеграція** — покривають більшість питань іспиту.

Готову схему перевіряють **шістьма стовпами** Well-Architected.

Ідеалу немає — є **компроміси**: надійність коштує грошей, кеш — свіжості, контроль — турбот.

Фрази-підказки: **LEAST operational overhead** → керований чи serverless; **MOST cost-effective** → найдешевше з того, що виконує всі вимоги; **without code changes** → рішення ззовні застосунку.

🎯 Іспит зсередини: формат, анатомія питання і розбори

👴 Простими словами

Питання іспиту — завжди маленька історія з однією головною вимогою: «найдешевше», «найшвидше», «найменше роботи» чи «найбезпечніше». Знайди це слово — і половина відповідей відпаде сама.

🎯 Що ти дізнаєшся

- як влаштований SAA-C03 — питання, час, бал, домени, мови
- як отримати +30 хвилин, якщо англійська не рідна
- з чого складається кожне питання і як розібрати його за п'ять кроків
- які варіанти викреслюють одразу
- англійські фрази, що підказують відповідь
- сім повних розборів у стилі іспиту
- як записатися й що чекає в день іспиту

🥐 Після зустрічі з інвестором Оля несподівано каже: «Знаєш, я теж хочу скласти той іспит. Я ж тепер розумію, навіщо кожна стрілка на твоєму аркуші». Тарас аж підстрибує: «Тоді почнімо з того, як він влаштований. Цей іспит — не про те, хто більше вивчив напам'ять. Він про те, хто вміє уважно читати й обирати».

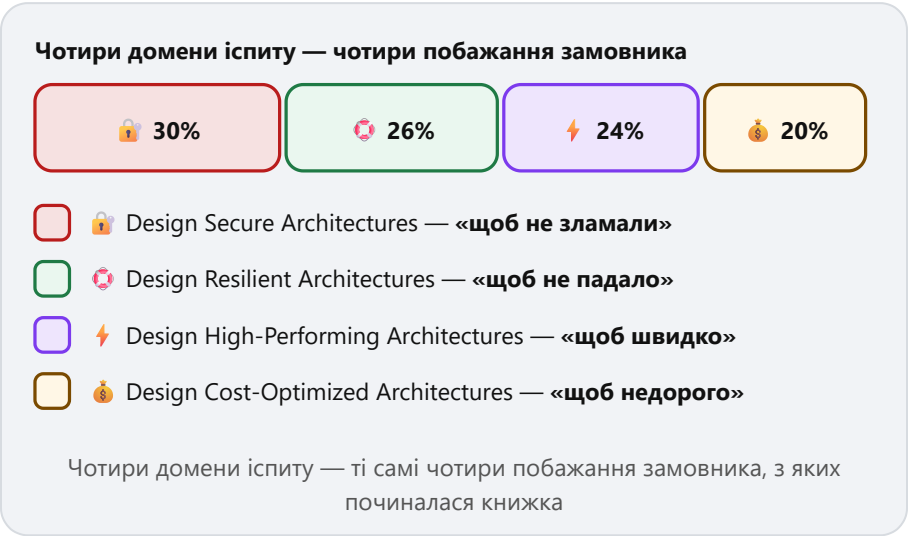
Іспит у цифрах

SAA-C03 — іспит **AWS Certified Solutions Architect – Associate**. Станом на вересень 2026 року актуальний саме він: якщо десь пишуть про «SAA-C04», це помилка — на сайті AWS такого іспиту немає.

Що	Як
Питань	**65**: 50 оцінюваних + 15 неоцінюваних (які саме — невідомо, тож старайся на кожному)
Час	**130 хвилин** (+30 із ESL, про це нижче)
Прохідний бал	**720 з 1000**
Ціна	**$150**
Штраф за помилку	**немає** — відповідай на всі питання
Типи питань	одна правильна з 4 (multiple choice) або кілька з 5 і більше (multiple response — скільки обрати, сказано в питанні)
Де складати	центр тестування Pearson VUE або онлайн удома з наглядачем (проктором)

Що	Як
Результат	на пошту й у кабінет, зазвичай за кілька годин (AWS обіцяє до 5 робочих днів)
Сертифікат	діє **3 роки**; після здачі — знижка 50% на наступний іспит AWS

Оцінка **компенсаторна**: не треба «здавати» кожен домен окремо, рахується загальний бал. Слабше місце можна перекрити сильнішим.



Чотири домени іспиту — ті самі чотири побажання замовника, з яких починалася книжка

Домени — це ті самі **чотири побажання замовника** з першого розділу: щоб не зламали, щоб не падало, щоб швидко, щоб недорого. Безпека — найбільший шматок, майже третина питань.

Мова іспиту і +30 хвилин

Іспит можна складати англійською, французькою, італійською (до 31 грудня 2026 року), японською, корейською, португальською (бразильською), іспанською чи китайською. **Української немає**. Саме тому в цій книжці кожен термін подано ще й англійською — на іспиті ти впізнаєш їх.

Назви сервісів в іспиті короткі: «Amazon SNS», а не «Amazon Simple Notification Service». Повні назви можна подивитися за кнопкою **Help** просто під час іспиту.

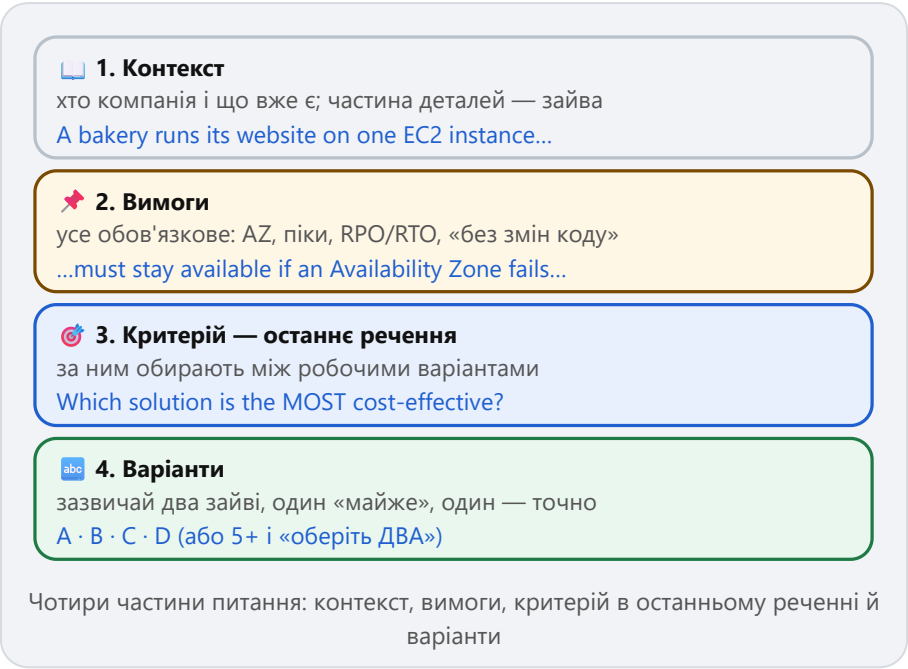
Якщо англійська не рідна, можна отримати **ще 30 хвилин** — це називається **ESL +30** (English as a Second Language, «англійська як друга мова»). Як це зробити:

1. Увійди в кабінет сертифікації AWS (через **aws.training** → **Certification**).
2. Обери **Request Exam Accommodations** → **ESL +30 MINUTES**.
3. Підтвердь запит — його схвалюють автоматично.
4. **Лише тепер** записуйся на іспит.

⚠️ Запит подають **один раз** і **до запису** на іспит. Якщо запис уже є, — скасуй його, подай запит і запишися знову. Інакше додаткових 30 хвилин не буде.

Анатомія питання

Майже кожне питання — міні-історія з чотирьох частин:



Чотири частини питання: контекст, вимоги, критерій в останньому реченні й варіанти

Ось приклад у стилі іспиту — спершу спробуй відповісти без підказок, а потім читай розбір:

A bakery runs its online ordering website on a single Amazon EC2 instance. During promotions the website becomes slow, and when the instance fails, customers cannot place orders. The company needs the website to remain available if an Availability Zone fails and to handle traffic spikes automatically. **Which solution meets these requirements MOST cost-effectively?**

A. Change the instance to a larger instance type.

B. Run the website on EC2 instances in an Auto Scaling group across two Availability Zones behind an Application Load Balancer.

C. Deploy a full copy of the website in a second AWS Region and send traffic to both Regions with Route 53.

D. Create hourly EBS snapshots and restore the instance from the latest snapshot when it fails.

🔍 **Для допитливих: Переклад питання українською**

Пекарня приймає онлайн-замовлення через сайт, який працює на одному сервері Amazon EC2. Під час акцій сайт гальмує, а коли сервер ламається, клієнти взагалі не можуть замовити. Компанії потрібно, щоб сайт працював навіть тоді, коли падає зона

доступності, і сам витримував піки трафіку. **Яке рішення виконує ці вимоги найбільш економно?**

- **A.** Замінити сервер на більший тип інстансу.
- **B.** Запустити сайт на серверах EC2 у групі Auto Scaling у двох зонах доступності за Application Load Balancer.
- **C.** Розгорнути повну копію сайту в другому регіоні AWS і розподіляти трафік між регіонами через Route 53.
- **D.** Щогодини робити знімки диска EBS і відновлювати сервер з останнього знімка, коли він зламається.

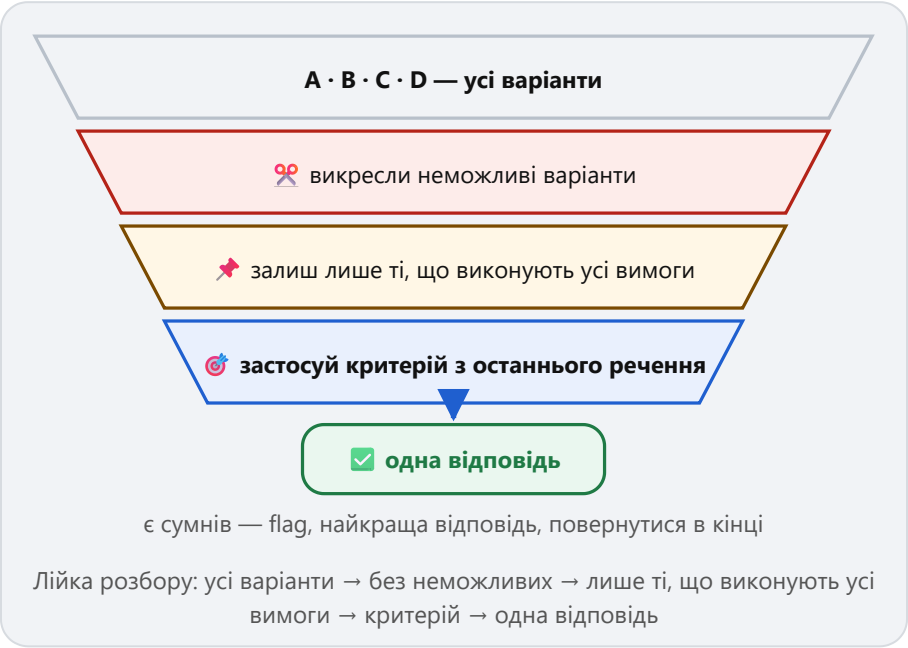
Розбір:

- **Контекст:** один сервер, гальмує в акції, падає разом із замовленнями.
- **Вимоги:** (1) працювати, якщо впала **зона**; (2) **автоматично** витримувати піки.
- **Критерій:** MOST cost-effectively — найдешевше з того, що виконує обидві вимоги.
- **A** — більший сервер усе одно один і в одній зоні: вимога (1) не виконана. ✗
- **D** — бекап не дає доступності: відновлення займе час і ручну роботу. ✗

- **C** — виконує все, але другий регіон для пекарні — надмірно й дорого. ✗ за критерієм.
- **B** — дві зони й автомасштабування: обидві вимоги, і дешевше за C. ✅

Зверни увагу: **C технічно працює**. Саме так і виглядають пастки іспиту — правильну відповідь від «майже правильної» відрізняє критерій з останнього речення.

П'ять кроків розбору



1. **Прочитай останнє речення** — там критерій: MOST cost-effective, LEAST operational overhead, MOST secure, highly available.

2. **Знайди вимоги** і подумки підкресли кожну: регіон чи зона, RPO і RTO, «без змін коду», real-time, serverless, шифрування.
3. **Викресли неможливе** — варіанти, які в AWS просто не працюють (список нижче).
4. З решти **обери керований і простіший**, що виконує **всі** вимоги.
5. **Є сумнів** — дай найкращу відповідь, познач **Flag for review** і йди далі. Наприкінці повернися до позначених.

Що викреслюють одразу

Ці варіанти неможливі — вони з'являються лише як пастки:

- **Правило Deny у Security Group** — охоронець лише впускає за списком; заборони ставлять у NACL чи WAF.
- **WAF на Network Load Balancer** — WAF працює з CloudFront, ALB, API Gateway, а з NLB — ні.
- **Gateway endpoint для SQS** чи іншого сервісу — gateway буває лише для **S3 і DynamoDB**.
- **Читання з дублера RDS Multi-AZ** — він лише чекає на аварію, читати — з read replicas.
- **Диск EBS для сервера в іншій AZ** — диск прикручений до своєї зони.

- **Lambda на 2 години** — максимум 15 хвилин.
- **Сертифікат ACM для CloudFront не в us-east-1** — для CloudFront — лише us-east-1.
- **EFS для Windows** — EFS лише для Linux; для Windows — FSx for Windows File Server.
- **SCP, що «дає» права** — SCP лише обмежує.

> 💡 Ще одне правило: варіант «напишемо власний скрипт на EC2 і запускатимемо за розкладом» майже завжди програє готовому керованому сервісу.

Фрази-підказки

Фраза в питанні	Що означає	Типова відповідь
MOST cost-effective / LOWEST cost	найдешевше з того, що виконує **всі** вимоги	Spot, lifecycle, serverless, Savings Plans, gateway endpoint
LEAST operational overhead	мінімум турбот	керований чи serverless-сервіс, без власних скриптів
MOST secure / least privilege	мінімум прав	ролі замість ключів, KMS, приватні підмережі, endpoints
highly available	переживе збій зони	кілька AZ: Multi-AZ, ASG + ALB
fault tolerant	працює взагалі без перерви	повне дублювання, active-active
durable	дані не загубляться	S3, реплікація, бекапи
decouple / buffer / spikes	розв'язати частини	SQS
real-time, кілька споживачів, replay	потік з пам'яттю	Kinesis Data Streams
near real-time у S3 без коду	трубопровід	Data Firehose

Фраза в питанні	Що означає	Типова відповідь
without code changes	рішення ззовні застосунку	RDS Proxy, Storage Gateway, EFS, Global Accelerator, MGN
temporary credentials	тимчасові ключі	IAM Role + STS
static IP	постійна адреса	NLB з Elastic IP або Global Accelerator
orchestrate / workflow / human approval	процес із кроками	Step Functions
migrate TBs over the network	переїзд мережею	DataSync

Старі назви в питаннях

Питання оновлюють не одразу, тож можуть траплятися старі назви. Не лякайся — це ті самі сервіси:

- Kinesis Data Firehose = **Amazon Data Firehose**;
- QuickSight = **Amazon Quick**;
- Kinesis Data Analytics = **Managed Service for Apache Flink**;
- SageMaker = **SageMaker AI**;
- AWS SSO = **IAM Identity Center**;
- CloudWatch Events = **EventBridge**.

Сім розборів

Кожен розбір побудований однаково: що сказано, що вимагається, що викреслюємо, що обираємо. Спершу спробуй відповісти без підказок, а потім звір.

1. Журнали на сім років. «Logs are read often for the first 30 days. After that they are needed only for rare audits, a wait of several hours is acceptable, and they must be kept for 7 years at the LOWEST cost.»

- Вимоги: 30 днів — часто, потім — рідко й можна чекати години, 7 років, найдешевше.
- Викреслюємо: лишити все в S3 Standard (найдорожче); Intelligent-Tiering (він для **невідомого** патерну, а тут патерн відомий); One Zone-IA (ризик втратити разом із зоною).
- ✅ **S3 Standard + правило lifecycle: через 30 днів — у S3 Glacier Deep Archive**, видалити через 7 років.

2. Фото з телефона без ключів. «Mobile app users must upload photos directly to a private S3 bucket. AWS credentials must not be stored in the app.»

- Вимоги: прямо в S3, сховище приватне, жодних ключів у застосунку.
- Викреслюємо: зробити сховище публічним; вписати ключі в застосунок; сервер-посередник на EC2 (зайва робота й вузьке місце).
- ✅ **Presigned URL**: бекенд видає застосунку тимчасове посилання на завантаження одного файлу.

3. Оновлення для приватних серверів. «EC2 instances in private subnets must download software updates from

the internet. They must not be reachable from the internet.»

- Вимоги: вихід в інтернет є, входу ззовні немає.
- Викреслюємо: маршрут з приватної підмережі напряму на Internet Gateway (підмережа перестане бути приватною); gateway endpoint (лише S3 і DynamoDB); VPC peering (це не інтернет).
- ✅ **NAT Gateway у публічній підмережі** + маршрут `0.0.0.0/0` на нього з приватної.

4. Звіти гальмують базу. «Reporting queries slow down the production Amazon RDS database. The company wants to offload the reports with minimal changes.»

- Вимоги: розвантажити читання, мінімум змін.
- Викреслюємо: читати з дублера Multi-AZ (він не обслуговує читання); більший сервер (дорожче й не цілить у причину); ElastiCache (потребує змін у коді, а звіти — важкі неповторні запити).
- ✅ **Read replica** — звіти йдуть на її адресу.

5. Гра з усього світу. «A multiplayer game uses UDP. Players around the world need low latency, static IP addresses, and fast failover between Regions.»

- Вимоги: UDP, увесь світ, статичні IP, швидке перемикання регіонів.

- Викреслюємо: CloudFront (він для вебконтенту й кешу, а не для UDP-гри); Route 53 latency (DNS кешується, статичних IP немає); один NLB (лише в одному регіоні).
- ✅ **Global Accelerator** — дві статичні адреси, вхід у мережу AWS біля гравця, перемикання за секунди.

6. Ніхто не вимкне журнали. «Administrators in member accounts of an organization must never be able to disable AWS CloudTrail.»

- Вимоги: заборона для всіх, навіть для адміністраторів акаунтів.
- Викреслюємо: політика IAM у кожному акаунті (адміністратор її змінить); правило AWS Config (помітить, але вже після); permission boundary (адміністратор її зніме).
- ✅ **SCP в AWS Organizations** — «конституція», яку не обійти зсередини акаунта.

7. Оберіть ДВА. «A web application runs on one EC2 instance. Which TWO actions will make it highly available? (Choose two.)»

- Варіанти: більший тип інстансу; Elastic IP; **група Auto Scaling у двох AZ**; **Application Load Balancer**; cluster placement group.

- Викреслюємо: більший сервер (усе одно один); Elastic IP (адреса не робить сервер надійнішим); cluster placement group (збирає сервери **в одній** зоні — доступність навіть гірша).
- ✅ **Auto Scaling у двох AZ** + **ALB**. У питаннях «оберіть ДВА» кожна правильна відповідь — окрема частина рішення.

День іспиту

Запис. У кабінеті сертифікації (aws.training → Certification) обираєш іспит і спосіб: **центр Pearson VUE** чи **онлайн удома** (OnVUE). Але спершу — ESL +30.

Документ. Паспорт чи інше посвідчення з фото. Ім'я має **точно збігатися** з ім'ям у профілі AWS Certification.

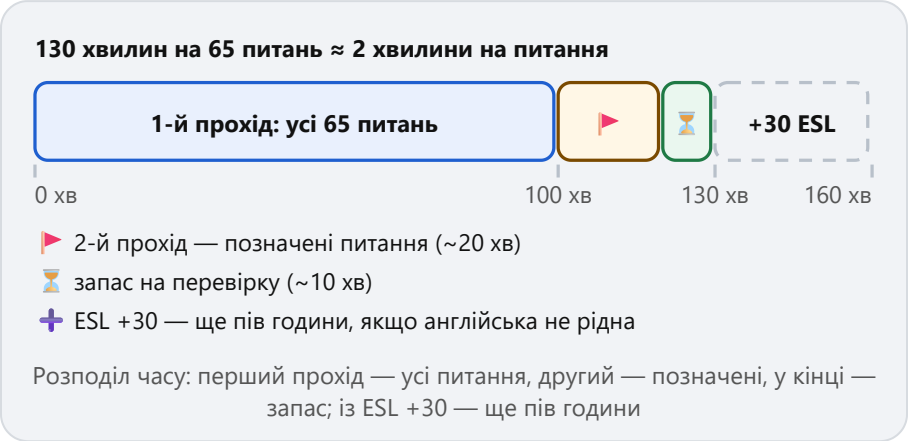
Онлайн-іспит:

- заздалегідь пройди **перевірку системи** (system test) — на тому самому комп'ютері й у тій самій мережі;
- check-in відкривається **за 30 хвилин** до початку; кімнату фотографуєш під час check-in;
- потрібні окрема тиха кімната, чистий стіл, телефон поза досяжністю;
- перерв, як правило, **немає** — подбай про все заздалегідь;

- говорити вголос і виходити з кадру **не можна**; нотатки — лише на вбудованій цифровій дошці (whiteboard).

У центрі тестування приходь раніше: речі залишаєш у шафці, для нотаток дають дошку.

Час. 130 хвилин на 65 питань — приблизно дві хвилини на питання:



- **Перший прохід:** відповідай на **всі** питання підряд. Сумнівні — познач прапорцем, але відповідь однаково обери.
- **Другий прохід:** повернися до позначених зі свіжою головою.
- **Запас:** переконайся, що порожніх відповідей немає.

Результат приходить на пошту й у кабінет — зазвичай за кілька годин, іноді до 5 робочих днів. Одразу після іспиту

бал не показують, тож не хвилюйся, якщо на екрані не буде цифри.

> 🥐 Оля записала все в блокнот і того ж вечора подала запит на ESL +30. «Спершу +30 хвилин, потім запис, — повторює вона. — Останнє речення, вимоги, викреслити неможливе. І жодного порожнього питання». Тарас усміхається: «Ти вже думаєш як архітекторка».

📖 **Словничок**

SAA-C03 — іспит AWS Certified Solutions Architect – Associate.

Оцінювані / неоцінювані питання — 50 рахуються, 15 — ні (невідомо, які).

Компенсаторна оцінка — рахується загальний бал, а не кожен домен окремо.

Multiple choice / multiple response — одна правильна з 4 / кілька з 5+.

ESL +30 — додаткові 30 хвилин, якщо англійська не рідна; подати до запису.

Pearson VUE / OnVUE — центри тестування / онлайн-іспит удома.

Проктор — наглядач, що стежить за онлайн-іспитом через камеру.

Flag for review — позначка «повернутися пізніше».

Критерій — вимога з останнього речення: MOST cost-effective, LEAST operational overhead…

🧩 Міні-тест: перевір себе

? Скільки балів знімають за неправильну відповідь на SAA-C03?

А. Чверть бала

Б. Один бал

В. Жодного: штрафу немає, тож відповідати треба на всі питання

Г. Залежить від домену

Відповідь: В. Порожня відповідь — гарантовано нуль, а спроба — шанс. Відповідай на все.

? Який прохідний бал іспиту?

А 650 з 1000

Б 720 з 1000

В 800 з 1000

Г 70% у кожному домені

Відповідь: Б. 720 з 1000, і рахується загальний бал, а не кожен домен окремо.

? Коли треба подати запит на ESL +30?

А До запису на іспит; якщо запис уже є — скасувати, подати запит і записатися знову

Б Під час check-in

В Після іспиту, щоб перерахували бал

Г У будь-який момент, хвилини додадуть автоматично

Відповідь: А. Запит подають один раз і до запису — інакше додаткового часу не буде.

? Де в питанні найчастіше захований критерій вибору?

- **А** У першому реченні
- **Б** В останньому реченні
- **В** У варіантах відповіді
- **Г** У назві домену

Відповідь: Б. Останнє речення: MOST cost-effective, LEAST operational overhead, MOST secure…

? Який варіант можна викреслити одразу, навіть не дочитавши питання?

- **А** Додати read replica
- **Б** Додати правило Deny у Security Group
- **В** Поставити ALB перед серверами
- **Г** Увімкнути шифрування KMS

Відповідь: Б. У Security Group немає правил Deny — заборони ставлять у NACL чи WAF.

? У питанні написано «Choose two». Скільки варіантів обрати?

А Один, найкращий

Б Рівно два

В Усі правильні, скільки знайдеш

Г Три, для надійності

Відповідь: Б. Скільки обрати, завжди сказано в питанні; кожна правильна відповідь — частина рішення.

? Звіти уповільнюють робочу базу RDS. Треба розвантажити читання з мінімальними змінами. Що обрати?

- **А** Читати з дублера Multi-AZ
- **Б** Read replica
- **В** Більший тип інстансу бази
- **Г** ElastiCache перед кожним звітом

Відповідь: Б. Дублер Multi-AZ не обслуговує читання, а read replica — саме для цього.

? У питанні згадано «Kinesis Data Firehose». Що це?

- **А** Окремий сервіс, якого вже немає
- **Б** Стара назва Amazon Data Firehose
- **В** Новий режим Kinesis Data Streams
- **Г** Частина Amazon MSK

Відповідь: Б. Сервіс перейменували, а питання могли ще не оновити. Відповідай за суттю.

✅ **Головне**

65 питань, 130 хвилин, 720 з 1000, $150, штрафу немає — відповідай на все. Домени: безпека **30%**, надійність **26%**, продуктивність **24%**, вартість **20%**.

Української немає — терміни вчимо англійською. **ESL +30** — подати **до запису**.

Питання = **контекст + вимоги + критерій (останнє речення) + варіанти**.

П'ять кроків: **останнє речення** → **вимоги** → **викреслити неможливе** → **керований варіант, що виконує все** → **flag, якщо сумніваєшся**.

Неможливе: Deny у SG, WAF на NLB, gateway endpoint не для S3/DynamoDB, читання з Multi-AZ standby, EBS в іншій AZ, Lambda довше 15 хвилин.

Два проходи по часу; результат — на пошту за кілька годин.

🏁 Що далі: від книжки до сертифіката — і після нього

👴 Простими словами

Книжка прочитана — далі закріплюєш у коротких уроках, шпаргалці й тестах, а потім ідеш на іспит. І після нього практика в справжньому AWS — найкращий учитель.

🎯 Що ти дізнаєшся

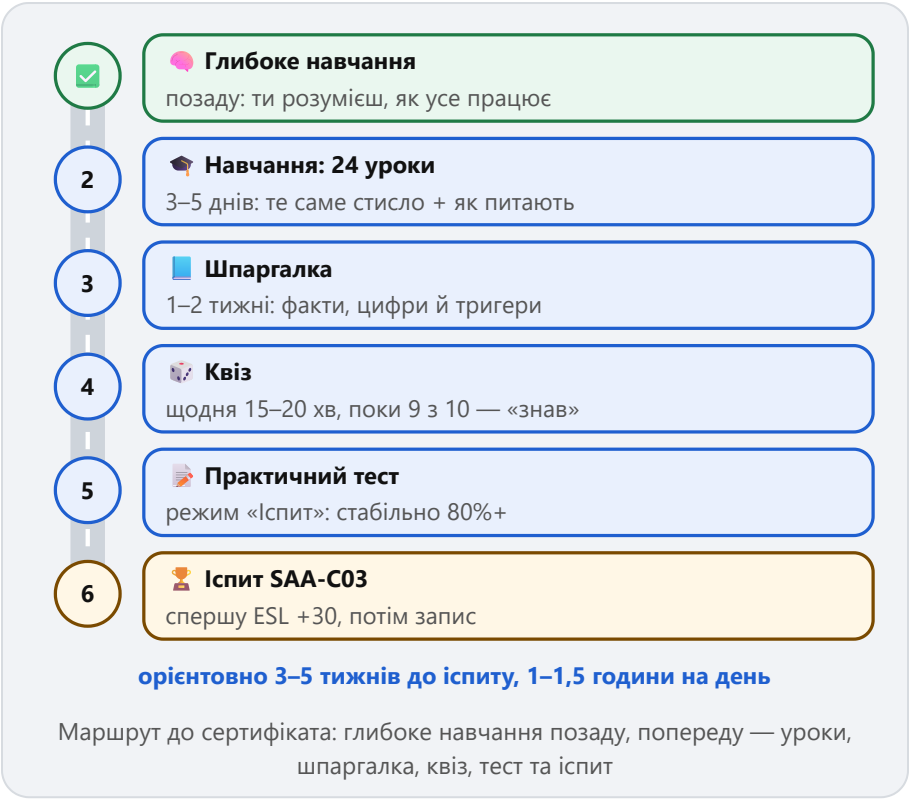
- що вже позаду і що лишилося на маршруті
- як працювати з уроками, шпаргалкою, квізом і тестом
- план на 14 днів перед іспитом
- як зрозуміти, що час записуватися
- як перетворювати помилки на бали
- що робити в останній тиждень, останній вечір — і якщо з першого разу не вийшло
- куди рости після сертифіката

🥐 Оля закриває останню сторінку конспекту й питає: «А тепер що? Я все прочитала — але чи означає це, що я складу?» Тарас ставить на стіл тарілку з круасанами: «Пам'ятаєш, як ти вчилася пекти? Спершу зрозуміла, чому тісто піднімається. А потім пекла знову і знову, поки руки не запам'ятали. Розуміння в тебе вже є. Тепер — тренування».

Що вже позаду

Позаду — найважча частина. Ти знаєш, що всередині комп'ютера і як пакети мандрують мережею. Що таке хмара і як влаштований AWS. Як захистити акаунт і дані, збудувати мережу, сервери, сховища й бази. Як течуть потоки даних і що вміють готові AI-сервіси. Як стежити за системою, пережити аварію й не переплатити. І як мислить архітектор, коли читає питання іспиту.

Але **розуміти** і **швидко впізнавати** — різні вміння. На іспиті на кожне питання — близько двох хвилин. Решта маршруту саме про це: перетворити розуміння на швидкий, упевнений вибір.



Як пройти решту маршруту

- 🎓 **Навчання.** 24 короткі уроки по 5–8 хвилин. Тепер вони читатимуться як повторення знайомого: ті самі образи, тільки стисло, плюс блок «🎯 Як питають на іспиті». Прикрий його рукою й спробуй відповісти, не підглядаючи.

- 📘 **Шпаргалка.** Прочитай розділ, потім увімкни «🙈 Сховати відповіді» й пройди тригери: спершу відповідай подумки і лише тоді торкайся розмитої відповіді. Позначай «Вивчено».
- ⚡ **Швидко** — лише тригери, пастки, порівняння й цифри. Для останніх днів перед іспитом.
- 🗺️ **Схеми** — усі архітектурні схеми в одному місці.
- 🎲 **Квіз** — випадкові тригери. Ті, що «не знав», повертаються частіше, — так пам'ять закріплює саме слабкі місця. Ціль — 9 з 10.
- 📝 **Тест** — 65 питань у стилі іспиту. Режим «Тренування» показує пояснення одразу, режим «Іспит» — з таймером і результатом у кінці.

Прогрес зберігається в браузері цього пристрою. Сайт працює й без інтернету, а на телефоні його можна додати на головний екран як застосунок. І не забувай про 🤖 **підказки**: торкнись підкресленої назви сервісу — з'явиться коротке пояснення й посилання на розділ цієї книжки.

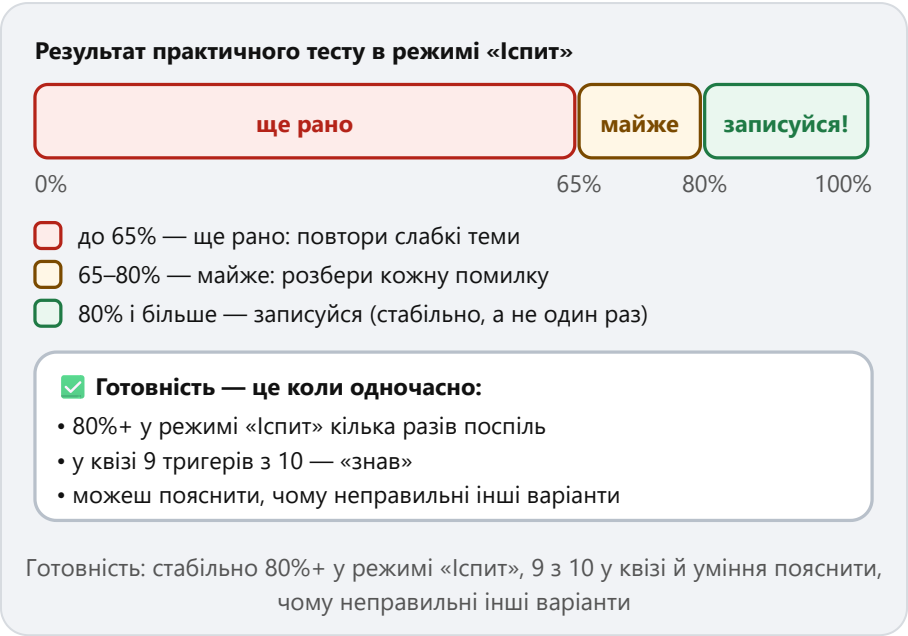
План на 14 днів

Півтори-дві години на день — і за два тижні шпаргалка, квіз і тест складуться в систему. Номери — це розділи шпаргалки.

День	Що вчити	Практика
1	Розділи 1–3: іспит, тактика, основи	Квіз: 20 тригерів
2	4–5: IAM, шифрування	Квіз по розділах 4–5
3	6: сервіси безпеки	Тест: набір «Безпека», режим «Тренування»
4	7: VPC і мережа	Квіз по розділу 7
5	8–9: Route 53, CloudFront, Global Accelerator, ELB, Auto Scaling	Квіз по розділах 8–9
6	10–11: EC2, serverless, контейнери	Квіз по розділах 10–11
7	12: SQS, SNS, EventBridge, Step Functions	« ⚡ Швидко» по днях 1–6
8	13–14: S3, EBS, EFS, FSx	Тест: набір «Продуктивність»
9	15–17: міграція, RDS, Aurora	Квіз по розділах 15–17
10	18–19: DynamoDB, кеш, інші бази	Тест: набір «Відмовостійкість»
11	20–22: аналітика, ML, моніторинг	Квіз по розділах 20–22
12	23–25: DR, вартість, цифри	Тест: набір «Вартість»
13	Повний тест у режимі «Іспит» (130 хвилин)	Розібрати **кожну** помилку
14	26–28: порівняння, пастки, зміни	«Мої помилки» в тесті + « ⚡ Швидко»

Додатково варто пройти безкоштовний офіційний набір питань AWS (**Official Practice Question Set**) в **AWS Skill Builder**. Популярні платні тести — Tutorials Dojo.

Як зрозуміти, що час записуватися



Один вдалий тест — ще не сигнал: може, пощастило з питаннями. Сигнал — коли **80% і більше** виходить **кілька разів поспіль**, у квізі 9 тригерів з 10 — «знав», а для кожного питання можеш пояснити не лише «чому ця», а й «чому не інші». Тоді — ESL +30, запис, іспит.

Помилки — найцінніше в тесті

- Розбирай **кожну** помилку: чому правильна відповідь правильна і чому обрана — ні.

- Веди **зошит помилок** у три колонки: **тема** → **пастка** → **правило**. Наприклад: «RDS → читання з дублера Multi-AZ → дублер не читається, для читання — read replica».
- Якщо незрозуміло саме «чому», — повернися до розділу цієї книжки на цю тему. Розуміння тут, а не в шпаргалці.
- Через день-два повтори «Мої помилки» в тесті: мають залишитися лише нові.

> 💡 Помилка в практичному тесті коштує нуль. Та сама помилка на іспиті — 150 доларів. Тож радій кожній помилці на тренуванні.

Останній тиждень, вечір і ранок

- **Останній тиждень:** « ⚡ Швидко», розділи шпаргалки 25 (цифри), 26 («X чи Y?»), 27 (пастки) і 28 (що змінилося). Щонайменше один повний тест у режимі «Іспит».
- **Останній вечір:** нічого нового не вчи. Переглянь «Головне» кількох розділів, підготуй документ з фото. Для онлайн-іспиту перевір комп'ютер і кімнату. І виспися — свіжа голова дає більше балів, ніж ще одна година зубріння.

- **Ранок:** легкий сніданок і вода. На онлайн-іспит — check-in за 30 хвилин, у центр тестування — прийди раніше.
- **Під час іспиту:** незнайоме питання — не привід для паніки. Вимоги й критерій однаково підкажуть напрям, а неможливі варіанти викреслюються навіть без знання сервісу.

Якщо з першого разу не вийшло

Таке трапляється й з досвідченими інженерами. Це не вирок, а інформація.

- У звіті про результат видно, які домени вийшли слабшими, — з них і починай.
- Повторна спроба — не раніше ніж через **14 днів**, і її оплачують знову.
- Перечитай розділи цієї книжки зі слабких доменів, пройди їхні міні-тести, потім знову практичний тест — і вперед.

Після сертифіката

- Сертифікат діє **3 роки**. Після здачі в кабінеті з'являється **знижка 50%** на наступний іспит AWS.

- Знання без практики тануть. Будуй невеликі проєкти у власному акаунті — з бюджетом і MFA! Наприклад, збери справжню «Хмаринку»: статичний сайт у S3 за CloudFront, API на Lambda й DynamoDB, а потім опиши все це шаблоном CloudFormation.
- Наступні сходинки — рівень Professional (**AWS Certified Solutions Architect – Professional**) і спеціалізації, наприклад з безпеки чи мереж.
- Найкращий спосіб закріпити знання — пояснити іншому. Розкажи комусь, як працює хмара, — так само, як ця книжка розповідала тобі.

> 🥐 Через два місяці Тарас отримав від Олі знімок листа від AWS — з привітанням і новим званням: AWS Certified Solutions Architect – Associate. Під знімком — підпис: «Рік тому я не знала, що таке сервер. А сьогодні знаю, як збудувати пекарню, якій не страшні ні Чорна п'ятниця, ні пожежа у Франкфурті». Тарас відповів одним словом: «Архітекторка».

Тепер твоя черга. Позаду той самий шлях, який пройшла Оля, — від «а що взагалі всередині комп'ютера?» до

архітектури, яка витримує мільйони клієнтів. Залишилося тренування — і впевнений вибір на іспиті. Успіху!

📖 Словничок

Тригер — пара «бачиш у питанні → відповідь».

Режим «Тренування» / «Іспит» — пояснення одразу / таймер і результат у кінці.

Зошит помилок — тема → пастка → правило.

Official Practice Question Set — безкоштовний офіційний набір питань в AWS Skill Builder.

Повторна спроба — не раніше ніж через 14 днів, оплачується знову.

Professional — наступний рівень сертифікації архітектора.

🧩 **Міні-тест: перевір себе**

? Глибоке навчання завершене. Який наступний крок маршруту?

- **А** 24 короткі уроки «Навчання»
- **Б** Одразу записатися на іспит
- **В** Ще раз перечитати всю книжку
- **Г** Почати з повного тесту в режимі «Іспит»

Відповідь: А. Уроки повторюють ті самі образи стисло й додають «як питають на іспиті» — міст до шпаргалки й тестів.

? Коли час записуватися на іспит?

А Після першого тесту, де вийшло 80%

Б Коли 80%+ у режимі «Іспит» виходить кілька разів поспіль, а квіз — 9 з 10

В Коли прочитано всю шпаргалку

Г Щойно з'явилося вільне місце в розкладі

Відповідь: Б. Один вдалий тест може бути везінням. Сигнал — стабільність.

? Що робити з помилкою в практичному тесті?

А Запам'ятати правильну літеру

Б Розібрати, чому правильна відповідь правильна, а обрана — ні, і за потреби повернутися до розділу книжки

В Пропустити: на іспиті буде інше питання

Г Пройти тест ще раз одразу, поки пам'ятаються відповіді

Відповідь: Б. Літери на іспиті будуть інші, а от правило з помилки знадобиться.

? Що найкорисніше зробити в останній вечір перед іспитом?

А Вивчити нову тему, яку пропущено

Б Пройти три повні тести поспіль

В Переглянути «Головне», підготувати документ і виспатися

Г Не спати й повторювати всю шпаргалку

Відповідь: В. Свіжа голова дає більше балів, ніж нічне зубріння.

? Іспит не вдалося скласти з першого разу. Коли можна спробувати знову?

- А Наступного дня
- Б Не раніше ніж через 14 днів
- В Лише через рік
- Г Тільки після окремого дозволу AWS

Відповідь: Б. Повторна спроба — щонайменше через 14 днів; зосередься на слабких доменах зі звіту.

? Навіщо квіз частіше повертає тригери, позначені «не знав»?

- А Щоб тест тривав довше
- Б Щоб пам'ять закріплювала саме слабкі місця
- В Щоб перевірити швидкість
- Г Це випадковість

Відповідь: Б. Повторення слабкого — найкоротший шлях до 9 з 10.

✅ **Головне**

Розуміння вже є — тепер **тренування**: уроки → шпаргалка → квіз → тест → іспит.

План на 14 днів: щодня розділи шпаргалки + квіз або набір тесту; день 13 — повний тест у режимі «Іспит».

Записуватися — коли **стабільно 80%+** у режимі «Іспит», 9 з 10 у квізі й можеш пояснити, чому неправильні інші варіанти. Спершу — **ESL +30**.

Кожна помилка — у зошит: тема → пастка → правило; незрозуміле «чому» — назад до розділу книжки.

Останній вечір — нічого нового, «Головне» і сон. Не вийшло — повтор через **14 днів**, починаючи зі слабких доменів.

Після сертифіката — практика, рівень Professional і пояснення іншим.