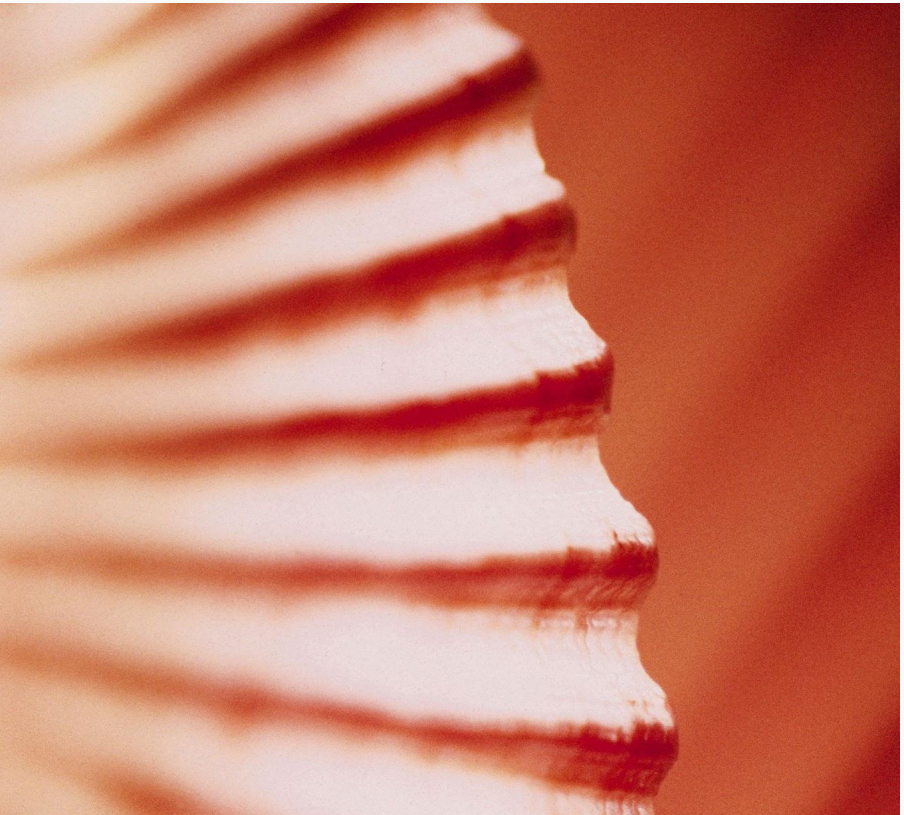
Хмарні провайдери

Agenda



01 Що таке хмарні обчислення?

02 AWS, Azure, GCP. Чому AWS?

03 Фундамент AWS

04 Стратегії розгортання в AWS

05 Deployment на практиці

06 Автоматизація з CI/CD

Моделі хмарних сервісів

Infrastructure as a Service (IaaS)

- Що це? Оренда базових обчислювальних ресурсів: віртуальних машин, сховищ, мереж.
- Ви керуєте: ОС, проміжним ПЗ, застосунком.
- Приклад: Amazon EC2, DigitalOcean Droplets.
- Аналогія: Оренда земельної ділянки. Ви самі будуєте дім.

Platform as a Service (PaaS)

- Що це? Платформа для розробки та розгортання застосунків без управління інфраструктурою.
- Ви керуєте: Лише застосунком та його даними.
- Приклад: AWS Elastic Beanstalk, Heroku, Google App Engine.
- Аналогія: Оренда готового будинку. Ви лише завозите меблі.

Software as a Service (SaaS)

- Що це? Готове програмне забезпечення, доступне через інтернет.
- Ви керуєте: Нічим, окрім свого акаунту.
- Приклад: Gmail, Dropbox, Slack.
- Аналогія: Проживання в готелі. Ви просто користуєтесь сервісом.

Великі гравці та домінування AWS

Amazon Web Services (AWS)

Запуск: 2006 рік.

Сильні сторони: Найширший портфель сервісів, найбільша ринкова частка, величезна спільнота та детальна документація. Піонер ринку.

Microsoft Azure

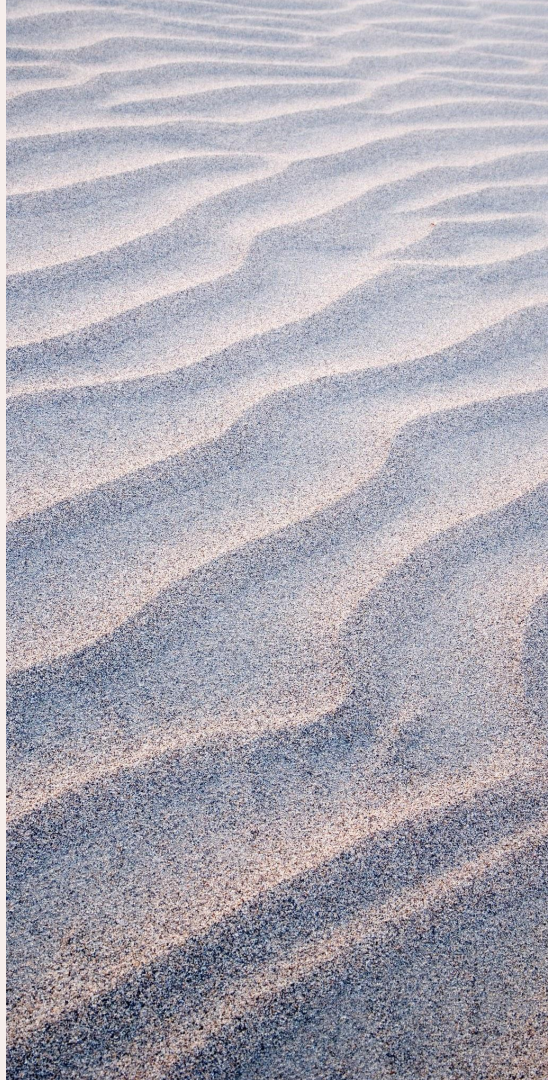
Запуск: 2010 рік.

Сильні сторони: Глибока інтеграція з корпоративними продуктами Microsoft (.NET, Office 365), сильні позиції в гібридних хмарах.

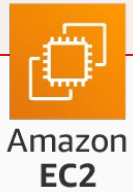
Google Cloud Platform (GCP)

Запуск: 2008 рік.

Сильні сторони: Експертиза в Kubernetes (вони його створили), аналітиці даних (BigQuery) та машинному навчанні.



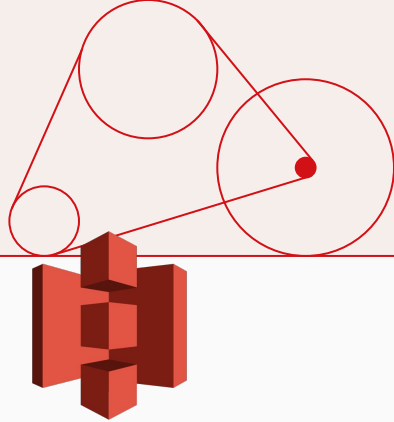
Фундамент AWS: EC2 та S3



Amazon EC2 (Elastic Compute Cloud)

Віртуальні сервери в хмарі. Основа основ AWS

- AMI (Amazon Machine Image): Шаблон для запуску інстансу (ОС + ПЗ).
- Instance Types: Різні конфігурації CPU, пам'яті, мережі (напр., t3.micro, m5.large).
- Security Groups: Віртуальний фаєрвол для контролю трафіку.
- EBS (Elastic Block Store): "Жорсткий диск" для вашого EC2 інстансу



Amazon S3 (Simple Storage Service)

Майже нескінченне, високодоступне сховище для будь-яких файлів (об'єктів).

- Buckets (Кошики): Контейнери для зберігання об'єктів з унікальним глобальним іменем.
- Objects (Об'єкти): Файли та їхні метадані.
- Використання: Зберігання статичних файлів для вебсайтів, бекапів, логів, даних для аналітики.

Фундамент AWS: RDS та VPC

Amazon RDS (Relational Database Service)

amazon RDS

- Що це? Керована служба реляційних баз даних.
- Навіщо? AWS бере на себе рутинні задачі: встановлення, патчі, бекапи, реплікацію. Ви просто користуєтесь базою.
- Підтримувані рушії: PostgreSQL, MySQL, MariaDB, Oracle, SQL Server.
- Це приклад PaaS для баз даних.

Amazon VPC (Virtual Private Cloud)

- Що це? Ваша приватна, ізольована ділянка в хмарі AWS.
- Навіщо? Дозволяє створювати власну мережеву топологію, контролювати IP-адреси, підмережі, таблиці маршрутизації та шлюзи.

Ключові поняття:

- Subnets: Публічні (з доступом до інтернету) та приватні (без прямого доступу).
- Security Groups / NACLs: Рівні безпеки на рівні інстансу та підмережі.

Стратегії розгортання в AWS

All-at-once" (In-place deployment)

- Процес: Зупинити стару версію, розгорнути нову на тих же серверах, запустити.
- Переваги: Просто та швидко.
- Недоліки: Потребує простою (downtime). Складний відкат.

Rolling Deployment

- Процес: Оновлення відбувається поступово, на частині серверів за раз.
- Переваги: Без простою. Менший ризик, оскільки проблема торкнеться лише частини користувачів.
- Недоліки: Тимчасово існують дві версії застосунку.

Blue/Green Deployment

- Процес: Створюється точна копія продакшен-середовища ("Green"). Нова версія розгортається на "Green". Після тестування трафік миттєво перемикається з "Blue" на "Green".
- Переваги: Миттєвий відкат (просто перемкнути трафік назад).
- Недоліки: Вимагає подвійних ресурсів, що дорожче.

Deployment на практиці: AWS Elastic Beanstalk

Як це працює?

1. Ви створюєте Application в Elastic Beanstalk.
2. Завантажуєте свій код (zip-архів або через CLI).
3. Обираєте платформу (Node.js, Python, Go, Docker).
4. Elastic Beanstalk автоматично створює все необхідне:
   - EC2 інстанси.
   - Auto Scaling Group (для масштабування).
   - Elastic Load Balancer (для розподілу трафіку).
   - Security Groups.
   - Моніторинг та логування.
5. Ви отримуєте готовий URL вашого застосунку.

Переваги

Простота: Мінімальні знання інфраструктури.
Швидкість: Розгортання за хвилини.
Вбудовані інструменти: Масштабування, моніторинг, оновлення.

Недоліки

Менше гнучкості: "Чорна скринька", важко кастомізувати.

Підходить для стандартних веб-застосунків.

Deployment на практиці: AWS ECS

Ключові компоненти

- Task Definition: "Рецепт" для запуску контейнера (аналог секції template.spec в Deployment K8s). Описує образ, CPU, пам'ять, порти.
- Task: Запущений екземпляр Task Definition (аналог Pod в K8s).
- Service: Керує довготривалим запуском та масштабуванням тасків. Забезпечує бажану кількість запущених тасків та інтегрується з Load Balancer.
- Cluster: Логічне групування ресурсів, де запускаються ваші таски.

Режими запуску (Launch Types)

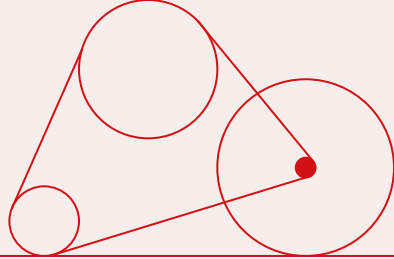
- EC2: Ви керуєте кластером з EC2-інстансів. Більше контролю, але й більше роботи.
- Fargate: "Серверний" режим. Ви просто запускаєте контейнери, а AWS керує інфраструктурою. Простіше, але дорожче.

AWS ECS

Task Definition для ECS

```
{
  "family": "my-nodejs-app-task",
  "containerDefinitions": [
    {
      "name": "my-nodejs-container",
      "image": "your-account-id.dkr.ecr.region.amazonaws.com/my-nodejs-app:latest"
      "cpu": 256,
      "memory": 512,
      "portMappings": [
        {
          "containerPort": 3000,
          "hostPort": 3000,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/my-nodejs-app",
          "awslogs-region": "eu-central-1",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
      "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "256",
  "memory": "512"
}
```

Автоматизація з CI/CD в AWS



Набір інструментів AWS "Code"

- AWS CodeCommit: Керований Git-репозиторій. (Альтернатива: GitHub, GitLab)
- AWS CodeBuild: Сервіс для збірки та тестування коду. Збирає Docker-образи.
- AWS CodeDeploy: Автоматизує процес розгортання на EC2, ECS, Lambda.
- AWS CodePipeline: "Оркестратор", що об'єднує всі етапи в єдиний пайплайн.

Етапи пайплайну для ECS

- **Source**: git push в CodeCommit або GitHub. CodePipeline реагує на зміни.
- **Build**: CodeBuild запускає тести, збирає Docker-образ (docker build), тегує його і завантажує в Amazon ECR.
- **Deploy**: CodeDeploy бере новий образ, оновлює ECS Service і запускає розгортання за обраною стратегією (напр., Blue/Green).

Переваги та виклики AWS

Переваги

Гнучкість та масштабованість: Платіть лише за те, що використовуєте, і масштабуйтесь від одного користувача до мільйонів.
Широкий спектр сервісів: AWS має інструмент практично для будь-якої задачі.
Надійність та безпека: Глобальна інфраструктура з високими стандартами безпеки.
Величезна спільнота: Легко знайти документацію, туторіали та відповіді на Stack Overflow.

Виклики

Складність: Величезна кількість сервісів може спантеличити початківця.
Вартість: Без належного контролю та оптимізації витрати можуть швидко вийти з-під контролю ("bill shock").
Vendor lock-in: Глибоке використання специфічних сервісів AWS може ускладнити перехід до іншого провайдера.