CLOUD IT

Report

오픈소스를 활용한 3-Tier 환경 구축 #3주차

- 3 Tier 구축 (1) -

작성자	코더 – 정지호, 최예진, 김재현
검수자	송인섭 이사, 윤상훈 수석
작성일	2022-10-02

목차

개요

오픈소스 사전 조사

- WEB
- WAS
- DB

WEB, WAS, DB 표준 구성

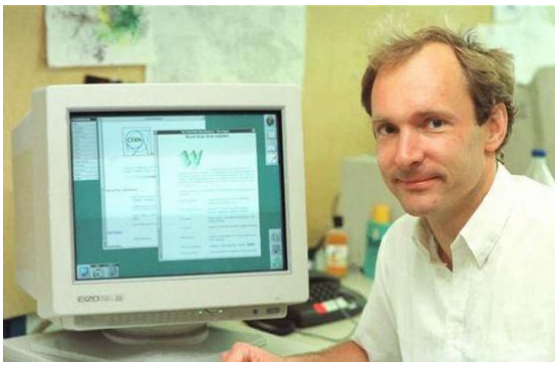
WEB 개요

CLOIT

- 『월드 와이드 웹. World Wide Web』의 준말이다.
  인터넷에 연결된 사용자들이 서로의 정보를 공유할 수 있는 공간을 의미한다.
  WWW 나 W3 라고 부르기도 하지만, Web이라는 말로 가장 많이 부른다.
- 인터넷과 같은 의미로 많이 사용되지만, 엄밀히 말하면 웹은 인터넷 상의 인기 서비스 중 하나. 인터넷과는 다른 개념이다.
  하지만 두 용어가 혼용되어 사용될 정도로 인터넷에 큰 부분을 차지하고 있는 것이 사실이다.
- 웹 이전의 인터넷은 초기의 명령어 기반 구조를 가지고 있었으며, 하드웨어와 OS에 따라 다른 명령어를 사용해야 했지만,
  인터넷 상에서의 정보저장과 사용환경에 대한 연구를 통해 웹에서는 컴퓨터의 종류를 불문하고 한 가지 종류의 표준 사용자 환경으로 조작이 가능하도록 했다.

웹의 역사

CLOIT

- 1969년 인터넷이 개발되었다.
- 웹은 1989년 3월 유럽 입자 물리 연구소의 컴퓨터과학자 "팀 버너스리"에 의해 연구와 개발이 시작되었다.
-  팀 버너스리
- E-mail이나 FTP를 통해 주고받는 것을 비효율적으로 생각하여 제안한 공통된 공간의 정보를 공유하고 관리할 수 있게 하자는 아이디어에서 출발했다.
  세계 여러 대학과 연구기관에서 일하는 학자 상호간의 신속한 정보교환과 공통 연구를 위한 프로그램으로 고안되었다. 문자, 사진, 동영상 등이 조합된 데이터베이스인 사이트의 정보를 전용 열람용 소프트웨어인 웹 브라우저를 통해 입수하고, 입수한 정보를 간단한 방식으로 전송하는 것도 가능하다.

- 초창기의 웹은 지금과는 사뭇 다른 양상을 보였으며, 단순한 정보저장의 역할만 수행했다고 한다.
  세계적으로 대중화된 최초의 GUI 기반 웹 브라우저는 일리노이 대학의 "마크 안데르센"이 작성했다고 한다. "모자이크 웹 브라우저"

  사용자들은 인터넷 '브라우저'의 도움으로 웹 세계를 서핑할 수 있고, 이를 통해 엄청나게 다양한 영역의 자료, 프로그램 등을 접하고 다운받을 수 있다.
- WWW에 관련된 기술은 `월드 와이드 웹 컨소지엄(W3C)`이 개발하고 있다.
  HTML, HTTP 등의 표준화를 진행하고 있으며, 최근에는 시맨틱 웹에 관련된 표준을 제정하고 있다고 한다.

웹의 구성

CLOIT

웹 페이지(web page)

- 웹 브라우저에서 보여지는 문서.
- HTML로 작성되거나 이미지 파일 등을 포함한 하이퍼 텍스트 문서

웹 사이트(web site)

- 인터넷 프로토콜 기반의 네트워크에서 도메인 이름이나 IP 주소, 루트 경로만으로 이루어진 일반 URL을 통해 보이는 웹 페이지들의 의미 있는 묶음.
- 웹 페이지 중 서로 관련된 내용으로 작성된 페이지들의 집합으로 웹 페이지들이 하이퍼링크를 통해 서로 연결되어 구성된다.
  즉 웹 페이지의 집합을 웹 사이트라고 할 수 있다.

- 웹 사이트와 홈페이지라는 용어를 같은 의미로 사용하는 경우가 많지만, 조금 다르다.
  홈페이지의 정확한 의미는 웹 브라우저가 시작될 때 맨 처음 자동으로 표시되는 웹 페이지나 웹 브라우저의 홈 버튼을 누를 때 이동되는 웹 페이지를 지칭하는 것이다.

- 최초의 웹사이트는 팀 버너스리가 만든 `info.cern.ch`
- 웹 사이트는 인터넷이나 랜과 같은 네트워크를 통해 접속할 수 있는, 적어도 하나의 웹 서버 상에서 호스팅된다. 웹 페이지는 HTML의 형식으로 표현되지만, 일반적으로 순수 문자열로 쓰여진 문서이며, HTTP를 통해 접속된다. (HTTPS를 통한 암호화를 사용하여 웹 페이지 콘텐츠를 이용한 사람들에게 보안과 개인 정보 보호를 제공하기도 한다.)
- 공식적으로 접속할 수 있는 모든 웹사이트는 총체적으로 WWW를 이루고 있다.

웹 브라우저(web browser)

- 인터넷 브라우저, 웹 탐색기 등.
- 웹 서버에서 이동하며 쌍방향으로 통신하고 HTML 문서나 파일을 출력하는 그래픽 사용자 인터페이스 기반의 소프트웨어.
- 대표적인 HTTP 사용자 에이전트의 하나.
- 구글 크롬, 인터넷 익스플로러/마이크로 소프트 엣지, 사파리 등이 이에 해당한다.

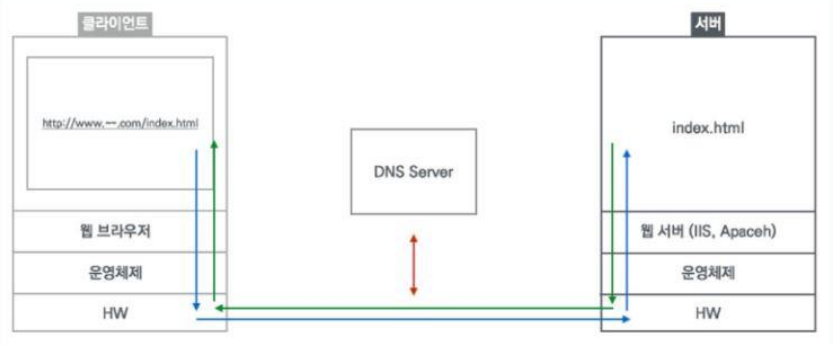
웹 클라이언트 (Web Client)

- 웹을 사용하는 고객.
- 디바이스 장치에서 동작하는 웹 클라이언트 프로그램을 '웹 브라우저'라고 한다.
- 웹 클라이언트는 웹 브라우저에서 발생하는 특정 행위를 웹 서버에 요청(Request)한다.
- 사용자가 실제로 보는 view-page이며, 역할 차원에서는 프론트 엔드.

웹 서버 (Web Server)

- 클라이언트가 송신한 요청을 처리하고 필요한 데이터들을 응답(Response) 한다.
- 웹 브라우저는 웹 서버에서 응답 받은 데이터(이미지, 텍스트, 파일, 동영상
- 등)를 화면에 표시하며, 사용자가 생성한 웹에 대한 요청을 받아들이고 이에 대한 로직 및 데이터베이스와 연동을 하는 곳으로, 역할 차원으로는 백엔드.
- 클라이언트에게 콘텐츠를 전달받는 역할도 함. 파일 업로드를 포함하여 클라이언트에서 제출한 웹 폼을 수신하기 위해 사용됨.
- 대다수의 웹 서버는 ASP, PHP등 `서버 사이드 스크립트 언어`를 지원한다.
- 서버 소프트웨어의 변경 없이도 웹 서버가 동작을 수행한다는 것은 분리된 서버 사이드 스크립트 언어에 기술할 수 있다는 의미이다. 이를 통해 구현되는 기능이란 HTML 문서를 동적으로 생성하는 것을 의미하며, 이렇게 생성된 문서를 동적 컨텐츠라 하는데, DB의 정보를 조회해서 보여주거나 수정하기 위해 사용된다.
- 이와 대비되는 개념으로 정적 컨텐츠가 있는데 동적 컨텐츠보다 더 빠르게 동작하고 쉽게 캐시될 수 있지만, 반환되는 컨텐츠의 내용이 동일하다는 점이 다르다.

(클라이언트와 서버의 요청 응답)



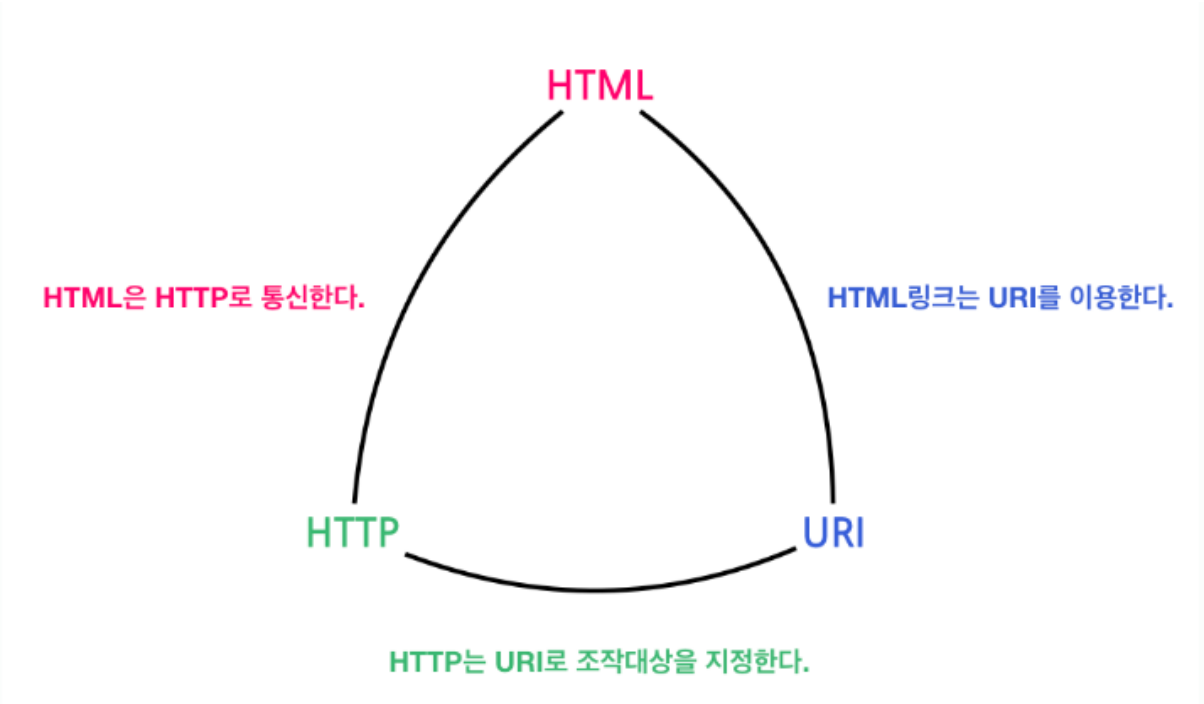
클라이언트-서버의 관계

웹의 기술

CLOIT

프로토콜과 표준

- 웹 페이지를 가져오거나 정보를 송신하기 위해(웹 문서를 열기 위해) 대부분의 웹 서버가 사용하는 HTTP(hyper-text transfer protocol)로 통신한다.
- 페이지들은 주소처럼 이용되는 URL(uniform resource locator)을 통해 장소가 정해지고, HTTP 접근을 위해 `http:`로 시작된다
- FTP를 위한 `ftp:` 암호화 HTTP인 HTTPS를 위한 `https:`와 같은 다양한 URL 종류와 대응 프로토콜을 지원한다.
- 웹 페이지의 파일은 보통 HTML(hyper-text markup language)으로 작성된다.
- 대부분의 브라우저는 HTML외에도 JPEG, PNG, GIF 등 이미지 포맷을 지원하고, 그 밖에도 플러그인을 통해 확장할 수 있다.



HTTP (Hyper Text Transfer Protocol)

- 애플리케이션 컨트롤
- 웹 상에서 정보를 주고받을 수 있는 프로토콜. 주로 HTML 문서를 주고받는 데 쓰인다.
- 주로 TCP를 사용하고 HTTP/3부터는 UDP를 사용하며, 80번 포트를 사용한다.
- 클라이언트-서버 사이에 이루어지는 요청/응답 프로토콜.
- 클라이언트-서버 사이의 소통은 평문(ASCII) 메시지로 이루어진다.
  클라이언트는 서버로 요청메시지를 전달하며, 서버는 응답 메시지를 보낸다.
- 하이퍼텍스트(hypertext) : 문서 내부에 또 다른 문서로 연결되는 참조를 집어 넣음으로써 웹 상에 존재하는 여러 문서끼리 서로 참조할 수 있는 기술.
- 하이퍼링크(hyperlink) : 문서 내부에서 또 다른 문서로 연결되는 참조.

HTML (Hyper Text Markup Language)

- 하이퍼미디어 포맷
- 웹 페이지 표시를 위해 개발된 지배적인 마크업 언어.
- 제목, 단락 등과 같은 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공하며, 이미지와 객체를 내장, 대화형 양식을 생성하는데 사용할 수도 있다.
- 웹에서는 HTML이라는 언어를 사용해 누구나 자신만의 문서를 작성할 수 있으며, 이렇게 작성된 웹상 문서는 HTTP라는 프로토콜을 사용하면 누구나 검색하고 접근할 수 있다.

URI (Uniform Resource Identifier)

- 리소스 식별자
- 인터넷에 있는 자원을 나타내는 유일한 주소.
  인터넷에서 요구되는 기본조건으로서 인터넷 프로토콜에 항상 붙어 다닌다.
- URL (Uniform Resource Locator)
  네트워크 상에서 자원이 어디 있는지 알려주기 위한 규약.
  네트워크와 검색 메커니즘에서의 위치를 지정하는, 웹 리소스에 대한 참조.
  웹 사이트 주소 뿐만 아니라 컴퓨터 네트워크 상의 자원을 모두 나타낼 수 있음.
  주소에 접속하려면 해당 URL에 맞는 프로토콜을 알아야 하고, 그와 동일한 프로토콜로 접속해야 한다.
- URN (Uniform Resource Name)
  `urn:scheme`를 사용하는 URI를 위한 이름.
  영속적이고, 위치에 독립적인 자원을 위한 지시자로 사용하기 위해 정의됨.

용어

- DNS (Domain Name System)
  사람이 읽을 수 있는 도메인 이름을 머신이 읽을 수 있는 IP주소로 변환함
- IP
  인터넷으로 통신하는 각 디바이스에 부여된 고유의 값.
  인터넷 상의 모든 컴퓨터는 숫자를 사용하여 서로를 찾고 통신하며 이때 필요한 것이 IP주소
- Domain
  문자로 된 고유주소. 수많은 IP를 사람이 외워서 접속할 수는 없기 때문에 기억하기 쉽게 만들어져 있다.

오픈소스 웹서버

CLOIT

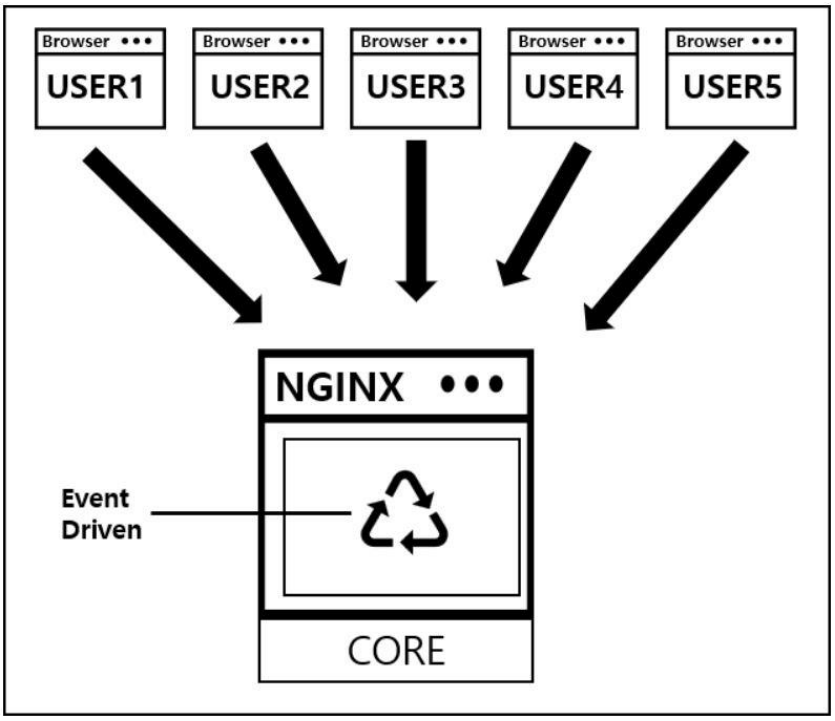
1. 엔진엑스 HTTPd 서버

- NGINX는 웹사이트 트래픽 증가와 광대역 인터넷의 성장, 이에 따른 동시 접속을 관리해야 할 필요에 대응하고자 2002년 러시아 엔지니어인 `이고르 시오세프`가 개발한 경량 웹 서버이다.
- HTTP와 HTTP/2를 지원하며, HTTP/3는 개발중.
- `클라우드플레어`에서는 자체적으로 개발한 NGINX 확장 모듈을 이용해 HTTP/3 상용 서비스를 하고 있다고 한다.
- HTTP Web Server로 활용될 수도, Reverse Proxy Server로 활용하여 WAS 서버의 부하를 줄일 수 있는 로드 밸런서로 활용할 수도 있다.

특징

1. Event-Driven(비동기) 방식

- 비동기 방식으로 먼저 처리해야 되는 것부터 진행한다.
- Single-Thread로 동작하며, non-blocking I/O 이벤트 기반으로 요청을 처리하기 때문에 적은 자원으로 효율적인 트래픽 처리가 가능하다.
- 새 요청이 들어오더라도 새로운 프로세스와 쓰레드를 생성하지 않기 때문에 프로세스와 쓰레드 생성 비용이 존재하지 않고, 적은 자원으로 효율적인 운용이 가능하다.
- 단일 서버에서도 동시에 많은 연결을 처리할 수 있다.



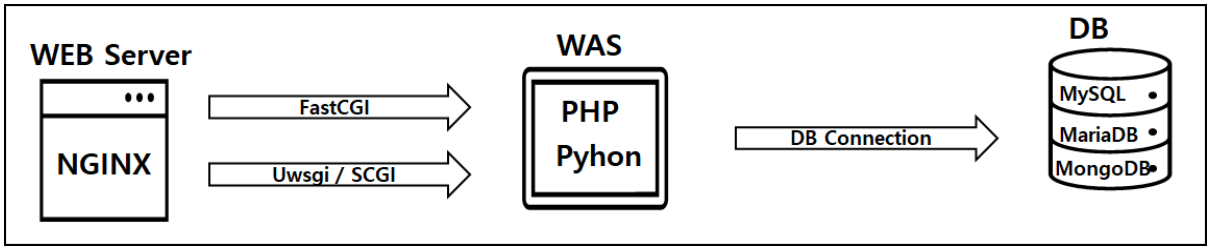
-

2. SSL 지원 및 SNI 기능 지원

- 예전 웹 서버 같은 경우 1개의 IP에는 1개의 도메인의 SSL 인증서만 적용해야 됐으며, 1개의 IP에 여러 개의 도메인 인증서로 적용하려면 몇 배 더 비싼 멀티 인증서로 적용해야 했다.
- 이런 부분을 해결하기 위한 기능이 SNI이며, Nginx에서는 SNI 기능을 지원한다.

3. FastCGI / SCGI 지원

- FastCGI /SCGI 를 기능을 통해 웹서버와 WAS를 분리하여 서버의 부하를 방지하고 물리적으로 분리하여 보안을 강화 할 수 있다.
- FastCGI : 기본 웹 서버 모듈의 일부 성능 특성을 CGI(Common Gateway Iterface) 프로그래밍 인터페이스의 웹 서버 독립성과 결합하는 웹 서버와 애플리케이션 간의 인터페이스.
  언어에 독립적이고, 확장가능한 아키텍쳐인 CGI에 대한 개방형 확장.
- SCGI(Simple Common Gateway Interface)
  좀 더 간편한 FastCGI.



4. Proxy 서버 지원 및 로드밸런싱 지원

- 프록시 서버 기능 지원으로 보안적으로 강화 가능.
  프록시 서버 구성으로 로드밸런싱 구성도 가능하다.
- 다음의 로드밸런싱 방법을 지원한다.
  Round Robbin : 주어진 서버들을 순차적으로 돌아가며 선택.
  Least-connected : 현재 접속 수가 가장 적은 서버를 선택.
  IP-hash : hash function을 사용, 클라이언트 ip를 hash 한 결과에 따라 다른 서버를 할당.
- 리버스 프록시(reverse proxy)로도 활용할 수 있다.
- 사용할 때 환경 설정을 바꾸고 나서 서버 데몬을 재시작할 필요 없이 reload 시그널을 보내면 된다. 즉 프로세스를 재시작할 필요가 없다는 장점이 있다.

2. 아파치 HTTP 서버

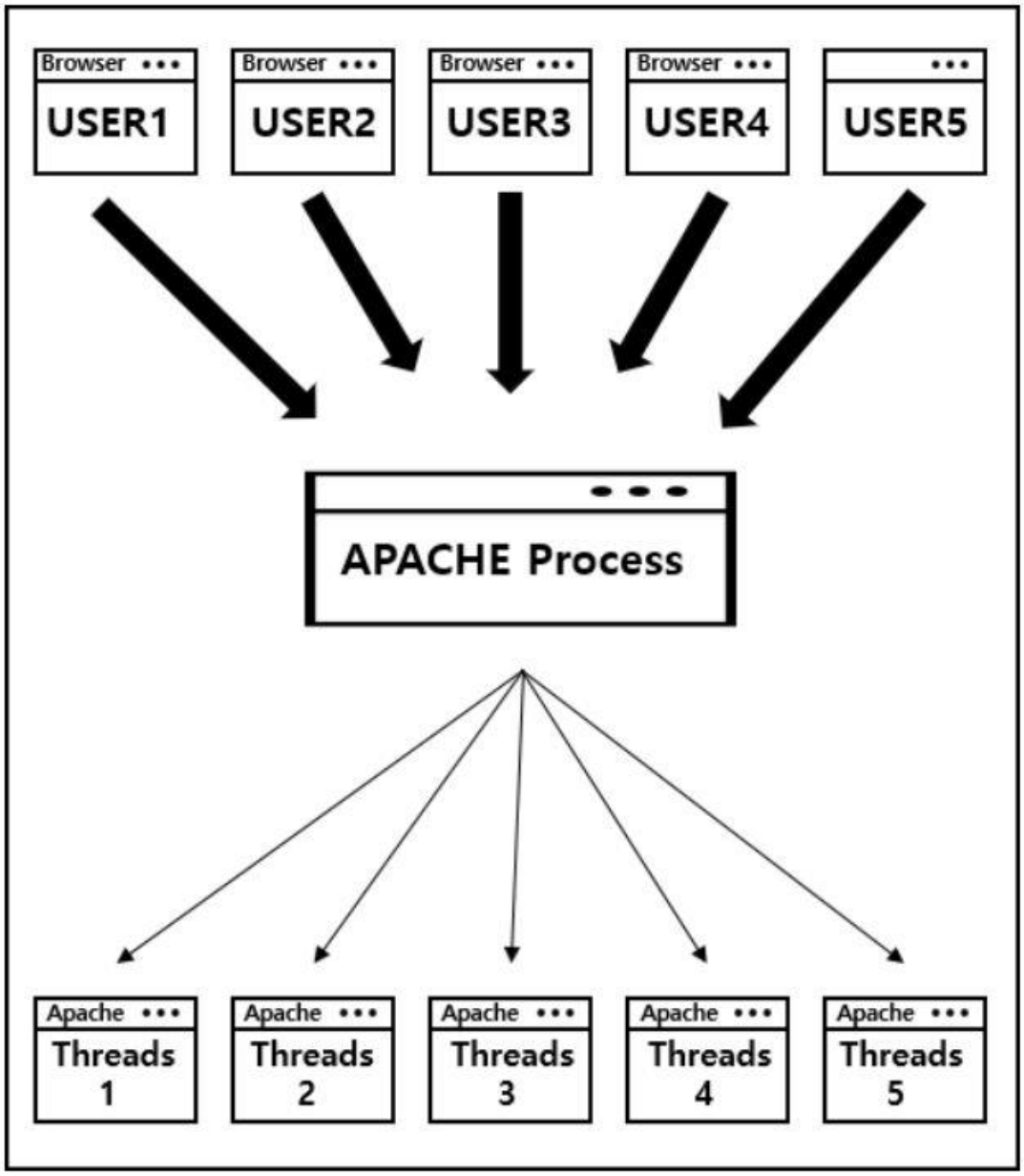
- 오픈 소스 소프트웨어 그룹인 `아파치 소프트웨어 재단`에서 만든 웹 서버 프로그램. 최초의 웹 서버 프로그램인 `NCSA HTTPd`를 기반으로 만들어졌다.
- `NCSA HTTPd`가 유닉스 기반으로 만들어졌기에, 아파치 HTTP 서버는 `NCSA HTTPd`를 리눅스에서도 구동하는 것을 목표로 만들어진 프로그램이다. 이후 리눅스와 함께 퍼져나갔고, 리눅스가 서버 OS를 최다 점유하자, 아파치도 자연스럽게 최다 점유율을 차지하게 되었다.

특징

- 확장성이 상당히 좋다. 모듈이라는 개념으로 수많은 기능을 덧붙일 수 있으며, 이 모듈을 통해 다른 프로그램과의 연동도 가능하다. 이 때문에 여러가지 서버사이드 프로그래밍 언어나 DBMS와도 궁합이 잘 맞았다.
  특히 리눅스 서버 운영체제, PHP와 MySQL과 함께 사용하는 것을 `LAPM`이라고 통칭되면서 웹 서버의 기본처럼 여겨졌다.
- 아파치 자체는 웹 서버로의 아주 기본적인 기능만 수행하고, PHP를 비롯해 대부분 발전된 기능은 외부 모듈로 구현한다. 오랜 역사와 높은 시장 점유율 덕분에 이런 외부 모듈이 많고, 안정화도 잘 되어있기에 아파치가 시장 점유율이 높은 것.
  대규모 엔터프라이즈, 거대 규모 웹서버 시장 점유율이 높으며, 금융권을 비롯한 안정성과 보안이 중요한 곳에서도 아파치를 사용한다.

- 라이선스는 GPL이 아닌 자체 라이선스를 사용한다. 아파치 라이선스 2.0을 따르는데, 아파치 소프트웨어 재단에서 만들었다는 사실을 밝히고, 아파치 라이선스를 따르면 자유롭게 수정 및 배포가 가능하다. 소스 공개 강제 사항도 없다.

- 요청 당 프로세스 또는 쓰레드가 처리하는 구조이며, 요청이 많아지면 CPU와 메모리 사용량이 높아져 성능이 저하될 수 있다.
- 이 문제들은 킵얼라이브(Keep Alive)를 활성화함으로 해결할 수 있지만, 대량 접속 시 효율이 급격히 떨어지는 문제점이 발생한다.



- Apache HTTPd에서 사용하는 요청 당 스레드 혹은 프로세스 기반의 구조.
- 높은 안정성, 다채로운 기능들, 다수의 서버 측면 프로그래밍 언어를 지원하는 것으로 유명하다.

3. 라이티(혹은 라이티피디)

- Lighttpd : `Light`와 `httpd`를 합친 말이지만, 웹 서버가 가진 속도, 유연성, 안정성을 표현하기 위해 '라이티'라고 발음한다.
- 고성능의 속도가 핵심인 환경에 맞춰 최적화된 경량 웹 서버이며, 고부하 서버들에게 이상적이다.
- `잰 네쉬체`가 개발했으며, 2003년 대학 논문을 쓰면서 개념 증명 설계로서 시작한 것이 현재 가장 인기 있는 웹 서버의 하나가 되었다.

- 메모리 풋프린트(memory footprint)가 비교적 낮고, CPU 부하가 적으며, 여러 고급 기능을 갖는다.
- 적은 자원으로 높은 성능을 내는 웹서버로서 고안되었으며, 적은 메모리를 사용하면서도 높은 속도를 낸다.
- 뛰어난 무결성 수준으로 외부 프로그램 인터페이스를 지원하고, 어떤 프로그래밍 언어로 작성된 웹 애플리케이션이든 전부 지원한다.

기능

- 외부 프로그램에 대한 FastCGI, SCGI, CGI 인터페이스를 지원하므로 모든 프로그래밍 언어로 작성된 웹 응용 프로그램을 서버에서 사용할 수 있다.
  특히 PHP를 적절하고 효율적으로 지원하도록 구성할 수 있다.
- SNI 지원이 있는 SSL/TLS
- 유연한 가상 호스팅과 모듈 지원.

히와이타

- Hiawatha는 '휴고 리싱크'가 2002년 네덜란드에서 컴퓨터 공학을 공부하던 중 기숙사에서 인터넷 서버들을 지원하고 싶어서 개발하였다.
- 목표는 서버들에서 발견된 보안 한계를 둘러싼 취약점들과 혼란스러운 구성 툴에 대처하는 시스템을 개발하는 것.
- 이 서버는 다른 웹 서버들이 가진 모든 일반적 보안 수단은 물론 독자적인 보안 기능을 여럿 포함한다. 가독성 있는 구성 구문으로 HTTP나 CGI에 대한 전문성 없이 사용할 수 있다.
- 크기가 작고, 보안이 안정적이며, 쉽게 설치할 수 있다는 강점.
- 고급 기능보다는 보안, 이용성, 속도, 성능을 우선시하는 아파치의 경량 대안을 찾는 사람들에게 이상적이다.

기능

- CGI 및 로드 밸런싱 FastCGI 지원
- 대용량 파일 지원
- 리버스 프록시 기능
- URL 재작성을 지원하는 URL 툴킷
- SSL 및 TLS 지원

체로키

- Cherokee는 2001년 아카마이 테크놀리지의 엔지니어링 총괄 `알바로 로페즈 오르테가`가 개발하였다. 오르테가는 모듈식 경량 설계 안에 속도와 기능성을 취합하고 싶어 했다.
- 체로키는 메모리 풋프린트가 낮고, 부하 밸런싱 장치들이 있고, 확장성이 있으며, 고성능에 이용자 친화적인 웹 서버로서 이내 탁월성을 인정받았다.
- 인상적인 여러 기능, 예컨대 서버 및 이의 모든 기능의 직접적 구성 설정을 지원하는 `체로키-어드민`이라는 웹-기반 관리 인터페이스를 포함한다.

기능

- Linux , BSD 변형 , Solaris , OS X 및 Windows 에서 실행 되는 오픈 소스 크로스 플랫폼 웹 서버.
- `cherokee-admin`이라는 그래픽 관리 인터페이스와 모듈식 경량 설계 지원.
- TLS/SSL
- URL 재작성 및 리디렉션
- 리버스 HTTP 프록시
- FastCGI 및 SCGI

몽키 HTTP 서버

- Monkey HTTP는 처음에는 리눅스용으로 최적화되었으나 현재에는 맥에서도 구동되는 경량 서버 및 개발 스택이다. 임베디드 디바이스를 위해 개발되었고, 결과적으로 확장성이 뛰어나고, 아울러 메모리와 CPU 소비가 낮다.
- 2002년 단순히 학습을 위한 실험 용도로 시작되었지만, 서버가 이벤트 중심 시스템으로 재개발되었던 2008년부터 상업적 용도로 성격이 변화했다.
- 서버는 스레드 별로 수천 클라이언트를 수용하는 하이브리드 메커니즘을 통해 기능. 설치가 쉽고 임베디드 장치에 이상적인, 초소형 서버의 과부하 하에서 뛰어난 성능을 제공한다.
- 특정 시스템 호출 및 최적화 기술 측면에서 Linux 커널을 최대한 활용하여 고부하에서 고성능을 달성하도록 설계되었다.

기능

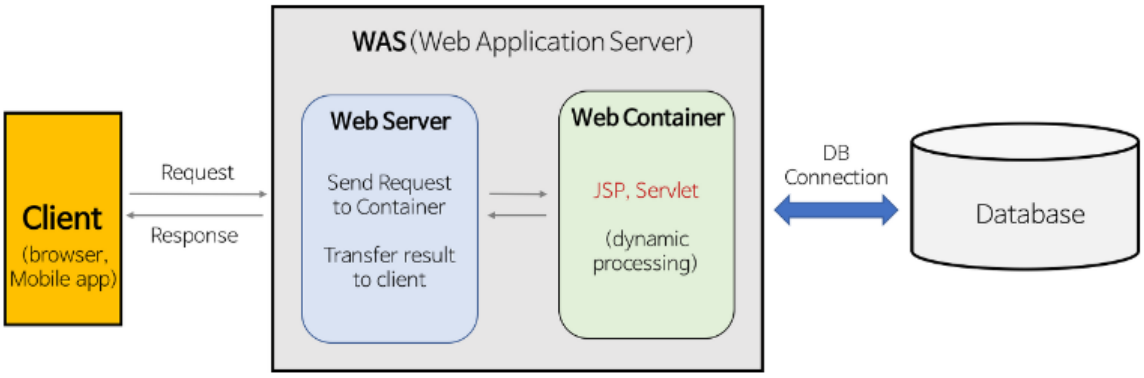
- IPv6 , TLS , 가상 호스트, CGI , FastCGI , 디렉토리 목록 및 보안 규칙과 같은 공통 기능을 지원한다.

WAS에 대한 오픈 소스

CLOIT

1. WAS

- 웹 애플리케이션 서버(Web Application Server)
- 기본 기능으로 다음 세 가지를 꼽을 수 있다.
  1) 프로그램 실행 환경과 DB 접속 기능을 제공
  2) 여러 개의 트랜잭션을 관리
  (트랜잭션: DB 에 접근하기 위해 수행하는 작업의 단위)
  3) 비즈니스 로직을 수행
  (비즈니스 로직: 공학/기술적인 문제를 해결하는 코드)
- 즉, 동적인 처리가 필요한 **동적인 요청(매번 같지 않은)을 처리**하기 위한 것이라고 생각하면 된다.
- **Web Server 와 WAS**



1. **Web Server**
   - 개념이 하드웨어적 관점과 소프트웨어적 관점으로 구분된다.
   - 하드웨어적 관점에서는 **Web Server 가 설치되어 있는 컴퓨터**를 의미한다.
   - 소프트웨어적 관점에서는 클라이언트로부터 HTTP 요청(클라이언트가 서버로 보내는 데이터 패킷)을 받아 **정적인 컨텐츠(.html, jpeg 등)를 제공하는 컴퓨터 프로그램**을 의미한다.

- 기능: 클라이언트에게 **정적인 요청을 받으면 정적인 컨텐츠를 전달**한다. 클라이언트에게 **동적인 요청을 받으면 WAS 로 처리를 이관**한 뒤, **WAS 에서 처리한 결과를 전달**한다.
- 대표적인 웹 서버로는 **Apache**, WebtoB, **Nginx**, IIS 가 있다.

2. **Web Server 와 WAS 를 분리해놓은 이유**
   - WAS 는 그럼 정적인 컨텐츠를 처리하지 못하는 것인가? -> **아니다!**
   - WAS 는 동적인 컨텐츠와 함께 **정적인 컨텐츠도 처리할 수 있도록 설계**되었다. 그래서 **WAS 만 사용해서 웹 서비스를 제공하는 것도 가능**하다. 사실, **트래픽(흐르는 데이터의 양)이 적다면 WAS 하나만 사용하여 서비스하는 것이 복잡하지도 않고, 유지 및 보수도 편하다.**
   - 그렇다면 왜 Web Server 와 WAS 를 분리하는 것일까?
     -> **WAS 는 트래픽 처리량에 있어 약점**이 있기 때문이다.
     -> 그렇기에 트래픽이 점차 많아지는 경우, **정적인 요청은 Web Server 가 처리**하도록 하여 트래픽을 분산시키는 것이다.

   - **Web Server 와 WAS 를 분리할 시 생기는 장점**

     1) **서버 부하 방지**

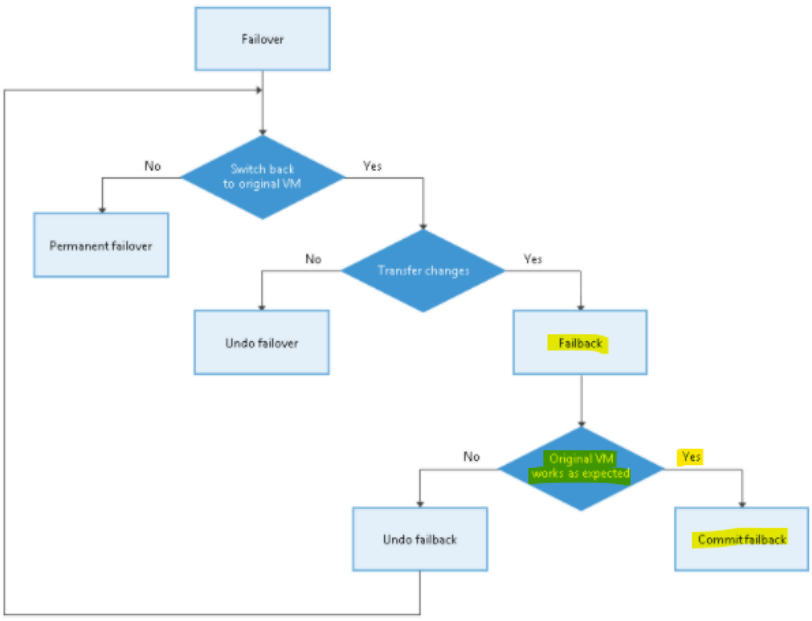
     - Web Server 를 두는 **가장 큰 이유**이다. 정적인 요청은 Web Server 가 처리하도록 하여 WAS 는 동적인 요청만 처리할 수 있도록 하면 서버 부하를 방지할 수 있다.

     2) **보안 강화**

     - Web Server 와 WAS 는 **포트 번호가 서로 다르다**(Web Server 는 표준 HTTP 포트가 80, WAS 는 표준 HTTP 포트가 8080). 그렇기에 이 두 개를 분리하여 사용하면 보안을 강화시켜 줄 수 있다.

     3) **여러 대의 WAS 를 연결 가능**

- Web Server 와 WAS 를 분리하여 사용하면, **Web Server 하나에 여러대의 WAS 를 설치하고 로드밸런싱을 하여 각 WAS 의 부하를 더 낮춰줄 수 있다(로드밸런싱, 말 그대로 Load, 즉 요청을 여러 개로 나누어 처리하는 것)**. 또한 여러 대의 WAS 연결을 위해 Web Server 를 사용하면 **fail over, fail back 처리에 더 유리**해진다.
- **fail over**
  : 장애 조치 기능. 서버나 네트워크 등에서 이상이 생겼을 때 **예비시스템으로 자동 전환되는 기능**이다.
  : 평상시에 A 장비를 사용(active)하다 A 장비에 장애(fail)가 발생하면 준비했던(passive) B 장비를 사용하는 것이다.
- **fail back**
  : fail over 에 따라 전환된 서버, 시스템, 네트워크 등을 **이상이 발생하기 전의 상태로 되돌리는 처리**



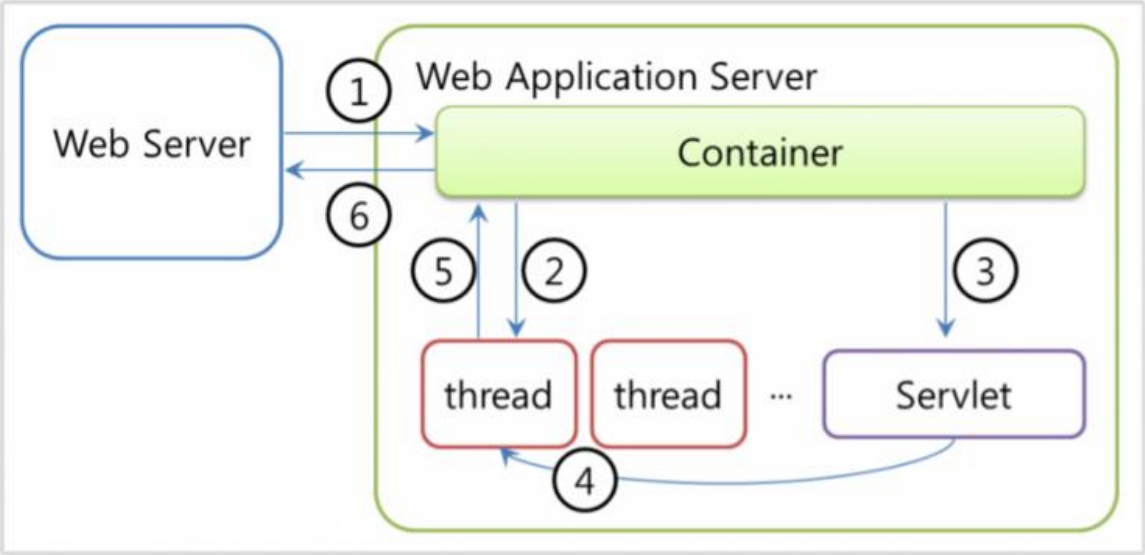
- 쉽게 생각하면, **장비를 여러 대 둘 수 있게 하여 장애를 대비**하는 것이다.

4) **여러 웹 어플리케이션 서비스 가능**

- 예를 들어, 하나의 서버에서 PHP Application 과 Java Application 을 함께 사용하는 경우, **자원 이용의 효율성 및 장애 극복, 배포 및 유지보수의 편의성을 위해 Web Server 와 WAS 를 분리**한다.

5) **접근 허용 IP 관리도 Web Server 에서 처리하면 효율적이다.**

- **WAS 작동 프로세스**



1) 웹 서버로 부터 요청이 들어오면 제일 먼저 컨테이너가 이를 알맞게 처리한다.

2) 컨테이너는 배포 서술자(web.xml)를 참조하여 해당 서블릿(요청에 대한 결과를 다시 전송해주는 자바 프로그램)에 대한 스레드(프로세스 내에서 실행되는 흐름의 단위)를 생성하고 http 요청(httpServletRequest) 및 http 응답(httpServletResponse) 객체를 생성하여 전달한다.

3) 컨테이너는 서블릿을 호출한다.

4) 호출된 서블릿의 작업을 담당하게 된 미리 생성된 스레드는 요청에 따라 doPost() 또는 doGet()을 호출한다.

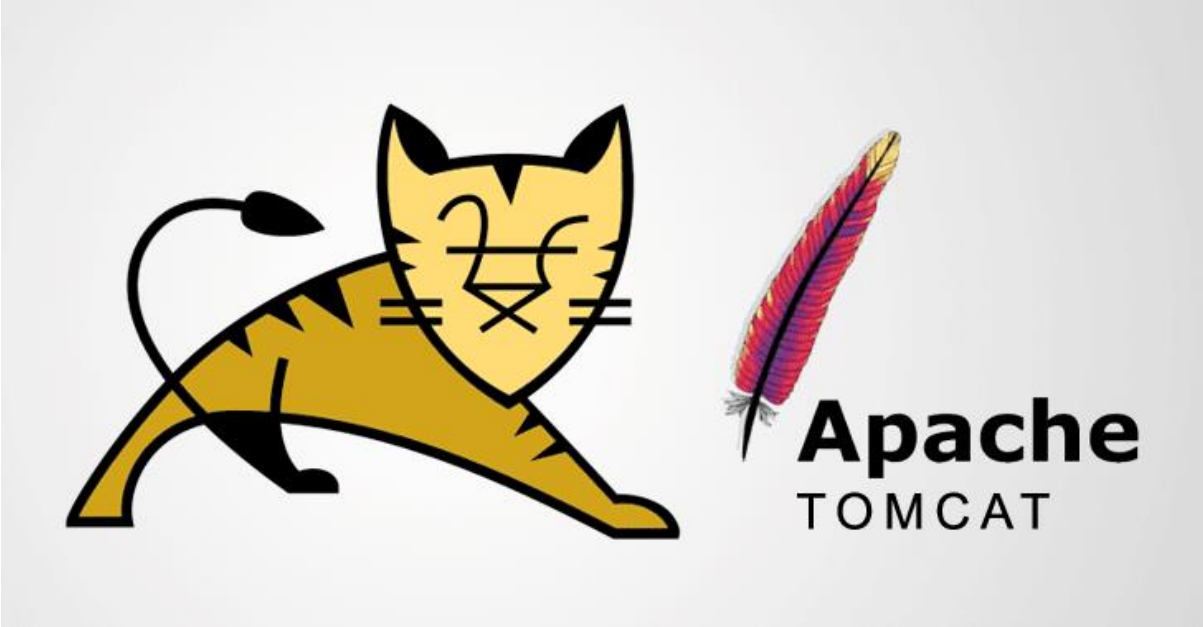
5) 호출된 doPost() 또는 doGet()메소드는 생성된 동적 페이지를 Response 객체에 실어서 컨테이너에 전달한다.

6) 컨테이너는 전달받은 Response 객체를 HTTPResponse 형태로 전환하여 웹서버에 전달하고 생성되었던 스레드를 종료하고 요청(httpServletReqeust)및 응답(httpServletResponse)객체를 소멸시킨다

2. 오픈 소스

- 오픈 소스 소프트웨어(Open Source Software).
- **공개적으로 액세스**할 수 있게 설계되어 누구나 확인, 수정, 배포가 가능한 코드라고 생각하면 된다.
- 공개성을 띄다보니 동료 평가와 커뮤니티 기반 프로덕션에 의지한다(**협업 방식**으로 개발이 됨).
- 커뮤니티가 개발하기 때문에 상용 소프트웨어보다 **저렴하고 유연하며 지속성**이 있다.
- 오픈 소스인 WAS 로는 **JBoss(레드햇), GlassFish(썬 마이크로시스템즈/오라클), Tomcat(아파치 소프트웨어 재단)** 등이 있다.

1. Tomcat(톰캣)



- 사전적 의미로 '수고양이'를 뜻한다.
- 아파치 소프트웨어 재단에서 개발한 WAS.
- **서블릿 컨테이너(또는 웹 컨테이너)만 있다**는 것이 특징이다.
- 웹 서버와 연동하여 실행할 수 있는 자바 환경을 제공하여 자바 서버 페이지(JSP)와 자바 서블릿이 실행할 수 있는 환경을 제공한다.
- 관리 툴을 통해 설정을 변경할 수도 있지만, XML(Extensible Markup Language) 파일을 편집하여 설정할 수도 있다. (XML 파일은 데이터의 이동, 구조 및 저장을 설명하는 텍스트 파일이라 생각하면 된다.)

- 톰캣(WAS)에는 아파치(Web Server)의 기능(웹서비스데몬, Httpd)를 포함하고 있다.
  그럼에도 톰캣 앞에 아파치를 두는 이유는 **하나의 서버에서 php 애플리케이션과 java 애플리케이션을 함께 사용**하거나, **httpd 서버를 간단한 로드밸런싱을 위해서 사용해야 할 때 필요**하기 때문이다.

  -> 앞서 Web Server 와 WAS 를 분리할 때 생기는 장점에 관해 얘기했었다. 그 장점 중에서는 **'여러 웹 어플리케이션 서비스 가능'** 과 **'여러 대의 WAS 를 연결 가능'** 이 있었다. 즉, **하나의 서버에서 php 애플리케이션과 java 애플리케이션을 함께 사용하는 경우** 이용의 효율성 및 장애 극복, 배포

및 유지보수의 편의성을 위해 Web Server 와 WAS 를 분리하여 사용하고, **로드밸런싱**을 통해 부하를 낮춰주기 위해 Web Server 와 WAS 를 분리하여 사용한다는 것이다.

위와 같이, 톰캣 앞에 아파치를 두는 이유를 Web Server 와 WAS 분리 시 생기는 장점에서 찾아볼 수 있을 것 같다.

- 여담으로, 톰캣은 CATALINA_HOME 이라는 변수를 사용한다(자바는 JAVA_HOME, 오라클 데이터베이스는 ORACLE_HOME). 개발자가 캘리포니아의 CATALINA 섬을 좋아해서 이렇게 지었다고 한다.

2. GlassFish(글래스피쉬)

- **자카르타 EE 기반** WAS
  - 자카르타 EE (Jakarta Enterprise Edition): 이전 명칭은 자바 플랫폼. 자바를 이용한 서버측 개발을 위한 플랫폼이다. 구체적으로 말하면 WAS 에서 동작하는 **장애 복구 및 분산 멀티티어를 제공하는 자바 소프트웨어의 기능을 추가한 서버를 위한 플랫폼**이다.

- 썬 마이크로시스템즈(2010 년에 오라클에 인수합병 됨)와 오라클의 탑링크(TopLink)를 기반으로 하고 있다.
  - 탑링크: 오라클에서 개발한 객체-관계 매핑 패키지. 프로그래밍 언어로 자바를 쓴다.

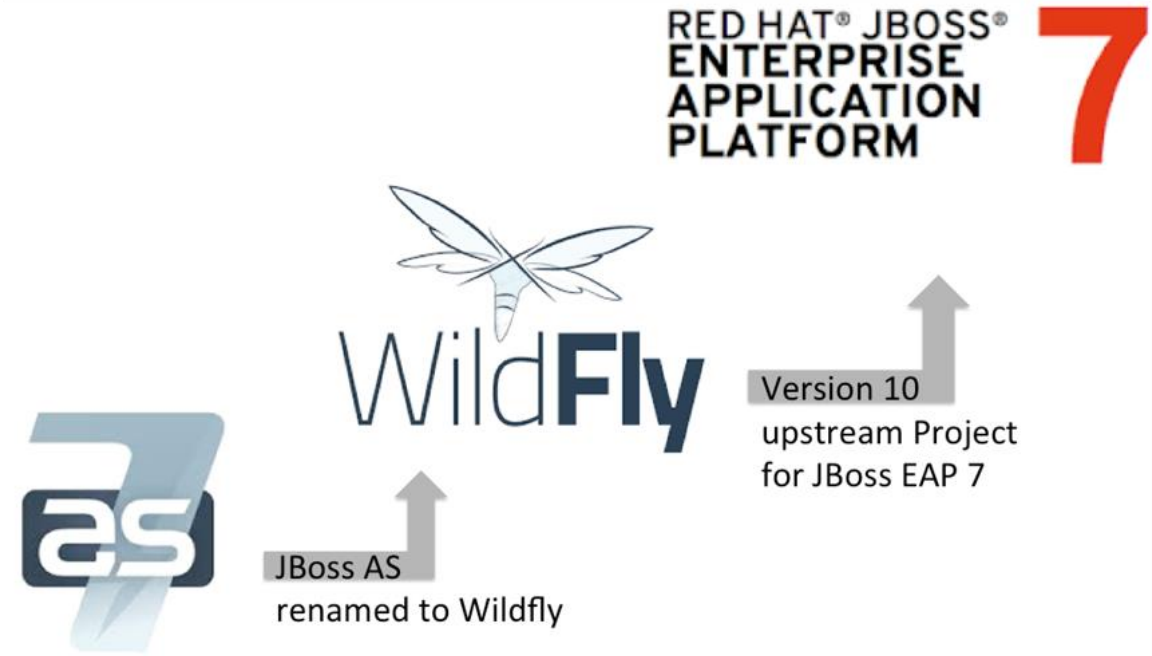
- 서블릿 컨테이너(웹 컨텐츠 제공)는 톰캣을 사용. 성능과 확장성을 높이기 위해 그리즐리라는 구성요소를 추가함
  - 그리즐리: 자바 언어의 네트워크 라이브러리. TCP 및 UDP 를 사용한 통신 기능 구현을 위한 프레임워크. HTTP/servlet 모듈 등을 제공해준다. 카카오톡 서버에도 사용되고 있다.

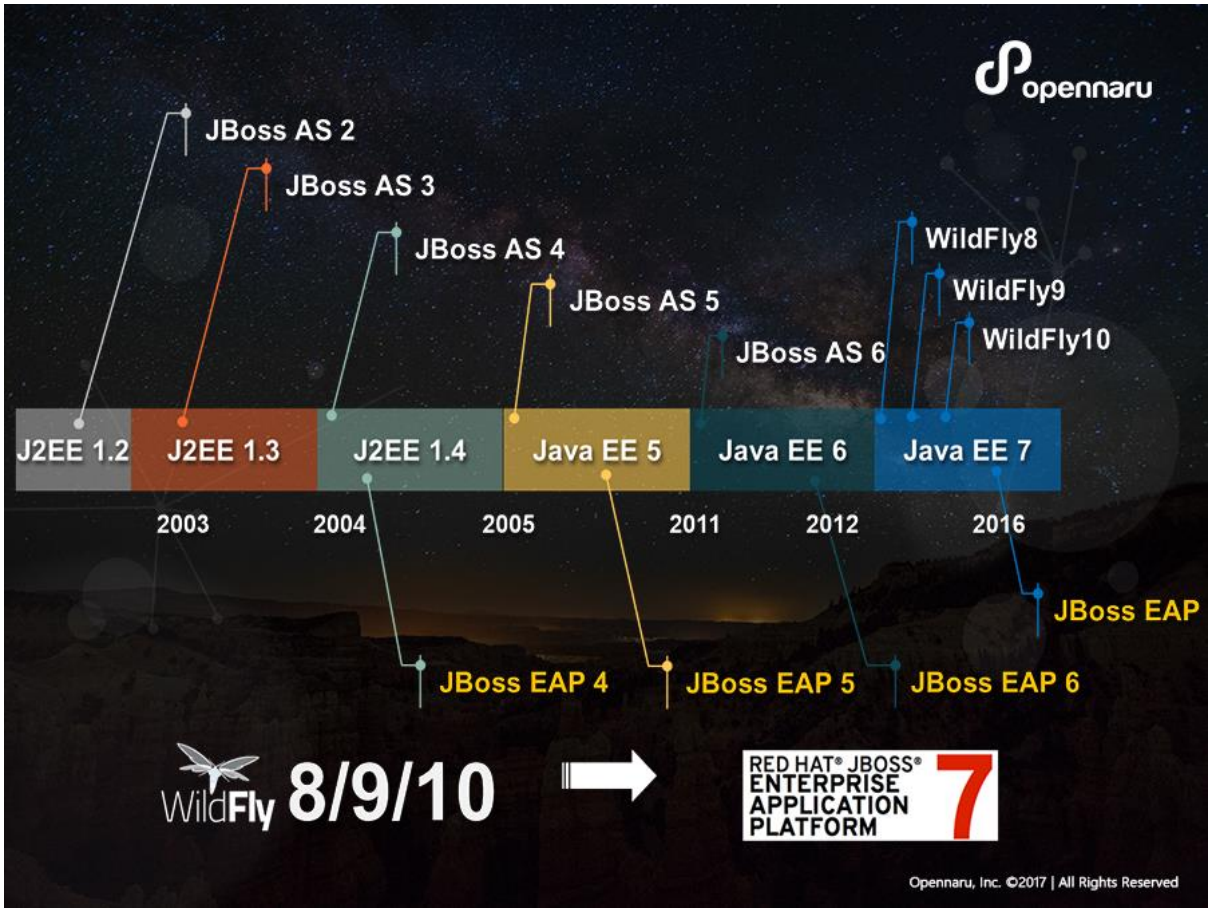
- 상업용 지원은 종료되었다. 하지만 오픈 소스 업데이트는 계속 하고 있다고 한다. 상업용 고객들은 오라클 웹로직 서버(상용 소프트웨어)로의 이전을 권고받았다고 한다.

- 자바 서버 페이지(jsp)형태의 웹 애플리케이션뿐만 아니라 Java EE5(자바 플랫폼)기반의 서비스를 할 수 있는 매우 강력한 애플리케이션 서버이다. 오픈 자바 개발 도구와 함께 자유롭게 소스 또는 바이너리를 가져다가 서버를 구축할 수 있게 해준다.

- 원격 보안 취약으로 인해, HTTP 를 통해서 네트워크 액세스 권한이 없는 공격자가 글래스피쉬를 손상시킬 수 있다는 취약점이 제기되었음. 그래서 대응 방안으로 지속적인 업데이트를 제시하였음.

3. JBoss(제이보스)


RED HAT® JBOSS®
ENTERPRISE
APPLICATION
PLATFORM
7
WildFly
Version 10
upstream Project
for JBoss EAP 7
JBoss AS
renamed to Wildfly


opennaru
JBoss AS 2
JBoss AS 3
JBoss AS 4
JBoss AS 5
JBoss AS 6
WildFly8
WildFly9
WildFly10
J2EE 1.2
J2EE 1.3
J2EE 1.4
Java EE 5
Java EE 6
Java EE 7
2003
2004
2005
2011
2012
2016
JBoss EAP
JBoss EAP 4
JBoss EAP 5
JBoss EAP 6
WildFly 8/9/10
RED HAT® JBOSS®
ENTERPRISE
APPLICATION
PLATFORM
7
Opennaru, Inc. ©2017 | All Rights Reserved

- 2006 년에 JBoss 프로젝트의 운영이 Red Hat 에 인수되었음.

- JBoss 의 WAS 가 고객의 혼돈을 막기 위하여 와일드 플라이라는 이름으로 변경되었다고 함(와일드 플라이 8/9/10).

- 와일프 플라이에서 더 발전되어 나온것이 바로 JBoss EAP7(JBoss Enterprise Aplication Platform 7)

- **JBoss EAP7**
  - 자바 플랫폼 Java EE 7 인증을 받은 제품
  - 와일드 플라이에 추가 테스트와 검증을 하여 안정성, 성능, 보안 수준을 높인 오픈 소스.
  - 와일드 플라이와의 차이점은, 와일드 플라이는 사용자 스스로 제품의 문제를 해결하면서 사용한다면(즉, 유지 보수를 지원 받지 않는다), JBoss EAP7 은 유료 유지 보수 기술 지원을 받을 수 있다는 것이다.
  - 쉽게 생각하면, 와일드 플라이에서 보안/성능에 관련된 패치가 이루어진 것이다.

JBoss 와 Tomcat 의 차이점

- JBoss 는 Tomcat 보다 훨씬 기능이 많다. JBoss 는 개발자에게 완전한 자바 엔터프라이즈 에디션(JEE)을 제공하는 반면, Tomcat 은 훨씬 제한적이다.
- 구체적으로 말해서, JBoss 는 서블릿 컨테이너와 Web Server 를 포함하는 JEE 스택이고, Tomcat 은 대부분 그냥 서블릿 컨테이너 + Web Server 이다.

- 기능 비교
  - JBoss 는 톰캣을 내장하고 있으나, 안정성, 성능, 가용성을 위하여 EJB, JMS, 클러스터링 기능을 제공한다.
  - JMS(JAVA Message Service): 자바가 데이터를 송수신하는 자바 API(Application Programming Interface, 소프트 웨어 간의 계약/연결)
  - EJB(Enterprise JAVA Bean): 썬 마이크로시스템즈가 기업 환경 개발을 단순화하기 위해 제창한 규약. 즉, 애플리케이션의 업무 로직을 가지고 있는 서버 애플리케이션이다.
  - 클러스터링: 클러스터는 여러개의 시스템이 하나의 거대한 시스템으로 보이게 만드는 기술. 디스크 공간 같은 하드웨어 자원을 공유할 수 있고, 여러 사용자에게 컴퓨팅 자원을 제공할 수 있다. 그래서 부하를 조정할 수 있고, 가용성이 높은 시스템을 구축할 수 있다.

기능	설명	Tomcat	JBoss AS
Servlet/JSP	JBoss는 Tomcat을 내장	O	O
JNDI	Tomcat은 읽기 기능만을 제공	O (limited)	O (clusterwide)
Database Connection Pooling		Not builtt-in (using DBCP)	O
JTA	Tomcat은 별도 제품을 통해 Transaction Service 가능	X	O
JMX		O	O
JMS	Tomcat은 독립적인 3rd Party 제품과 연동해서 사용 가능	X	O
EJB	Tomcat은 독립적인 3rd Party 제품과 연동해서 사용 가능	X	O
JAAS		X	O
Hot deploy	JBoss는 farm 디렉토리에 component패키지를 복사/삭제할 때 클러스터의 모든 서버에 deploy/undeploy 됨	O	O (more simple, clusterwide)
Clustering		O	O (improved performance)
Web Service	Tomcat은 별도 라이브러리를 통해 Service 가능	Not builtt-in	O

3. 상용 소프트웨어

- 상업적 목적으로나, 판매를 목적으로 생산되는 컴퓨터 소프트웨어
- 지적 재산이 철저히 보호된다. 상용 소프트웨어 소스 코드는 그 제품 또는 기술을 개발한 회사만이 보유할 수 있으며, 이를 허락받지 않고 사용할 경우 저작권법에 저촉된다.
- 상용 소프트웨어인 WAS 로는 Web Logic(BEA. BEA 는 2008 년에 오라클에 인수되었다), WebSphere(IBM), Jeus(Tmax) 등이 있다.

DB

1. 오픈소스 데이터베이스

1)오픈소스 데이터베이스란

오픈소스 데이터베이스는 코드베이스를 무료로 열람, 다운로드, 수정, 배포 및 재사용이 가능한 모든 데이터베이스 애플리케이션을 뜻합니다. 오픈소스 라이선스는 개발자들로 하여금 기존의 데이터베이스 기술을 사용하여 새로운 애플리케이션을 구축할 수 있는 자유를 제공합니다.

2)작동 방식

오픈소스 데이터베이스 관리 시스템은 개발자들이 조직과 애플리케이션을 위하여 정보를 저장할 수 있는 추상화 계층을 제공합니다.

-**관계형 DB**: 기존의 데이터 저장 접근법으로 키-값 짝을 사용하여 열과 행으로 이루어진 표의 형태로 정형 데이터를 저장하는 방식입니다. Join을 통해 정보 연계 가능하고, 트랜잭션의 안정성 확보를 위한 ACID(원자성, 일관성, 고립성, 지속성)과 정형적인 스키마 구조를 가집니다. 정규화된 데이터 모델과 데이터 무결성/정합성 보장하고 ,작은 크기의 트랜잭션 갖는 장점이 있고, 확장성의 한계와 클라우드 분산환경 부적합하다는 단점을 가집니다.

-**NoSQL(비관계형) DB**: 문서 데이터 스토어, 열-중심 데이터베이스, 키-값 스토리지 및 그래프 DB 등 대안적인 데이터 스토리지 아키텍처를 사용한 데이터 저장의 방식입니다. 비관계형 DB는 비정형 데이터를 다루는 데 적합합니다. 데이터의 수정/삭제가 거의 없고 Insert위주이며, 강한 Consistency 불필요하고, 노드 추가/삭제, 데이터 분산에 유연하다는 특징을 갖습니다. Web, SNS 환경의 다양한 정보를 검색 및 저장이 가능하지만 데이터 무결성/정합성 보장 못한다는 단점을 갖습니다.

2. 오픈소스의 종류

1.1 관계형 db

1) MySQL

MySQL은 제일 많이 사용되는 오픈소스 데이터베이스 입니다. 효율적인 아키텍처를 채택하여 빠른 성능과 쉬운 사용자 구성을 가능하게 해줍니다. 다양한 함수와 여러 기능면에서 상용 데이터베이스가 가지지 못한 점도 가지고 있습니다. 속도와 간결성, 안정성과 용이한 관리를 보장해주는 데이터베이스 관리 시스템입니다.

>특징

• ANSI SQL 표준을 준수하기 때문에 SQL 표준함수의 경우 타 DB와 동일하게 사용이 가능합니다.

• 자동 재시작/복구, 백업 및 PiT 복구, 논리적 온라인 핫 백업을 제공합니다.

• 스토리지 엔진은 동적으로 변경하여 관리할 수 있으며 사용하는 어플리케이션의 최적의 성능을 나타낼 수 있도록 도와줍니다.OLTP 및 트랜잭션을 제공합니다.

• 여러 분야에 적용되어 제품의 안정성이 검증되어 가장 많이 쓰이며, 선호되는 Open Source DBMS로 가격대비 효율성이 큰 특장점을 지니고 있습니다.

-단점

• 복잡한 쿼리에서는 성능이 저하됨.

• 트랜잭션 지원이 완벽하지 않음.

• 사용자정의 함수의 사용이 어렵고 유연하지 않음

2) PostgreSQL

PostgreSQL은 POSTGRES, 버전 4.2를 기반으로 하는 개체 관계 데이터베이스 관리 시스템입니다. 오픈 소스 코드이기 때문에 확장할 때 무료로 쉽게 사용할 수 있으며, GUI 인터페이스를 통해 DB 설계 및 관리 할 수 있습니다.

>특징

•이식성 (Portable)

지원하는 플랫폼: Windows, Linux, MAC OS/X 또는 Unix Platform등 다양

•신뢰성 (Reliable)

트랜잭션 속성인 ACID(원자성, 일관성, 고립성, 지속성)에 대한 구현 및 MVCC

• 확장성 (Scalable)

PostgreSQL의 멀티-버전 사용 가능

• 보안성 (Secure)

호스트 기반 접근제어, ssl 통신을 통한 클라이언트와 네트워크 구간의 전송 데이터 암호화

• Advanced

PostgreSQL upgrade, 모티너링 및 관리, 튜닝까지 가능, 스크립트 언어 지원(Perl, java, php)

-단점

•수많은 사용자를 동시에 서버에 접속 할 때 불안.

•기술지원에 대한 안정성 확보가 부족하며, PostgreSQL 매뉴얼을 확인 해보면 대부분 에러에 대한 내용을 사용자에게 맡김.(풍부한 데이터 사전이 존재하지 않는다.)

• 정기적인 Vacuum 작업 필요

3) MariaDB

MariaDB는 오픈 소스의 관계형 데이터베이스 관리 시스템(RDBMS)이다. MySQL과 동일한 소스 코드를 기반으로 하며, GPL v2 라이선스를 따릅니다. 오라클 소유의 현재 불확실한 MySQL의 라이선스 상태에 반발하여 만들어졌으며, 배포자는 몬티 프로그램 AB(Monty Program AB)와 저작권을 공유해야 합니다.

>특징

• MySQL과의 호환성

• 보안

암호화 된 연결, 암호화 된 파일 / 로그, 암호화 된 버퍼 및 동적 데이터 마스킹을 통해 보안을 제공합니다.

• 확장성

병렬 쿼리로 데이터베이스/데이터 웨어하우스를 확장하고, 복제 또는 multi-writer 클러스터링을 사용하여 읽기를 확장하고, 분산 SQL(MariaDB Xpand)을 사용하여 읽기와 쓰기를 모두 확장합니다.

• 고성능

고성능을 실현하기 위해 스레드 풀링, 병렬 복제, 멀티 소스 복제 등의 기능이 표준으로 구비되어 있습니다.

- 단점

• 복잡한 쿼리에서는 성능이 저하

• 대량 데이터를 조회할 때는 성능 저하가 발생

1.2 noSQL db

1) MongoDB

MongoDB는 문서 지향 데이터 모델(Document DB)을 사용하는 데이터베이스입니다. 이러한 유형의 모델을 사용하면 정형 및 비정형 데이터를 보다 쉽고 빠르게 통합할 수 있다는 장점이 있습니다.

>특징

•Schema-less 구조

다양한 형태의 데이터 저장 가능하고 데이터 모델의 유연한 변화 가능 (데이터 모델 변경, 필드 확장 용이)

• Read/Write 성능이 뛰어남

•Scale Out 구조

많은 데이터 저장이 가능하고 장비 확장이 간단함

• JSON 구조

데이터를 직관적으로 이해 가능

• 사용 방법이 쉽고, 개발이 편리함

- 단점

• 데이터 업데이트 중 장애 발생 시 데이터 손실 가능

• 충분한 메모리 확보가 필요함, 데이터 공간 소모가 많음

• 트랜잭션 지원이 RDBMS 대비 미약함

2) CouchDB

스케일러블 아키텍처를 쉽게 이용하고 보유하는데 초점을 둔 오픈DB 소프트웨어입니다. 도큐먼트 지향 NoSQL데이터 베이스 구조를 갖추고 있으며 범용 병렬 프로그래밍 언어인 얼랭(Erlang)으로 구현되어 있습니다.

>특징

• 문서지향(Document-oriented) 데이터베이스

CouchDB는 하나 이상의 필드를 가진 JSON 형식의 문서로 데이터를 저장한다.

• Map/Reduce 뷰와 색인

자바스크립트를 이용해 Map/Reduced 작업을 진행하여 값을 구할 수 있으며, 인젝스 뷰등을 생성하여 관리할 수 있다.

• 분산 아키텍처와 replication

양방향 복제를 허용하며, 복제된 db에서 변경이 일어나더라도 다른 복제본과 데이터를 서로 동기화한다.

• REST API

url을 기준으로 post, get, put, delete를 이용해서 crud 처리가 가능하다.

• Eventual consistency

CouchDB는 분산형 컴퓨팅에서 사용되는 동시성 일관성 모델인 eventual consistency를 보증한다.

• bulit for offline

스마트폰과 같은 기기에도 데이터 복제가 허용되며, 온라인 상화에서 데이터 동기화 실시

3) Cassandra

카산드라는 구글의 BigTable 컬럼 기반의 데이터 모델과 FaceBook에서 만든 Dynamo의 분산 모델을 기반으로 제작되어 Facebook에 의해 2008년에 아파치 오픈소스로 공개된 분산 데이터 베이스입니다.

>특징

- 분산 대용량 저장소(NoSQL)
- 수평적 확장이 용이한 Ring 구조의 클러스터 아키텍처
- Write가 Read 보다 빠르고, Read도 RDB(relational database)보다 빠름.
- 서비스 중에 노드를 추가하여 용량 확장 가능

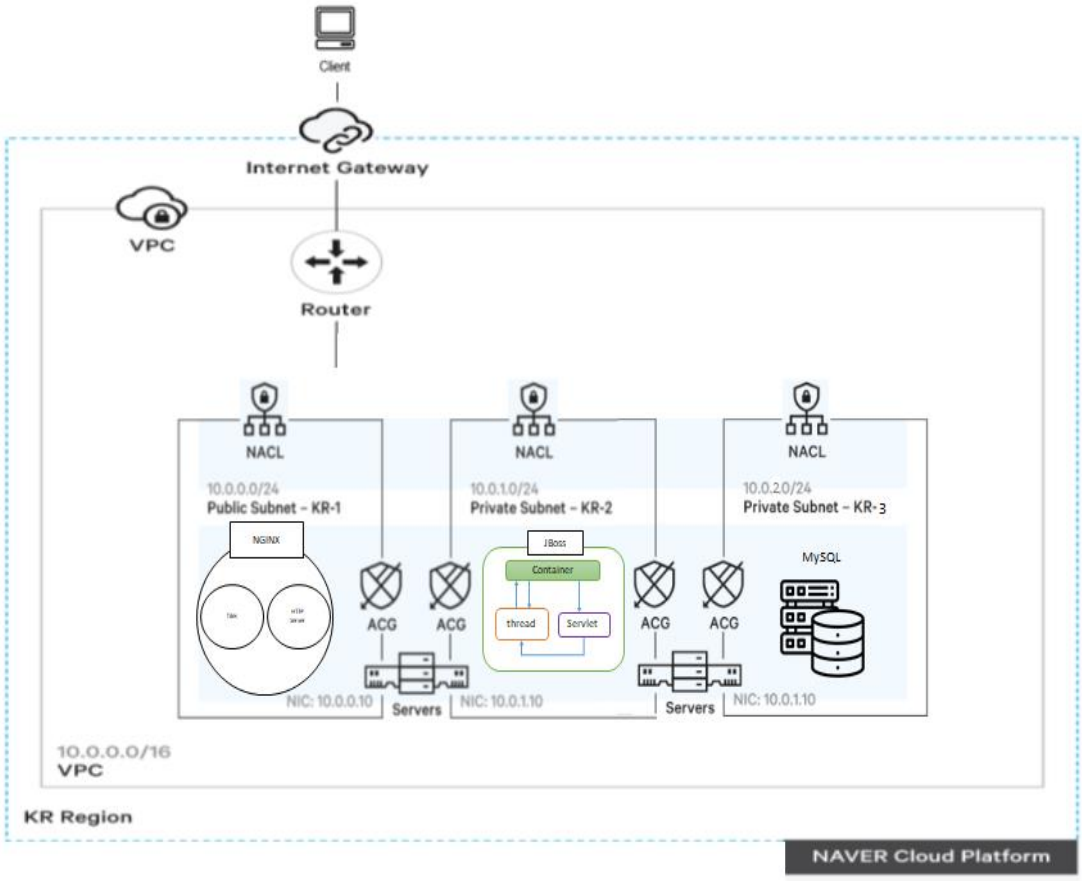
-단점

•높은 진입장벽

•복잡한 조건의 검색 불가

•데이터 입력시 자동화 처리가 어려움

표준구성

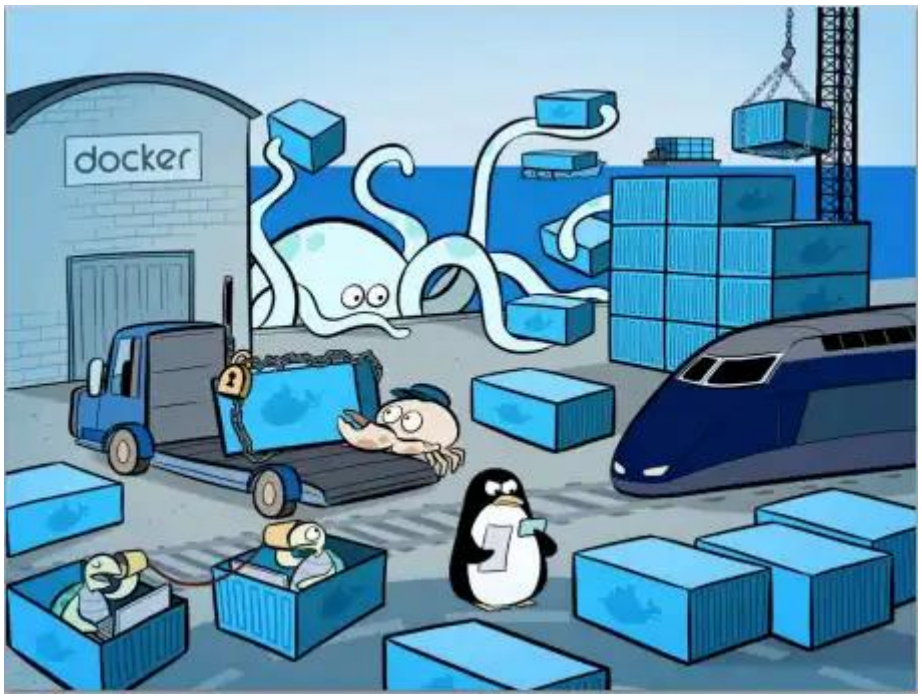

Client
Internet Gateway
VPC
Router
NACL
NACL
NACL
10.0.0.0/24
Public Subnet - KR-1
10.0.1.0/24
Private Subnet - KR-2
10.0.2.0/24
Private Subnet - KR-3
NGINX
JBoss
Container
thread
Servlet
MySQL
ACG
ACG
ACG
ACG
ACG
NIC: 10.0.0.10
Servers
NIC: 10.0.1.10
Servers
NIC: 10.0.1.10
10.0.0.0/16
VPC
KR Region
NAVER Cloud Platform

도커, 쿠버네티스 헬름, 앤서블

CLOIT

1. 도커

- **컨테이너 기반의 오픈소스 가상화 플랫폼**
- 쉽게 말하면, 도커는 컨테이너를 관리하는 플랫폼이다.



- 여기서 말하는 컨테이너란 애플리케이션을 **환경에 구애받지 않고 실행**하는 기술을 말한다.
- 서버 제작과정에 견고함과 유연성, 편리함이 좋고 다른 사람이 만든 서버를 소프트웨어 사용하듯이 바로 가져다가 사용할 수 있는 장점이 있다. 또한 여러 대에 배포할 수 있는 확장성도 지녔다.

도커를 사용하는 궁극적인 이유

1) 환경 표준화

- **환경이 일정하지 않아서 생기는 문제를 해결**할 수 있다.
  개발하려고 하는 여러 어플리케이션을 구동할 때 운영체제가 서로 다르면

환경에 따라 조금씩 변경할 부분이 생긴다. 같은 Linux 라고 하더라도 Ubuntu, CentOS, Debian 은 서로 다른 환경이므로 여러 버전이 존재할 수도 있다.

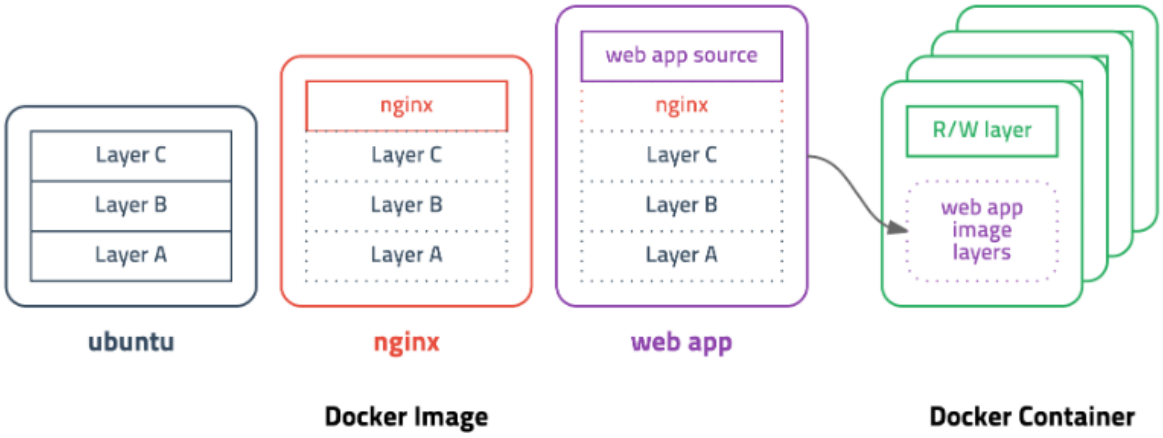
2) 개발 혹은 실행에 대한 환경 설정의 코드화

- 개발된 어플리케이션을 내 컴퓨터에 설치해 사용하고자 할 때 '내 컴퓨터' 혹은 '내 사용 목적'에만 맞는 설정이 따로 필요할 수 있다. 예를 들어 환경 변수처럼 같은 OS 라 하더라도 사용자에 따라 달라지는 구성이 있다. 만약 어플리케이션을 설치하거나 실행할 때 홈 디렉토리에 저장해야 한다면 모든 사용자가 같은 홈 디렉토리를 사용하지는 않을 것이다. 이 외에도 방화벽 설정, 사용자 권한 설정, Port 설정 등 어플리케이션을 설치할 때 컴퓨터에 맞게 변경해 줘야하는 부분들이 있다. 이러한 문제를 해결하기 위해 수작업을 하게 된다면 많은 시간이 걸린다. 하지만 도커를 사용하면 **환경 설정을 코드화 해 놓았기 때문에** 편하게 작업을 할 수 있는 것이다.

=> **즉, 표준화의 관점에서 이유를 찾을 수 있다!**
=> **도커 컨테이너는 소프트웨어를 소프트웨어 실행에 필요한 모든 것을 포함하는 파일 시스템 안에 감싼다. 이는 실행 중인 환경에 관계없이 언제나 동일하게 실행될 것을 보증한다.**

도커 사용

- 이미지: **컨테이너 실행에 필요한 파일과 설정 값 등을 포함하고 있는 것. 변하지 않음**
- 도커는 레이어(layer)라는 개념을 사용하고, 유니온 파일 시스템을 이용하여 **여러 개의 레이어를 하나의 파일시스템으로 사용할 수 있게 해준다.**

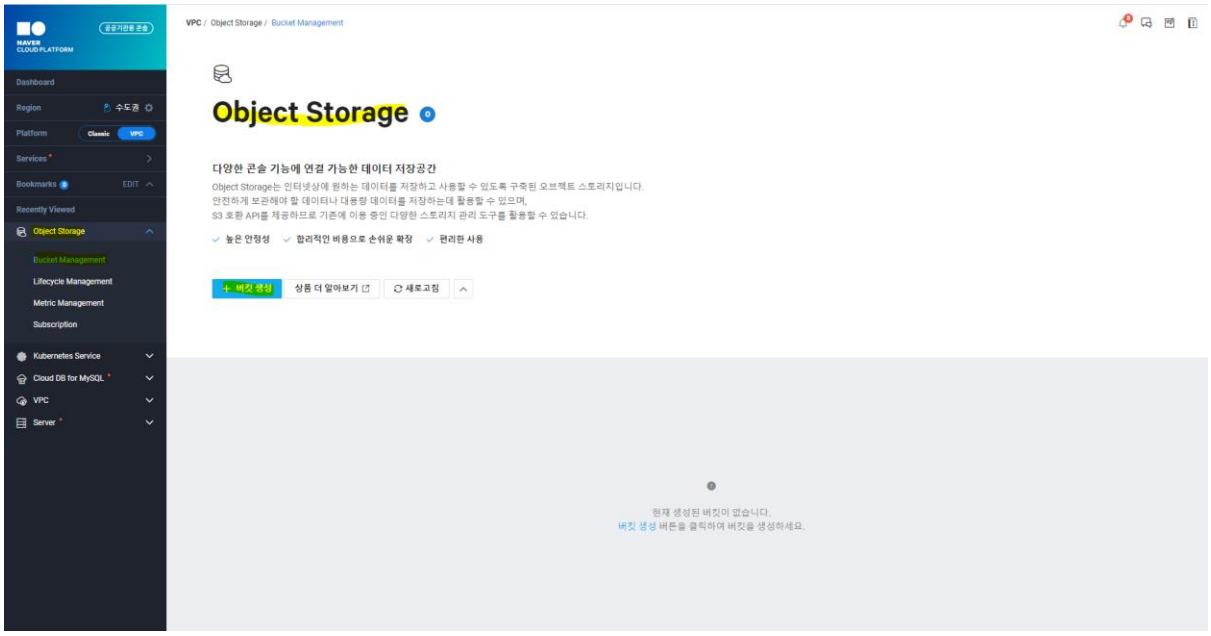


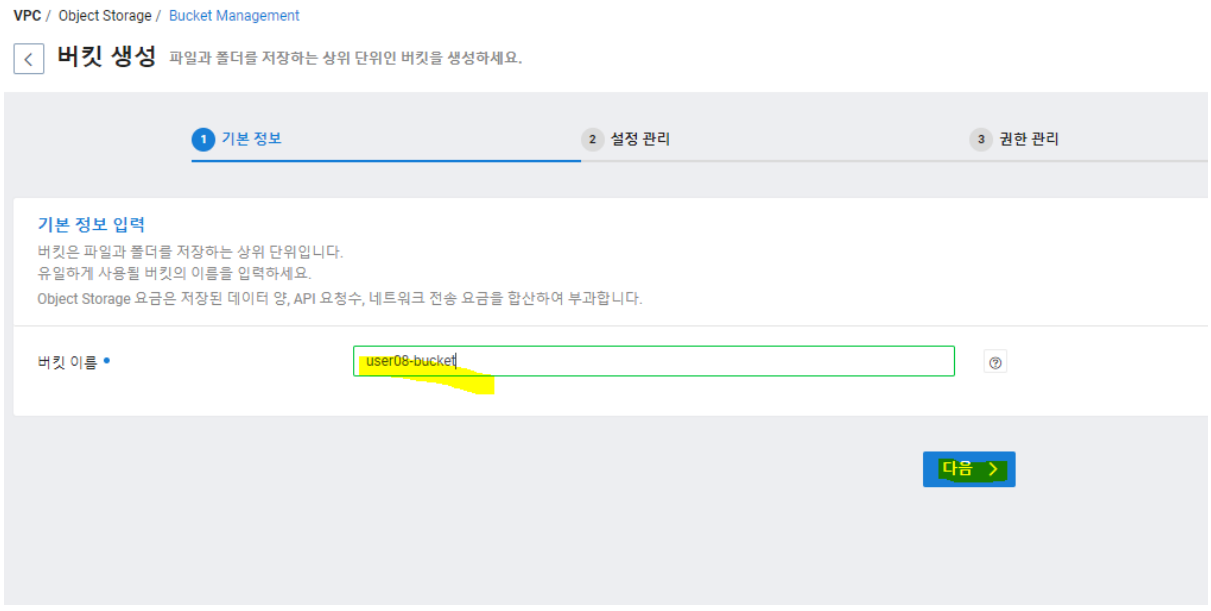
- webapp 소스를 수정하면 A, B, C, nginx 레이어를 제외한 **새로운 source(v2) 레이어만 다운받으면 되기 때문에 굉장히 효율적으로 이미지를 관리**할 수 있다.

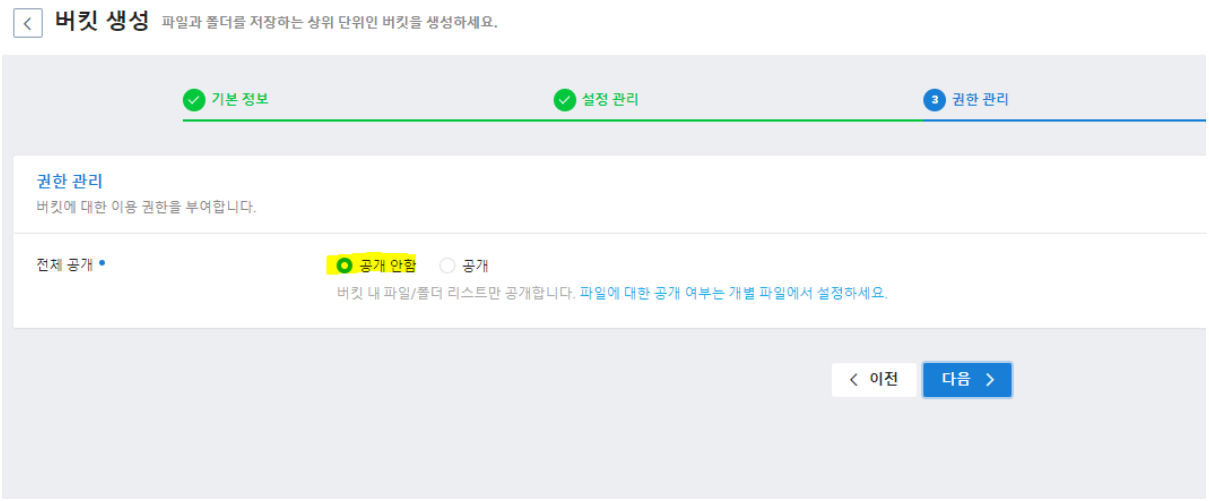
네이버 클라우드에서 컨테이너로 이미지 관리해 보기(생성 부분만 해보기)

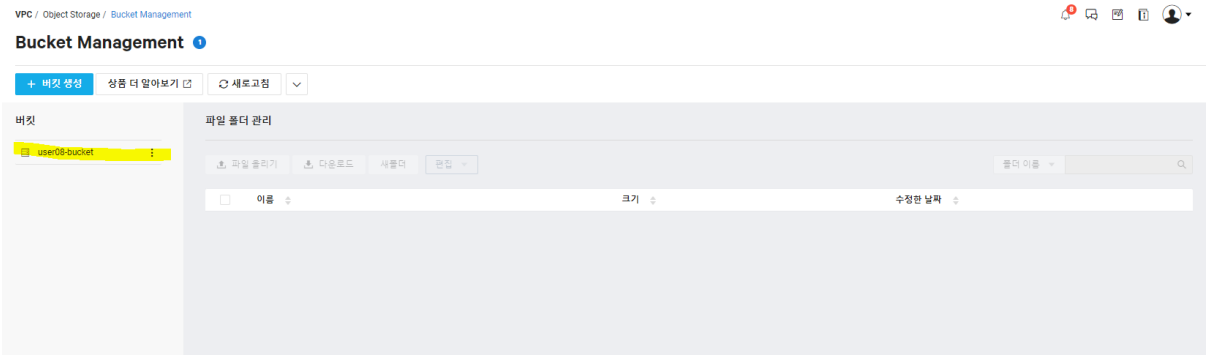
1) Object storage 생성

- Container Registry 를 생성하기 전에 Object Storage 를 먼저 생성해야 한다.

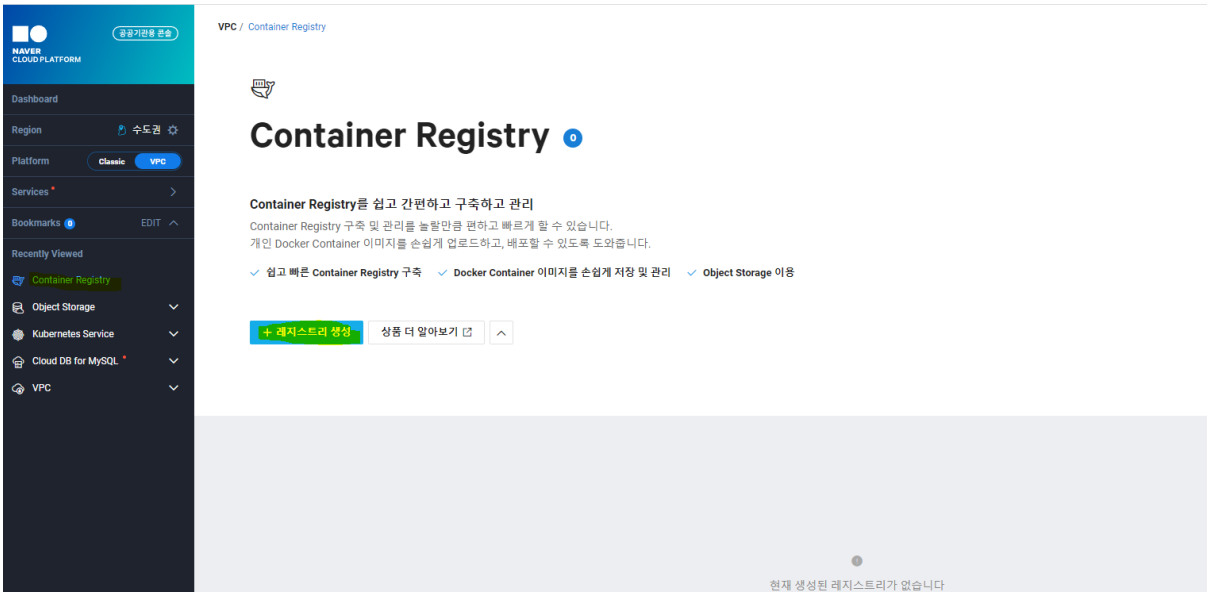


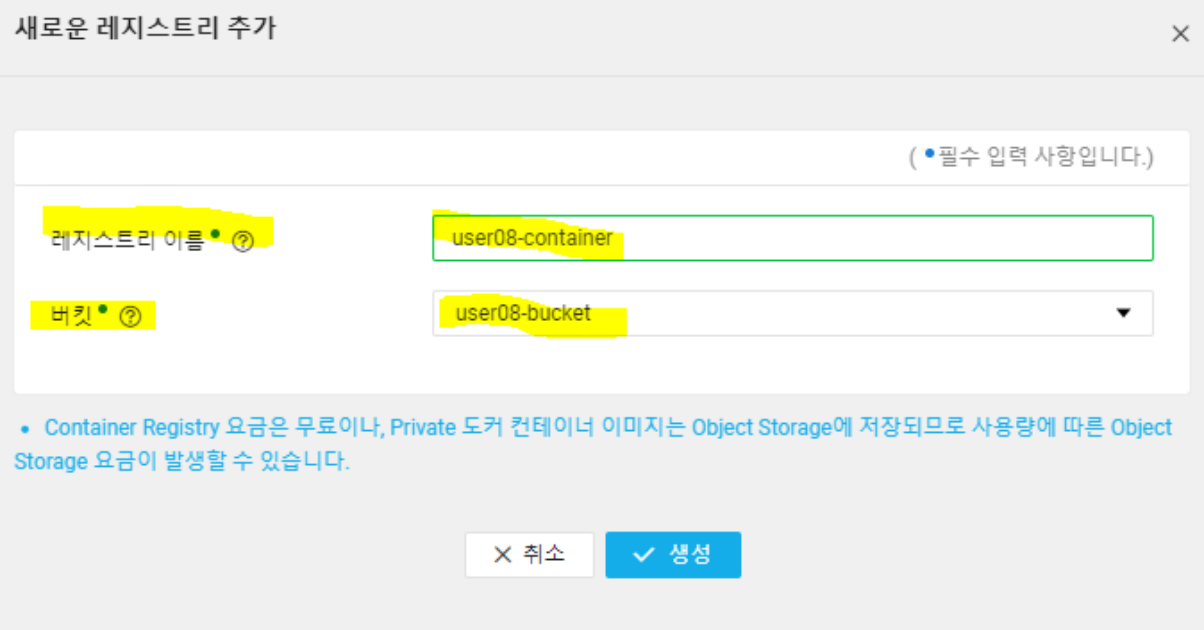


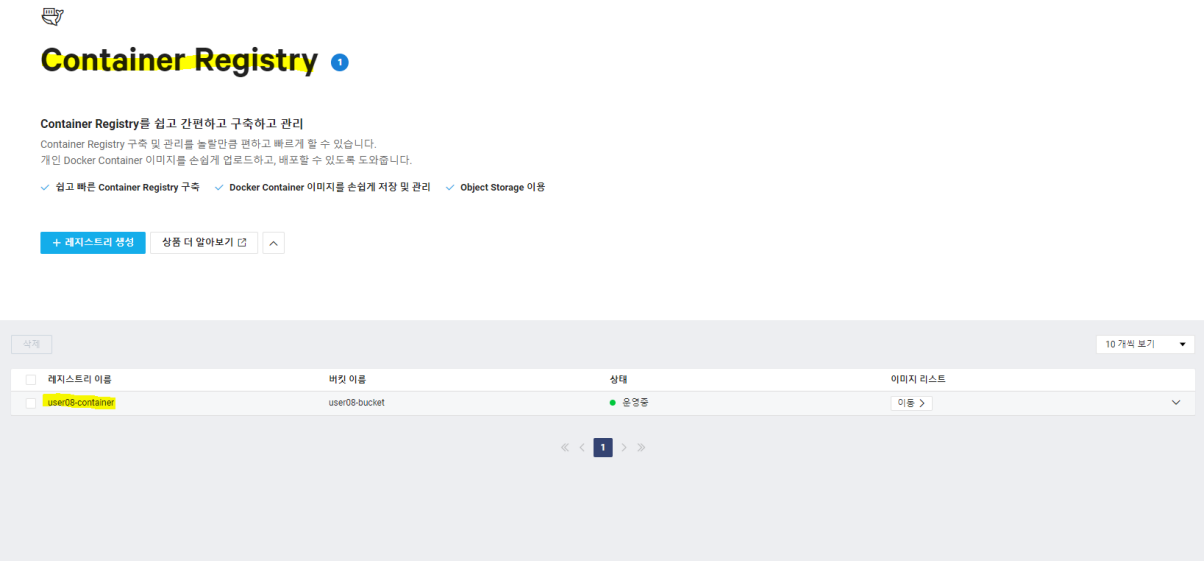




2) Container Registry 생성







컨테이너 레지스트리는 네이버 클라우드 플랫폼의 서버 상품 및 도커 CLI 와 통합하여 사용할 수 있다. 도커 CLI 를 사용해 도커 컨테이너 이미지를 컨테이너 레지스트리로 쉽게 전달할 수 있고, 이미지를 직접 운영 서버에 가져와 배포할 수 있어 개발부터 서비스 배포까지 필요한 작업을 간소화할 수 있다.

2. Kubernetes Helm

- 쿠버네티스 헬름이란 쿠버네티스 템플릿들을 모아서 관리하는

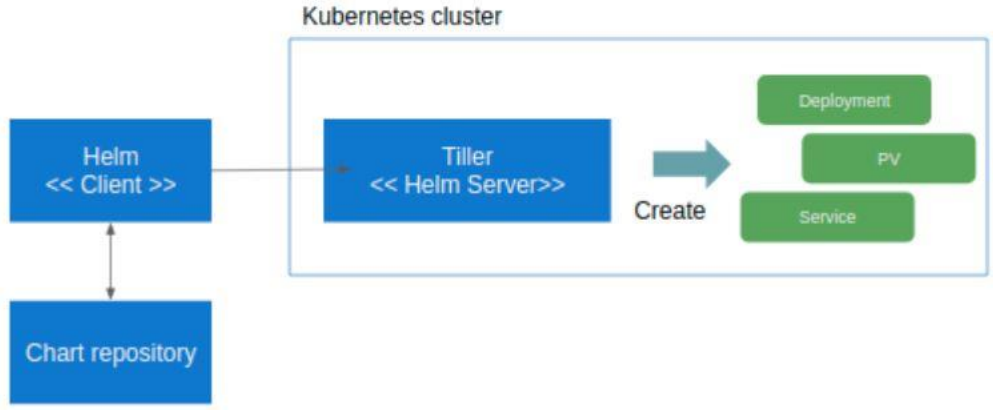
  "패키지 매니지먼트 도구"이다.

- Docker 의 출시가 container 을 혁신하고,

  Container 을 쉽게 관리하고 배포해주는 kubernetes 가 등장했으며

  Kubernetes 를 패키지로 관리해주는 것이 Helm 이다.

- 일반적으로 하나의 소프트웨어를 쿠버네티스에 배포하려면 컨테이너만을 배포해서는 사용하기 어려운 경우가 많다.
- 외부로 IP 를 노출하기 위한 서비스 배포, pod 를 관리할 deployment, 디스크 볼륨과 기타 정책 등등.
- Helm 은 애플리케이션 컨테이너 배포는 물론이고, 필요한 쿠버네티스 리소스를 모두 배포해주는 역할을 하며, 이 배포를 패키지 형태로 한다.
- IaaC fnf rbgysgjsns Terraform + 패키지 매니저인 npm 정도의 개념.



<Helm 개념도>

- CLI 툴인 클라이언트로 Helm 이 있다.

  클라이언트는 Helm 서버 모듈과 통신을 하는데, 이 서버를 Tiller 라고 하고, Tiller 은 쿠버네티스 클러스터 내에 설치된다.

- helm 을 통해 인스톨하는 패키지가 Chart.

  Chart 는 템플릿으로 설치하고자 하는 쿠버네티스 리소스의 설치 스크립트가 됨.

  Chart 들은 Helm Chart Repository 에 저장할 수 있다.

- 헬름은 차트와 차트 압축 파일을 만들고, 차트 저장소에 연결해 쿠버네티스 클러스터에 차트를 설치, 삭제할 수 있다. 배포 주기 관리도 가능.
- 헬름을 이용해서 잘 정리된 차트들로 필요한 애플리케이션을 빠르게 설치할 수 있다.

헬름 2 와 헬름 3

- 2 는 CLI 인 `헬름 클라이언트`와 쿠버네티스 클러스터 안에서 헬름 클라이언트의 명령을 받아 쿠버네티스 API 와 통신하는 `틸러 서버`로 구성된다.
- - 헬름 클라이언트는 로컬 서버에 차트를 만들거나, 차트 저장소들과 클러스터에 실행중인 애플리케이션(헬름 차트로 실행) 릴리즈를 관리하는데 필요한 요청을 하는 역할
- - 틸러 서버 : 헬름 클라이언트의 요청을 받아 실제로 처리하는 역할. 차트나 릴리즈를 만드는 것, 차트와 설정의 조합, 클러스터의 차트와 릴리즈의 설치와 관리를 담당한다. 틸러서버와는 gRPC 를 사용해 통신한다.

헬름을 사용하는 이유

- 쿠버네티스 애플리케이션 관리

  헬름 차트는 복잡한 쿠버네티스 애플리케이션도 편리하게 정의하여 설치하거나 업그레이드할 수 있다.

  편리한 차트 작성, 버전 관리, 공유 및 게시

- 복잡성 관리

  차트는 매우 복잡한 앱도 표현하고, 반복적인 애플리케이션 설치를 제공하며 단일 권한으로 서비스할 수 있음.

- 쉬운 업데이트

  즉석(in-place) 업그레이드와 커스턴 훅은 통해 업데이트하는 수고를 줄여줌.

- 간단한 공유

  차트는 버전 관리, 공유, 퍼블릭 혹은 프라이빗 서버 호스팅이 편리함.

- 롤백

- 쿠버네티스 위에서 동작하는 애플리케이션은 다양한 리소스의 조합으로 구성되며, 애플리케이션 배포 시 이런 리소스를 개별적으로 생성하는 것이 아닌, 하나의 패키지로 묶어서 배포한다.

  패키지로 묶어서 관리하면 여러 리소스들을 동시에 추가 및 업그레이드하기 편리해진다.

  예를 들어 `.yaml`은 정적 파일이기 때문에 리소스별로 yaml 파일을 만들어야 된다. 때문에 많은 리소스를 관리하게 될 때 yaml 파일에 대한 유지보수가 힘들어지게 된다.

- docker 가 단순히 프로세스 레벨에서 외부의 것을 가져다 쓸 수 있게 해준 것이라면, 쿠버네티스는 helm 을 이용, 프로세스(Pod)와 네트워크(Service), 저장소 (Persistent Volume)등 애플리케이션에서 필요한 모든 자원들을 외부에서 가져올 수 있게 한다.

3. 앤서블

앤서블이란

:Ansible 은 IaC 를 지향하는 오픈소스 기반의 자동화 관리 도구 입니다.
자동 구축/관리 하려는 원격 인프라에 무언가 명령을 전달하는 방식으로 동작 합니다.

-Infrastructure as Code (IaC)
인프라를 코드 기반으로 자동 설치 및 구축/관리/프로비저닝 하는 프로세스

특징

1) Agentless
명령을 내려주는 Controller 서버와 원격 서버에 설치된 Agent 들이 명령을 주고 받는 방식으로 동작하는 Chef/Puppet 과 같은 기존 IaC 솔루션들과 다르게 앤서블(Ansible)은 SSH 를 기반으로 원격 서버에 명령을 전달하기 때문에 각 원격 서버에 접속해서 agent 를 설치해줄 필요가 없습니다.

2) 접근 용이성
앤서블은 playbook 이라는 원격 서버에 전달할 명령들을 모아둔 명령집을 YAML 형식의 파일로 관리 합니다. 가독성이 좋은 YAML 파일로 접근이 용이합니다.

3) 멱등성 (idempotence)
멱등성이란 여러번 수행해도 같은 결과를 내는 성질을 말합니다.
앤서블은 YAML 로 관리되는 명령집을 여러번 수행하더라도 언제나 같은 결과가 나올 수 있도록 여러가지 관리를 합니다.

설치와 초기설정

1) Ansible 설치

- EPEP yum 레포지토리 추가
  :# yumm install -y epel-release
- 목록확인
  :# yum repolist
- ansible 설치
  :# yum install -y ansible
- ansible 설치 확인
  :# ansible --version

2) ssh key 생성

- Controller 서버에서 ssh key 를 하나 생성
  :# ssh-keygen
- 생성된 key 를 원격서버에 복사
  :# ssh-copy-id [원격서버계정 ID]@[원격서버 IP]
- 원격 서버로 ssh 접속을 시도
  :# ssh [원격서버계정 ID]@[원격서버 IP]

3)인벤토리 작성

- /etc/ansible/hosts 가 인벤토리 파일 열어서 원격서버 ip 작성
  :# vi /etc/ansible/hosts-서버들 정상 접속 확인
- 인벤토리 목록에 있는 서버들로 접속이 정상적으로 이루어지는지를 확인
  :# ansible all -m ping

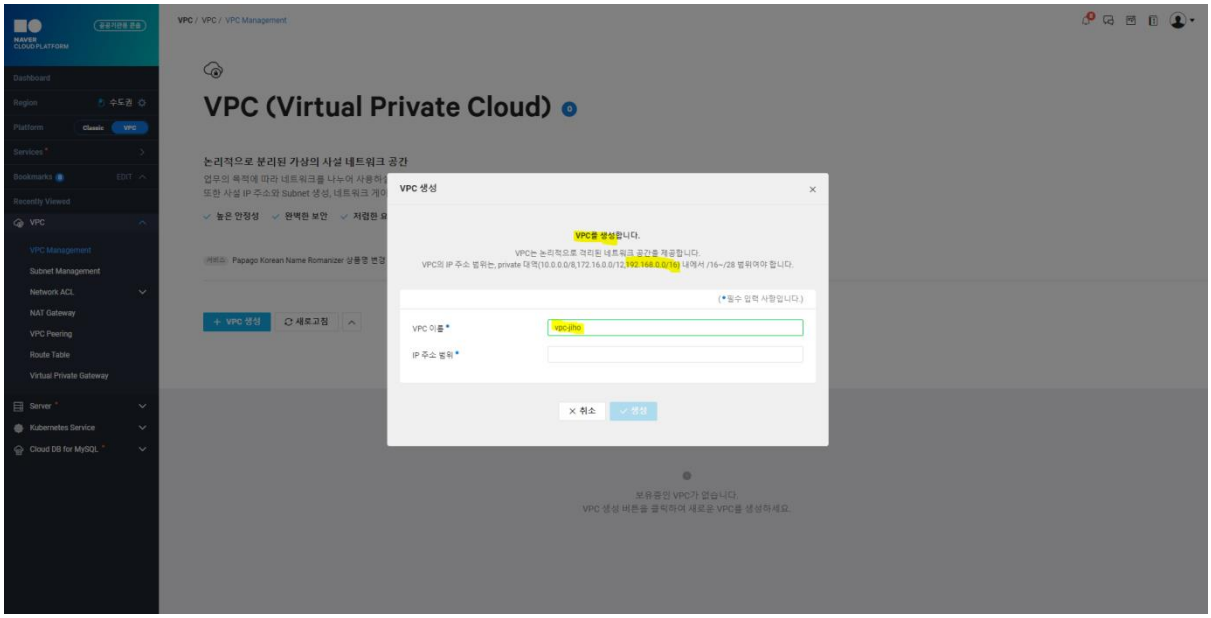
명령

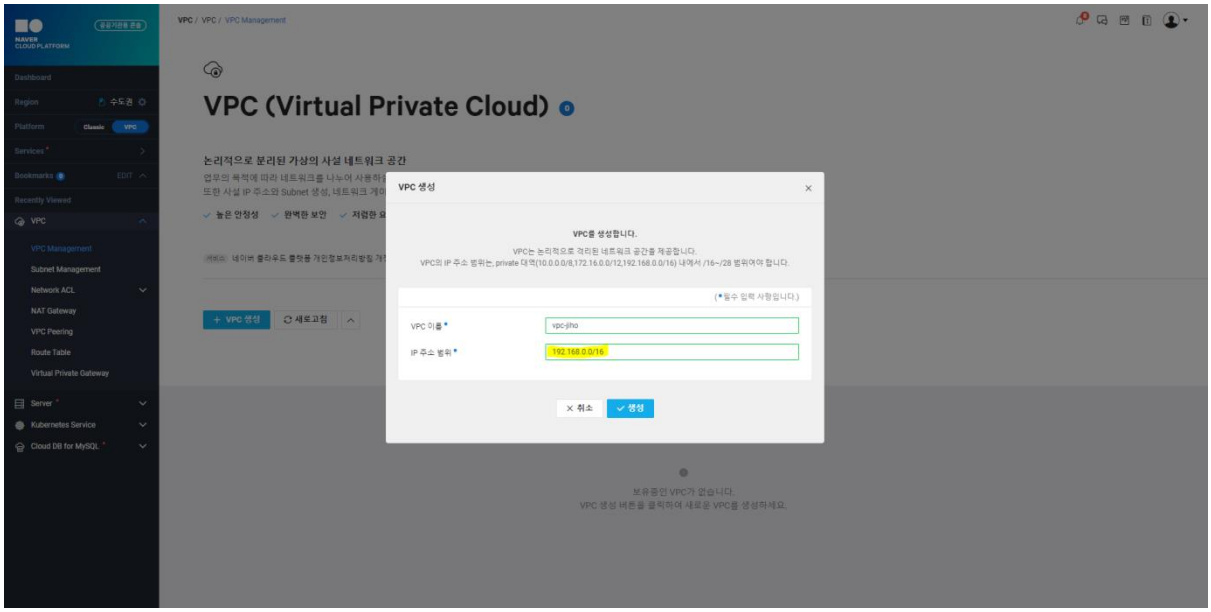
- 명령구조: $ ansible [host 또는 host 그룹] options
  -[host 또는 host 그룹]: ansible 명령을 전달할 원격 서버들을 설정

실제 구성 실습 (조금)

- VPC(Virtual Private Cloud): 다른 네트워크와 분리되어 있어 IT 인프라를 안전하게 구축하고 간편하게 관리할 수 있는 사설 네트워크

1. VPC 생성

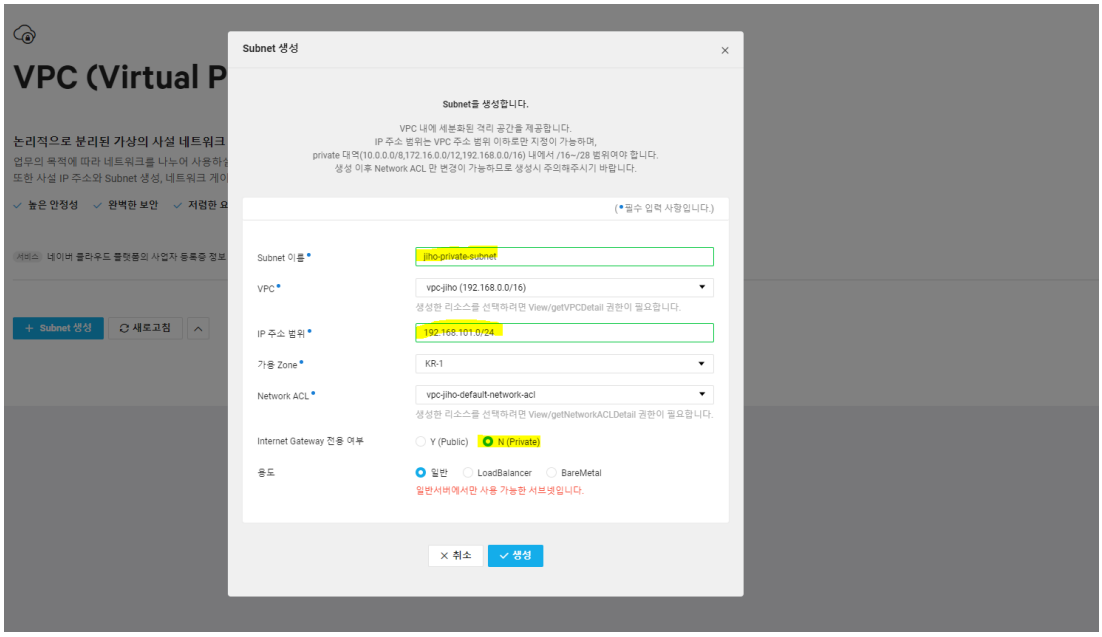


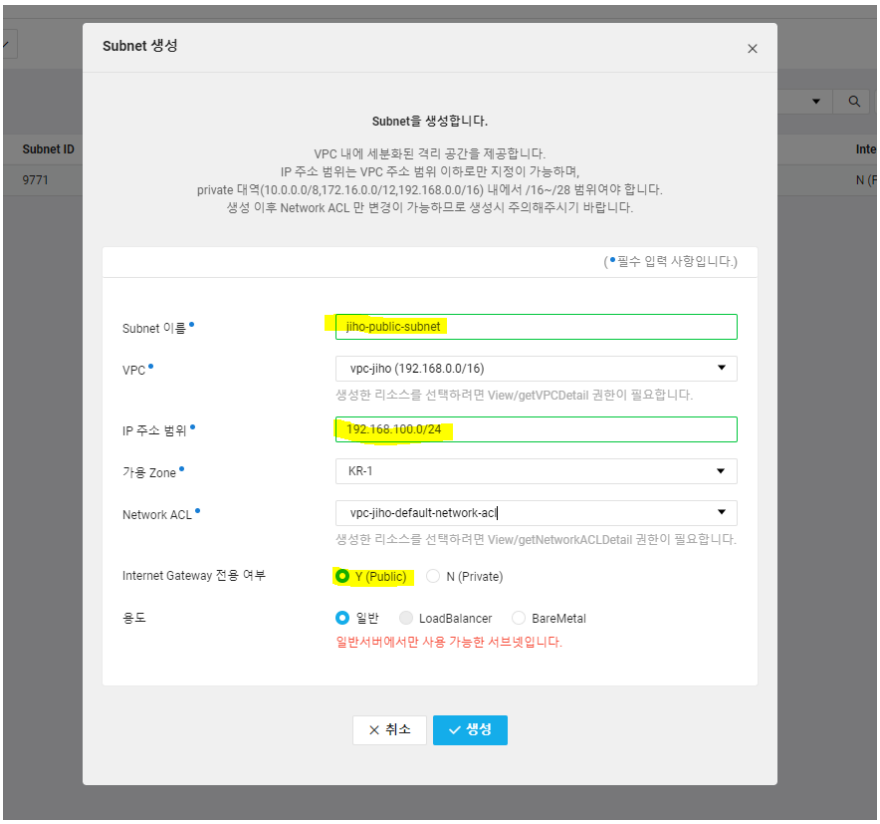


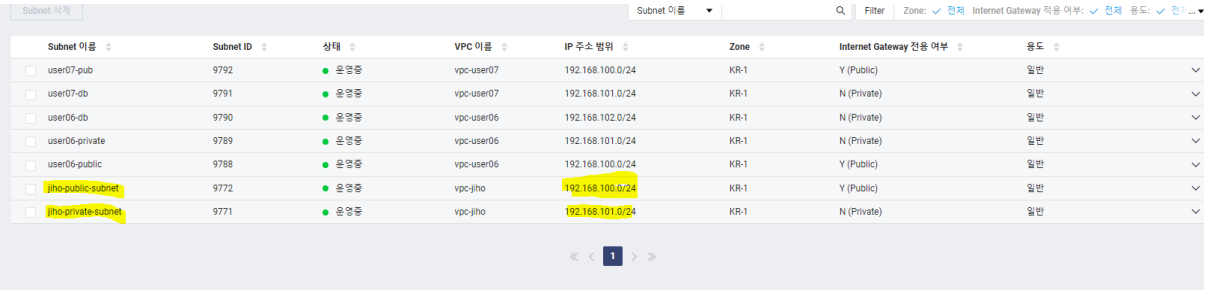
2. 서브넷 생성

- 서브넷: 부분망. 네트워크 영역을 부분적으로 나누어 IP 를 사용하는 네트워크 장치 수에 따라 효율적으로 사용할 수 있게 함.

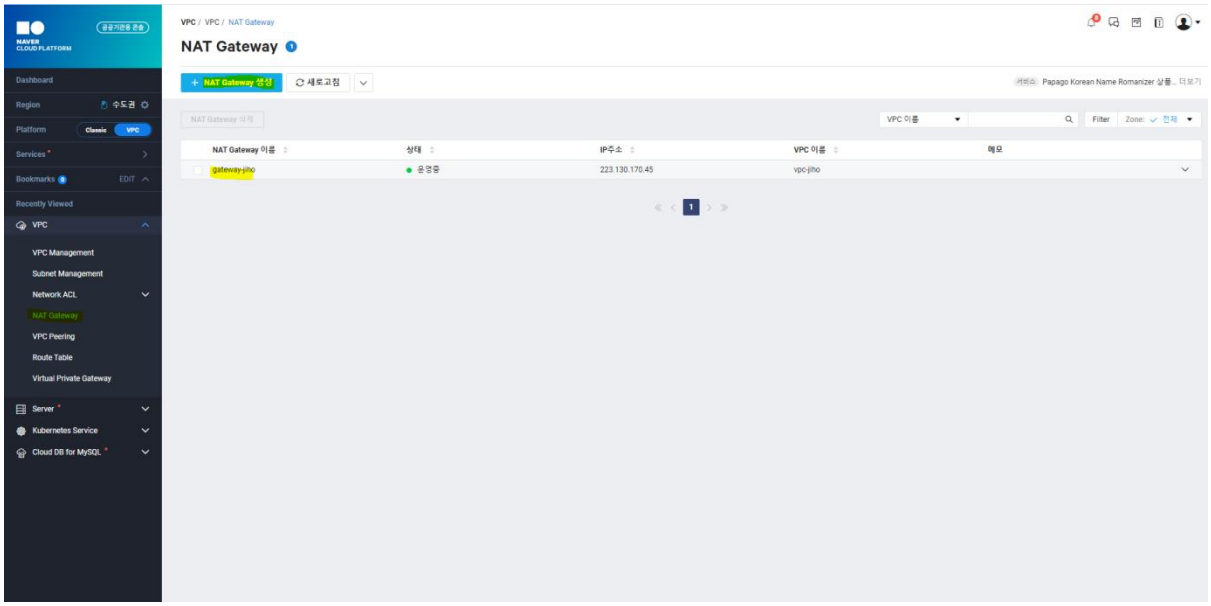
- private 1 개 / public 1 개(web server 용)



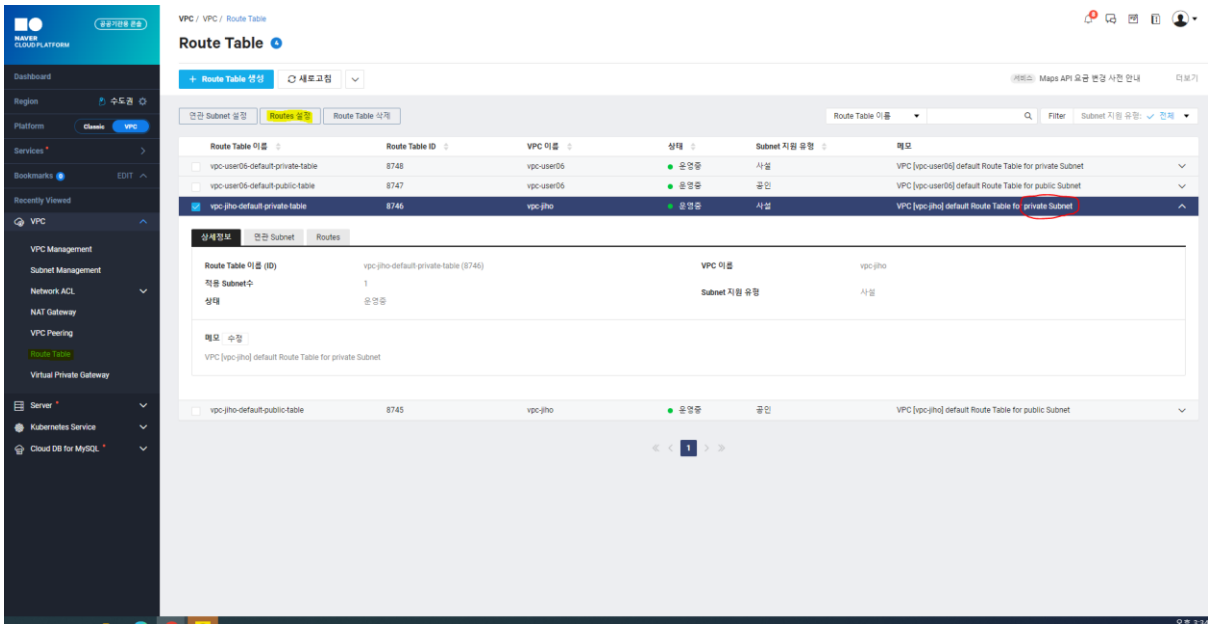


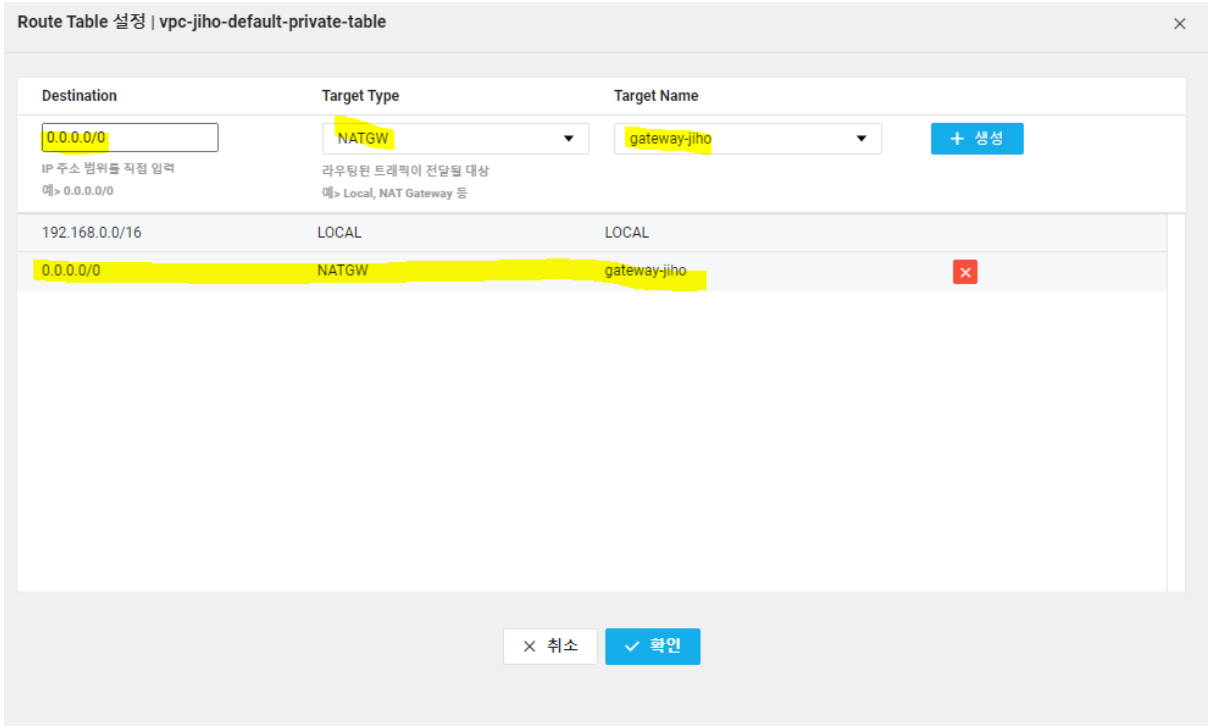

Subnet 삭제
Subnet 이름
Filter
Zone: 전체 Internet Gateway 적용 여부: 전체 용도:
Subnet 이름 Subnet ID 상태 VPC 이름 IP 주소 범위 Zone Internet Gateway 전용 여부 용도
user07-pub 9792 운영중 vpc-user07 192.168.100.0/24 KR-1 Y (Public) 일반
user07-db 9791 운영중 vpc-user07 192.168.101.0/24 KR-1 N (Private) 일반
user06-db 9790 운영중 vpc-user06 192.168.102.0/24 KR-1 N (Private) 일반
user06-private 9789 운영중 vpc-user06 192.168.101.0/24 KR-1 N (Private) 일반
user06-public 9788 운영중 vpc-user06 192.168.100.0/24 KR-1 Y (Public) 일반
jiho-public-subnet 9772 운영중 vpc-jiho 192.168.100.0/24 KR-1 Y (Public) 일반
jiho-private-subnet 9771 운영중 vpc-jiho 192.168.101.0/24 KR-1 N (Private) 일반
1

3. NAT Gateway 생성



4. Route Table 설정





5. ACG 생성

- 2 개의 ACG 를 생성해준다.