인프라 환경 공부 1~2장: 예진

Done	☐
Area	
Do Date	@2023년 6월 10일
Goal	
Projects	
URL	
When	

컨테이너 인프라 환경이란?

- 컨테이너를 중심으로 구성된 인프라 환경
- 컨테이너 : 하나의 운영 체제 커널에서 다른 프로세스에 영향을 받지 않고 독립적으로 실행되는 프로세스 상태

1. 개발자가 인프라를 이용해 개발하는 서비스의 설계 부분

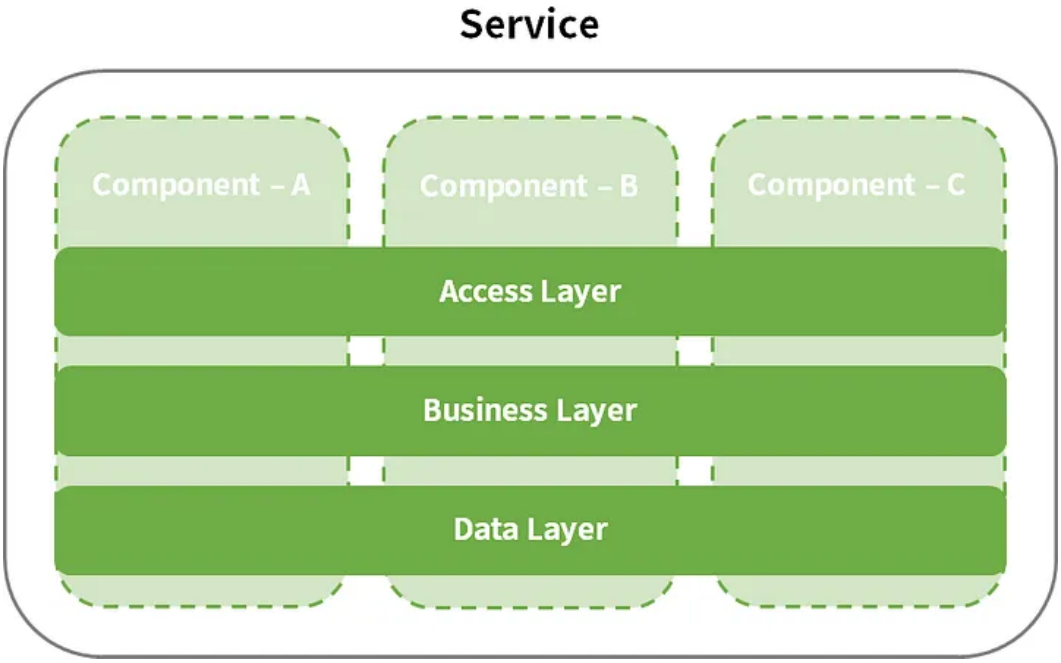
- 주어진 상황에 적합한 아키텍처를 사용하면 됨!

  ex)

  - 모놀리식 아키텍쳐로 시작 → 확장하며 마이크로 서비스 아키텍쳐로 전환
  - 중소기업의 소규모 프로젝트는 모놀리식 아키텍쳐를 선호하는 경향 있음
  - 컨테이너 인프라 환경은 특히 마이크로서비스 아키텍쳐로 구현하기에 적합!

모놀리식 아키텍쳐 (Monolithic Architecture)

- 하나의 큰 목적이 있는 서비스/애플리케이션에 여러 기능이 통합돼 있는 구조

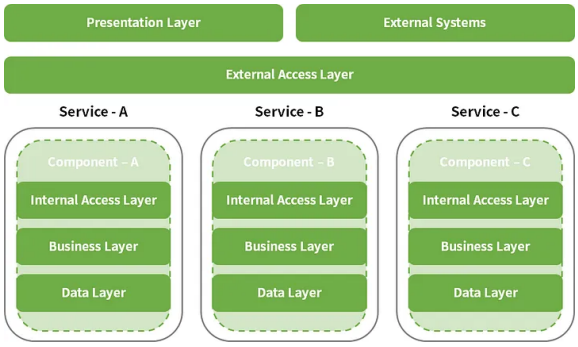


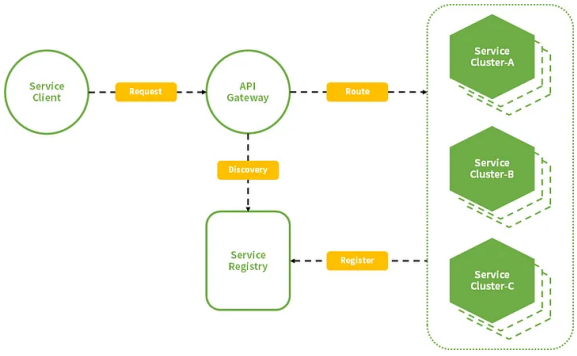
- 장점
  - 초기 단계에서 설계하기 용이
  - 개발이 단순하고 코드 관리 간편
- 단점

- 서비스를 운영하는 과정에서 수정이 많을 경우 → 어떤 서비스에서 이뤄진 수정이 연관된 다른 서비스에 영향을 미칠 가능성이 커짐
- 서비스가 점점 성장해 기능이 추가될수록 → 서비스 간의 관계가 매우 복잡해질 수 있음
- 향후 서비스가 변경됐을 때 다른 서비스에 영향을 미침

마이크로 서비스 아키텍쳐 (Microservices Architecture; MSA)

- 시스템 전체가 하나의 목적을 지향하며, 개별 기능을 하는 작은 서비스를 각각 개발해 연결함.
- 보안, 인증 등과 관련된 기능이 독립된 서비스를 구성하고 있으며, 다른 서비스들도 독립적으로 동작할 수 있는 완결된 구조





- 각 서비스는 **API 게이트웨이와 REST API**를 이용한 통신방식으로 사용자(외부)의 요청을 전달함
- 어떤 서비스가 등록돼 있는지 파악하기 위해 **서비스 디스커버리**를 사용함
- 수많은 서비스의 내부 통신을 이벤트로 일원화하고 이를 효과적으로 관리하기 위해 별도로 **이벤트 버스**를 서비스로 구성함

- 장점
  - 개발된 서비스를 재사용하기 쉬움

    - 향후 서비스가 변경됐을 때 다른 서비스에 영향을 미칠 가능성이 줄어들음
    - 사용량의 변화에 따라 특정 서비스만 확장할 수 있음
    - 사용자의 요구 사항에 따라 가용성을 즉각적으로 확보해야 하는 IaaS 환경에 적합함
- 단점
    - 모놀리식 아키텍처보다 복잡도가 높음
    - 각 서비스가 서로 유기적으로 통신하는 구조로 설계되기 때문에 네트워크를 통한 호출 횟수가 증가해 성능에 영향을 줄 수 있음

---

2. 컨테이너 인프라 환경을 지원하는 도구

컨테이너 인프라 환경 구성

⇒ `컨테이너`, `컨테이너 관리`, `개발 환경 구성 및 배포 자동화`, `모니터링`

컨테이너	컨테이너 관리	개발 환경 구성 및 배포 자동화	모니터링
도커	**쿠버네티스**	**젠킨스**	**프로메테우스 + 그라파나**

(컨테이너) 도커 Docker

- 컨테이너 환경에서 독립적으로 애플리케이션을 실행할 수 있도록 컨테이너를 만들고 관리하는 것을 도와주는 컨테이너 도구
- **운영 체제 환경에 관계없이 독립적인 환경에서 일관된 결과를 보장**함

> - 그 외 컨테이너 도구

> - 컨테이너디(Containerd), 크라이오(CRI-O), 파드맨(Podman) 등

(컨테이너 관리) 쿠버네티스 Kubernetes

- 다수의 컨테이너를 관리하는 데 사용
- 컨테이너 인프라에서 필요한 기능을 통합 및 관리하는 솔루션
- 컨테이너의 자동 배포와 배포된 컨테이너에 대한 동작 보증, 부하에 따른 동적 확장 등의 기능 제공

> -
> - 그 외 컨테이너 관리 도구
>   - 도커 스웜(Docker Swarm), 메소스(Mesos), 노마드(Nomad) 등

(개발 환경 구성 및 배포 자동화) 젠킨스 Jenkins

- **지속적 통합(CI**, Continuous Integration), **지속적 배포(CD**, Continuous Deployment) 지원
- 개발한 프로그램의 빌드, 테스트, 패키지화, 배포 단계를 모두 자동화해 개발 단계를 표준화함
- 개발된 코드의 빠른 적용과 효과적인 관리를 통해 개발 생산성을 높이는 데 초점

> -
> - 그 외 지속적 통합과 배포를 위한 도구

> - 뱀부(Bamboo), 깃허브 액션(Github Action), 팀시티(Teamcity) 등

(모니터링) 프로메테우스와 그라파나 Prometheus & Grafana

- 컨테이너 인프라 환경에서는 많은 종류의 소규모 기능이 각각 나누어 개발되기 때문에 중앙 모니터링이 필요함

  → 효율적으로 모니터링하는 방법

  ⇒ 프로메테우스 (상태 데이터를 수집) + 그라파나 (수집한 데이터를 관리자가 보기 좋게 시각화)
- 컨테이너로 패키징돼 동작하며, 최소한의 자원으로 쿠버네티스 클러스터의 상태를 시각적으로 표현함

> -
>
> - 그 외 컨테이너 관리 도구
>   - 데이터독(DataDog), 인플럭스DB(InfluxDB), 뉴 렐릭(New Relic) 등

---

3. 컨테이너 인프라 환경 구성

❤ 그림 1-8 실습용 컨테이너 인프라 환경 구성