Chapter 01 클라우드 네이티브 소개

클라우드 네이티브 스프링 인 액션 스터디

이동준

1.2 클라우드와 클라우드 컴퓨팅 모델

클라우드 종류

- private cloud: 회사 내부에서 운영되는 클라우드
- public cloud: 외부 서비스 제공자가 일반에 제공하는 클라우드 (AWS, GCP, Azure)
- hybrid cloud: 두 가지 클라우드를 결합하여 사용하는 방식

클라우드 서비스 모델

- IaaS(Infrastructure as a Service) : AWS EC2
- CaaS(Container as a Service) : AWS EKS
- PaaS(Platform as a Service) : AWS Elastic beanstalk
- FaaS(Function as a Service) : AWS Lambda
- SaaS(Software as a Service) : Microsoft office 365

1.3 클라우드 네이티브 애플리케이션의 특성

클라우드 네이티브 애플리케이션의 특성

- 확장성
  - 수직적 확장(질적 향상)과 수평적 확장(양적 향상)
- 느슨한 결합(loosely coupled)
  - 모듈 내의 응집성을 낮추고 모듈 간의 결합성은 높인다
- 복원력
  - 결함 : 의도하지 않은 고장
  - 오류 : 예상 작동과 실제 작동 간의 불일치
  - 실패 : 특정 시스템이 비정상적으로 동작하는 현상
- 관찰 가능성
  - 모니터링 : Spring Boot Actuator, Prometheus
  - 시각화 : Grafana
  - 로깅 : Fluent bit, OpenSearch
- 관리 용이성
  - Spring Cloud Config Server, Kubernetes ConfigMap, Secret, Kustomize

1.4 클라우드 네이티브를 지원하는 문화 및 관행

클라우드 네이티브 개발을 위한 문화 및 관행

- 자동화
  - Terraform과 Ansible을 주로 활용한 코드형 인프라, 코드형 설정
- 지속적 전달
  - 지속적 통합 : Jenkins를 사용한 지속적 통합
  - 지속적 전달 : Github Action을 이용한 지속적 전달
  - 지속적 배포 : ArgoCD를 이용한 지속적 배포 시각화
- 데브옵스
  - Jenkins, Github Action, ArgoCD

1.5 클라우드의 목표

클라우드의 목표

- 속도 : 애플리케이션을 빠르게 배포하기 위해 클라우드가 필요하다
- 복원력 : 일부 애플리케이션이 실패하더라도 복원하기 쉬워야 한다
- 확장성 : 애플리케이션을 확장하기 쉬워야 한다
- 비용 : 재정적으로 값싸야 한다

1.6 클라우드 네이티브 구성

항목	가상화(Virtualization)	컨테이너화(Containerization)
가상화 방식	하드웨어 가상화 (하이퍼바이저 사용)	운영 체제 가상화 (커널 공유)
운영 체제	각 가상 머신이 독립적인 OS를 가짐	호스트 OS의 커널을 공유
시작 속도	느림 (완전한 OS 부팅 필요)	빠름 (애플리케이션 레벨 부팅)
자원 소비	무거움 (각 VM이 자체 OS 포함)	가벼움 (커널 공유, 애플리케이션만 포함)
격리 수준	매우 높음 (각 VM이 완전히 독립적)	중간 (커널을 공유하지만 프로세스는 격리)
확장성	확장에 다소 불리	매우 유리 (컨테이너 단위로 손쉽게 확장 가능)

들어주셔서 감사합니다