EC2

- 뛰어난 유연성, 비용 효율성, 빠른 속도
- 온프레미스 방식보다 유연함 (물리적인 서버 구축, 비용 절감)

가상화

물리적 호스트 시스템에서 실행 -> 가동 시 소유하지 않아도 사용 가능

1. 가상 머신

- 여러 인스턴스와 호스트를 공유하여 사용하여 `물리적 공간 절약`

2. 하이퍼바이저

- 호스트 머신에서 실행됨
- 가상 머신끼리 물리적인 리소스 공유(`멀티 테넌시`)하는 것을 책임
- 가상머신 분리 -> EC2 인스턴스가 격리되어 안전함 (다른 인스턴스 인식 x)

작동 방식

1. 시작

- `인스턴스 시작`
    - OS, 애플리케이션 서버, 애플리케이션 포함
    - 인스턴스 유형(특정 하드웨어 구성) 선택
    - 네트워크 트래픽 제어 보안 설정
- `연결`
    - 프로그램 - 애플리케이션 : 직접 연결
    - 사용자 로그인 후 연결 및 액세스
- `사용`
    - 명령 실행, SW 설치, 스토리지 추가, 파일 복사 및 정리
    -

EC2 인스턴스 유형

- `인스턴스 패밀리`로 구현, 특정 작업에 최적화
- cpu, 메모리, 스토리지, 네트워크 용량 조합 가능

1. 범용 인스턴스

- 컴퓨팅, 메모리, 네트워크 리소스를 균형있게 제공
- 한 영역에 대한 최적화 필요 없는 경우

사용 예

- 애플리케이션 서버
- 게임 서버
- 엔터프라이즈 앱 백엔드 서버
- 중소 규모 DB

2. 컴퓨팅 최적화 인스턴스

- 고성능 프로세서 활용하는 컴퓨팅 집약적 애플리에킹션

사용 예

- 고성능 웹 서버
- 컴퓨팅 집약적 앱 서버
- 게임 전용 서버
- 일괄 처리 워크로드

3. 메모리 최적화 인스턴스

- 메모리에서 대규모 세트를 처리하는 워크로드
- 프로그램 실행을 위해 많은 데이터를 로드해야하는 경우

사용 예

- 고성능 DB
- 비정형 데이터를 실시간 처리하는 경우

4. 액셀러레이티드 컴퓨팅 인스턴스

- 하드웨어 액셀러레이터 / 코프로세서 사용
  -> 일부 기능을 `cpu보다 효율적으로 수행`
    - 데이터 처리 가속화

사용 예

- 부동 소수점 수 계산
- 그래픽 처리
- 데이터 패턴 일치
- 게임 스트리밍
- 애플리케이션 스트리밍

5. 스토리지 최적화 인스턴스

- 로컬 스토리지의 대규모 데이터에 대한 순차적 읽기 / 쓰기 액세스가 많은 워크로드
- 지연 시간이 짧은 임의 IOPS(초당 입출력 작업 수) 제공

사용 예

- 분산 파일 시스템
- 데이터 웨어하우징 애플리케이션
- 고빈도 온라인 트랜잭션 처리 (OLTP)

EC2 요금제

1. 온디맨드 요금제

- `중단 불가능`, `불규칙한 단기 워크로드`
- 인스턴스 `실행 기간` 동안의 요금 지불
- 서비스를 가동하고 테스트하는 경우 사용
- ex) 애플리케이션 개발 및 테스트, 예측불가능한 사용 패턴의 애플리케이션 실행

2. Saving Plan

- `일정 사용량을 약정`하는 대가로 저렴하게 제공
- 사용 시간 중 약정 용량만큼 할인된 비용 청구
- `AWS Fargate`와 `AWS Lambda` 에도 적용 가능

3. 예약 인스턴스

- 꾸준한 사용 / 예측 가능한 워크로드에 적합
- 약정 + 할인 결합

종류

- 전액 선결제 : 약정 시점에 전액 지불
- 부분 선결제 : 약정 시 일부 지불
- 선결제 없음 : 후불 결제

4. 스팟 인스턴스

- AWS가 필요 시 인스턴스 용량 회수
- 워크로드에서 중단 가능한지 확인 필수
- ex) 배치 워크로드

5. 전용 호스트

- 물리적 호스트를 전용으로 사용 가능
- 서버 머신 공유 x

EC2 확장 - 필요한 리소스만으로 시작하고, 확장 및 축소를 통해 수요 변화에 `자동 대응` 하도록 설계 - 수직 확장과 수평 확장으로 처리

- `수직 확장` : 실행중인 장치에 성능 추가
- `수평 확장` : 시스템을 분리해서 프로세스의 부분에 적절한 수준의 성능 제공

Amazon EC2 Auto Scaling

- 수요에 따라 `인스턴스를 자동으로 추가 및 제거` rksmd
- 가용성 효과적으로 유지

작동 방식

- 최소 EC2 인스턴스 설정 가능

구성

- `최소 용량` : 생성 직후 시작되는 인스턴스 수 (1개)
- `희망 용량` : 증설할 인스턴스 수
- `최대 용량` : 수요 증가에 대응하도록 구성하며 최대 인스턴수 수 제한

동적 조정

- 수요 변화에 대응

예측 조정

- 예측 수요에 따라 인스턴스 자동 예약

Elastic Load Balancing 사용한 트래픽 리디렉션

- 인스턴스 전체에 `워크로드를 균일하게 분산하는` 주요 관리형 서비스

1. 로드 밸런싱 부담 처리

- 요청을 받아 다음 처리할 인스턴스로 `라우팅` 하는 애플리케이션
- 인스턴스가 아닌 `region` 수준에서 실행되어 자동으로 고가용 서비스 제공

2. 자동 확장

- 트래픽 증가 시 시간당 비용 변경 없이 추가 처리량 처리

1. 플릿 확장 -> 트래픽 처리 준비 완료 후 종료
2. 플릿 축소 -> 모든 신규 트래픽 종료 후 인스턴스 종료

3. 내부 트래픽 통신

- 새로운 백엔드 인스턴스 추가 시 각 프론트엔드 인스턴스에 트래픽 수신 알림

- 대기 중 요청이 가장 적은 백엔드로 트래픽 송신

메시징 및 대기열

> 메시징
> 메시지를 완충 기억 장치에 배치
> 밀결합된 상태를 벗어나 프로세스 개선

> 페이로드
> 메시지에 포함된 데이터

Amazon SQS (Amazon Simple Queue Service)

- 규모에 상관 없이 소프트웨어 구성 요소 간에 메시지 전송, 저장, 수신
- 다중화 내재 -> 메시지 손실이나 서비스 중단 X
- 메시지가 처리되기 전까지 배치되는 영역
- 처리 후 대기열에서 삭제

모놀리식 애플리케이션

- DB, 서버, 사용자 인터페이스, 비즈니스 로직 등이 모두 직접 결합된 아키텍처

마이크로 서비스

- 애플리케이션 구성요소가 소결합
- 단일 구성에 장애가 발생해도 다른 요소들은 계속 작동 (전체 장애 X)

Amazon SNS (Amazon Simple Notification Service)

- 게시 / 구독 서비스
- 구독자 : 웹 서버, 이메일 주소, AWS 람다 함수 등

추가 컴퓨팅 서비스

서버리스

> 서버리스
> 애플리케이션을 호스팅하는 기본 인프라 / 인스턴스를 서버가 없는 것 처럼 관리 필요 X
> 코드가 서버에서 실행되지만 서버를 프로비저닝하거나 관리할 필요가 없다.
>
> - 유연성 : 애플리케이션 자동 확장

AWS Lambda

- 서버를 프로비저닝하거나 관리할 필요 없이 코드 실행 가능
- 트리거를 구성하고, 트리거 감지 시 함수를 `자동으로 실행`
- 15분 미만으로 실행 -> 딥러닝과 같은 장기 실행 프로세스에는 적합하지 않음
- 웹 서비스의 백엔드, 요청 처리, 이벤트 기반 애플리케이션에 적합

> **프로비저닝**

Lambda 동작 과정

1. 코드를 Lambda에 등록
2. AWS 서비스, 모바일 애플리케이션, HTTP 엔드포인트와 같은 이벤트 소스에서 `트리거` 되도록 설정
3. Lambda는 `트리거된 경우에만` 코드 실행
4. 사용한 컴퓨팅 시간에 대해서만 요금 지불

컨테이너 관리 시스템

- Docker 컨테이너 오케스트레이션 도구
- 두 모듈 모두 컨테이너 애플리케이션을 대규모 실행이 적합함
- 인스턴스를 직접 컨트롤할 필요가 없는 경우에는 `AWS Fargate`를 사용한다.

> **컨테이너**
>
> - 애플리케이션의 코드와 종속성을 `하나의 객체로 패키징`하는 표준 방식 제공
> - 보안성, 안정성, 확장성 요구 사항이 중요한 프로세스에 사용

> **컨테이너 오케스트레이션**
>
> - 컨테이너 관리를 지원하는 프로세스.
>
> EC2 인스턴스에서 실행, 가상 머신을 작동하는 방식과 비슷하게 격리되어 실행
> : 이 경우에 인스턴스가 호스트 -> 컨테이너와 같이 시작, 중지, 재시작, 모니터링 프로세스 필요
>
> 이러한 작업을 수행하는 프로세스를 컨테이너 오케스트레이션이라고 한다.

`ECS` (Amazon Elastic Container Service)

- 컨테이너식 애플리케이션을 실행, 확장 가능한 고성능 컨테이너 관리 시스템
- `Docker` 지원 -> API 호출을 사용하여 Docker 지원 애플리케이션을 시작 및 중지 가능

`EKS` (Amazon Elastic Kubernetis Service)

- Kubernetes를 실행하는데 사용하는 완전 관리형 서비스

AWS Fargate

- 컨테이너용 서버리스 컴퓨팅 플랫폼
- `인스턴스 관리까지` AWS에 맡길 수 있다.
- 서버 프로비저닝 / 관리 필요 없음

추가 자료

AWS에서의 컴퓨팅
AWS 컴퓨팅 블로그
AWS Compute Services
실습 자습서: 컴퓨팅
카테고리 심층 분석: 서버리스
AWS 고객 성공 사례: 서버리스