클라우드기초

세션클러스터링, SSH 포트 포워딩

최종석(jschoi@ssu.ac.kr)

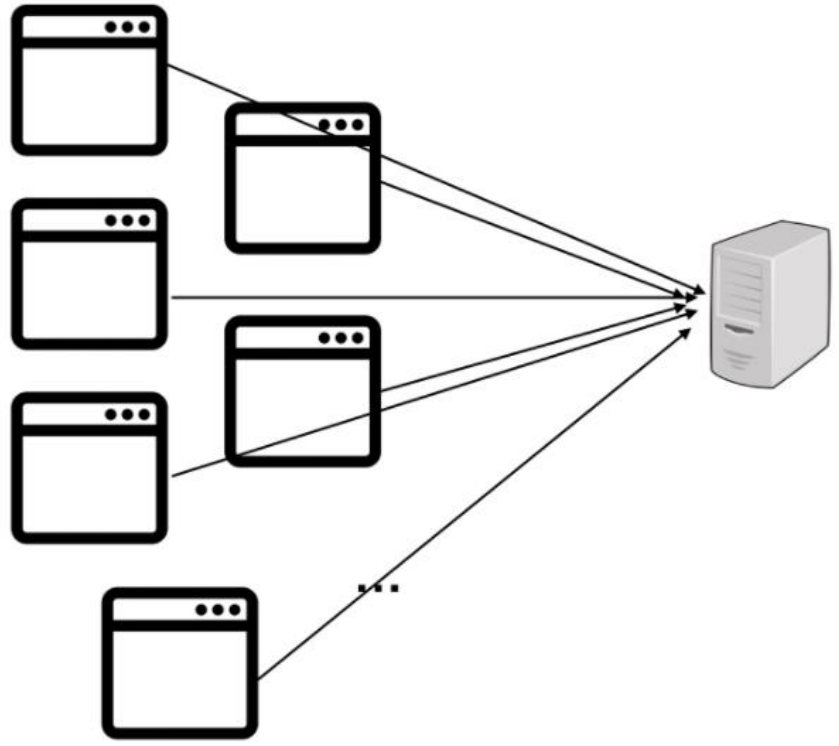
세션 클러스터링

00 Session Clustering

» 탄생 배경

대용량 트래픽을 장애 없이 처리하기 위해 여러 대의 서버에 적절히 트래픽을 분배

→ 로드밸런싱 수직적(물리적인 리소스가 늘려서)으로 분배하거나 수평적(컨테이너를 늘려서)으로 분배하거나



원래는 물리적인 서버가 다아아 처리

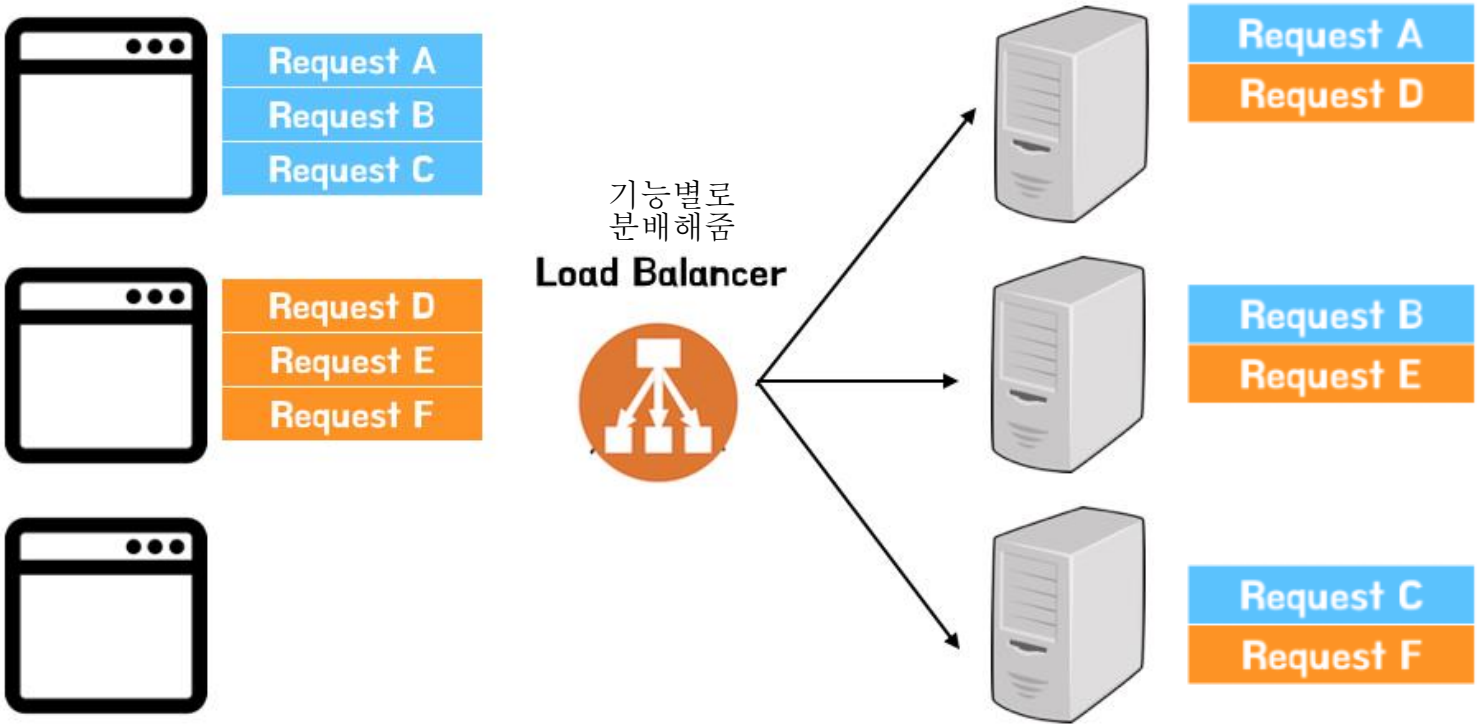
00 Session Clustering

» 탄생 배경

가상화가 없을 때는 물리적인 여러대의 서버로 늘려
ㅂ분배하여 로드 밸런싱

대용량 트래픽을 장애 없이 처리하기 위해 여러 대의 서버에 적절히 트래픽을 분배

→ 로드밸런싱



00 Session Clustering

» 탄생 배경

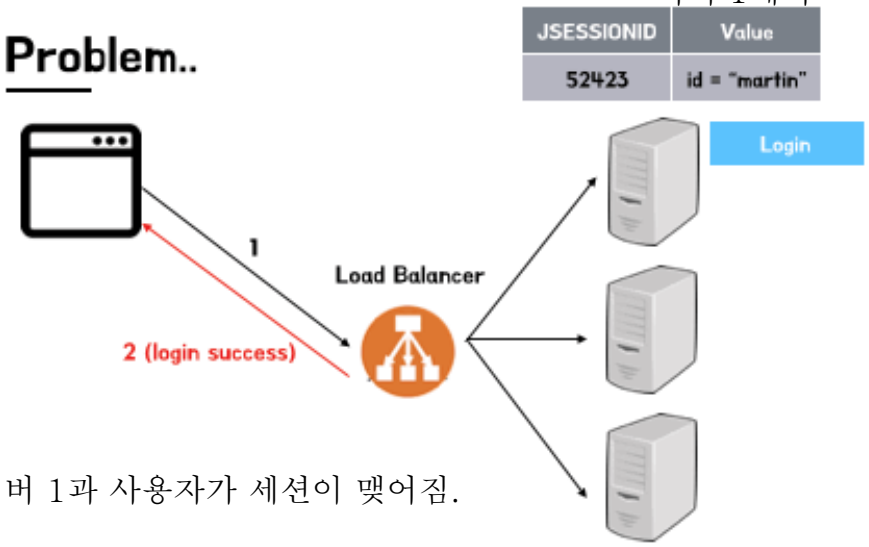
세션에서 문제가 생김.

저장의 문제

대용량 트래픽을 장애 없이 처리하기 위해 여러 대의 서버에 적절히 트래픽을 분배

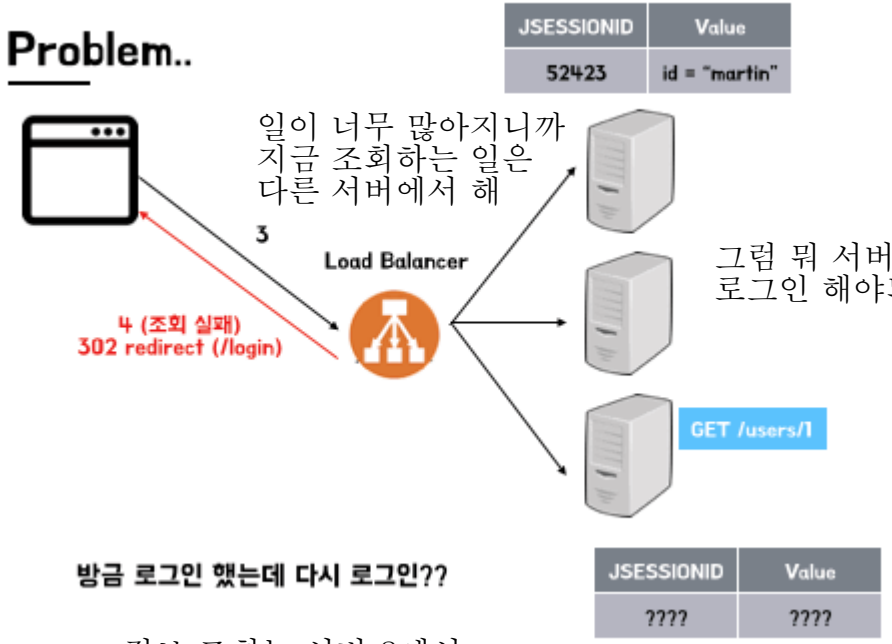
→ 로드밸런싱

로그인된 정보는
서버 1에서



서버 1과 사용자가 세션이 맺어짐.
!!

그럼 세션 정보를 다 퍼뜨려서 사용하면 되잖아.
그럼 언제 해제됐는지(나갔는지) 모른다.
세션 정보가 무수히 쌓이고 체크하는 것도 힘들다.
용량 문제도 있다. 모든 문제가 낭비가 심해진다.



정보 조회는 서버 3에서
하지만 로그인 되지 않았는데
서버3에서는
다시 요구하네

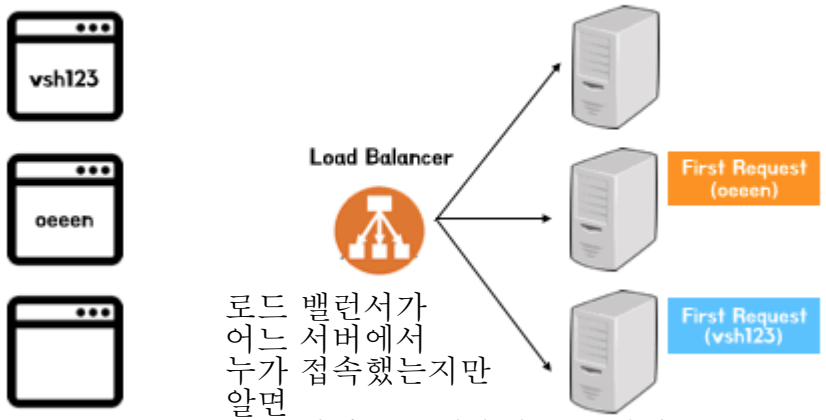
00 Session Clustering

» 탄생 배경

대용량 트래픽을 장애 없이 처리하기 위해 여러 대의 서버에 적절히 트래픽을 분배

→ 로드밸런싱 : Sticky Session

Sticky Session 처음 최초의 서버에 계속 접속하게끔 처리하면 됨



로드 밸런서가 어느 서버에서 누가 접속했는지만 알면 해당 서버로만 사용하게금 지정

또다른 문제점

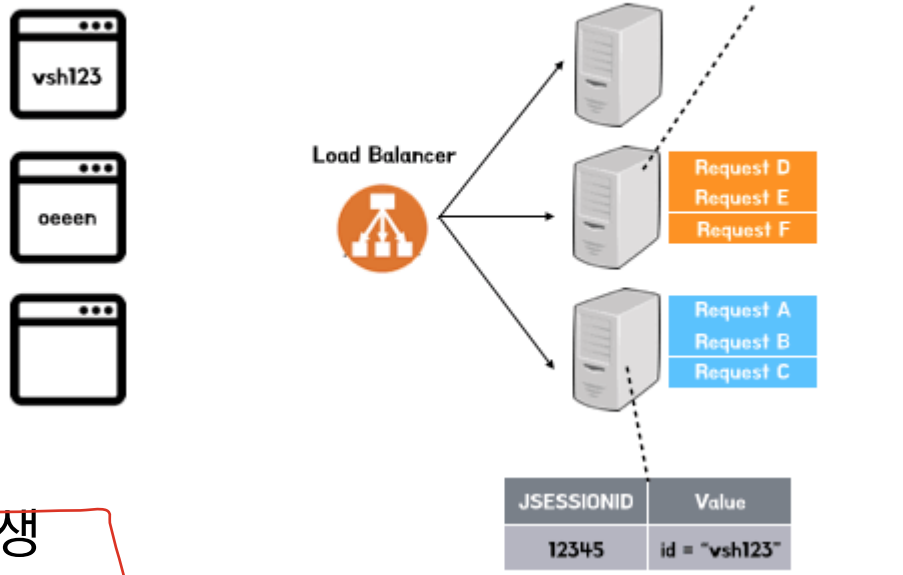
로드밸런싱이 잘 동작하지 않을 시 문제가 발생

특정 서버에 많은 과부하 로드 분재 기준 정책이 편파적일때

특정 서버 Fail시 해당 서버에 붙어 있는 세션이 소실

다 날라감 붙어 있는 세션들 다 날라가

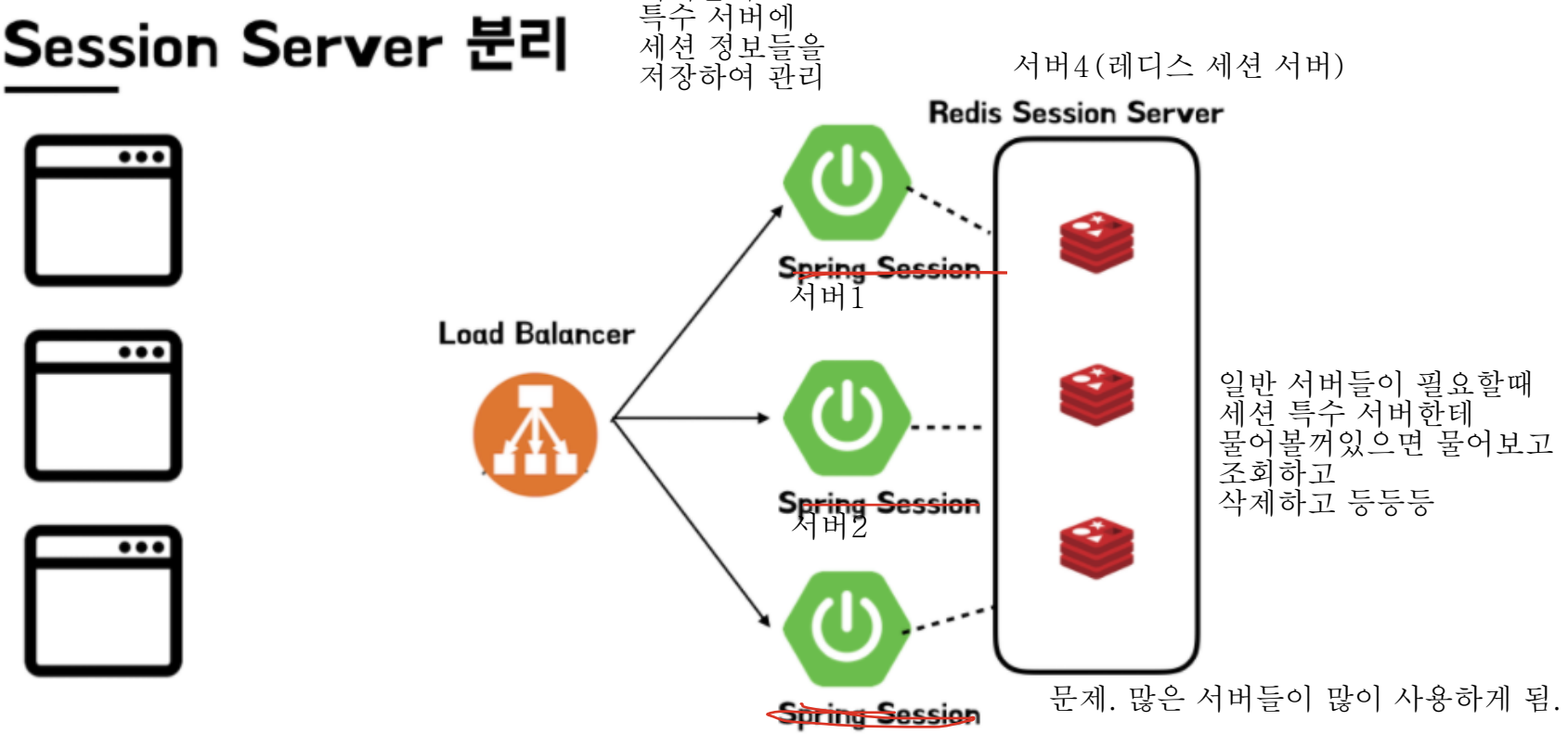
Sticky Session



00 Session Clustering

» 탄생 배경

세션 서버라는 것을 만들어서 해결



00 Session Clustering

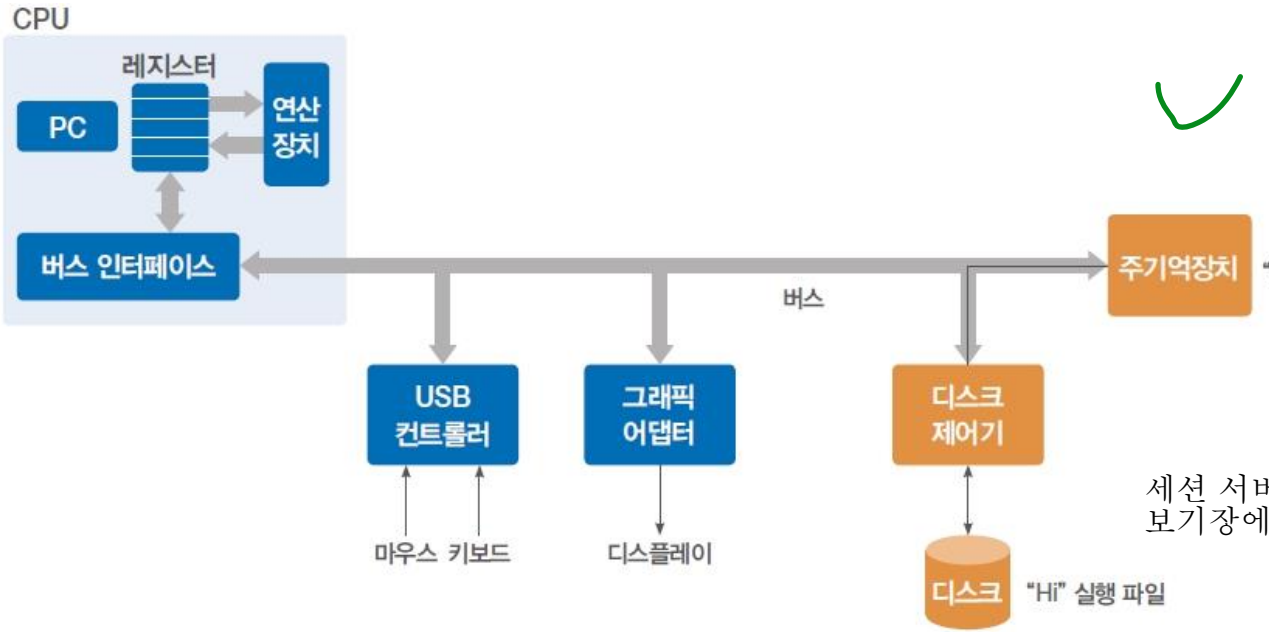
» 컴퓨터(서버)에서 프로그램 로딩이 이루어지는 과정

① C, 자바(Java), 파이썬(Python) 등의 고급 프로그래밍 언어로 프로그램 소스코드를 작성한다.

② 컴파일러와 어셈블러 등을 이용해 고급 프로그래밍 언어로 작성된 소스 프로그램을 기계어 프로그램으로 번역한다.

③ 사용자의 프로그램 실행 명령을 통해 보조기억장치에 저장된 기계어 프로그램을 주기억장치로 로드한다.

④ 주기억장치로 로딩된 프로그램의 명령어 하나하나가 폰 노이만 구조에 따라 수행된다.



보기장에서
주기장으로
반대로도
이동하는 동작이

컴퓨터에서 제일 오래
걸리는 동작이다.
이구간을 해결하는 것이

세션 서버가 원하는 빠르기
를 해결할 부분인것이다.

세션 서버는 그래서 세션 정보를
보기장에다가 저장하지 않음

» 기억장치(저장장치)

-기억장치의 분류 : 레지스터, 캐시메모리, 주기억장치, 보조기억장치

-기억장치를 계층으로 나눈 이유는 전체 비용을 고려했을 때 용량과 접근 속도를 최적화 위해

많은 정보를 빠르게 조회하고
조달해야하는
세션 서버는
보기장을 사용하지 않고
더 빠른 기억 장치들만을 사용한다.

아마 대부분 주기억장치만을 사용할 것이다.
세션 정보는 그렇게 큰 데이터가 아니기에.

- 접근 속도 증가
- 저장 비용 상승

- 기억 용량 감소

세션 서버는 주기억 장치를 사용한다!

데이터 접근 속도가 엄청 빨라지는 이유다!

» 기억장치(저장장치)

– 주기억장치(메인메모리, Main Memory)

- **주기억장치 :** 외부에서 들여온 데이터를 보관하는 공간으로, 작업에 필요한 프로그램과 데이터를 저장하는 장소(이하 메인메모리).

00 Session Clustering

» Redis

- 오픈 소스 기반의 비관계형 데이터베이스 관리 시스템
- 캐시, 세션 클러스터링 등의 용도로 사용되는 인 메모리 데이터 스토어

레디스라는 오픈 소스를 사용해서
메인 메모리 영역의 저장소를 사용하게끔 한다.

간단하게 저장하고 전달하는 체계 일때.

세션 처리할 때 유용함.
세션 클러스터링(세션 군집화(모아놓기))하기 좋음

세션뿐만아니라
빠르게
접근하여 젯공해야하는
서버 같은 경우
인메모리 데이터 저장
기능을 많이 사용한다.

아예 하드 저장소를
사용하지 않는다.

온 디스크 DB
캐시
메인 데이터

인메모리 DB
메인 데이터
백업

In Memory DB

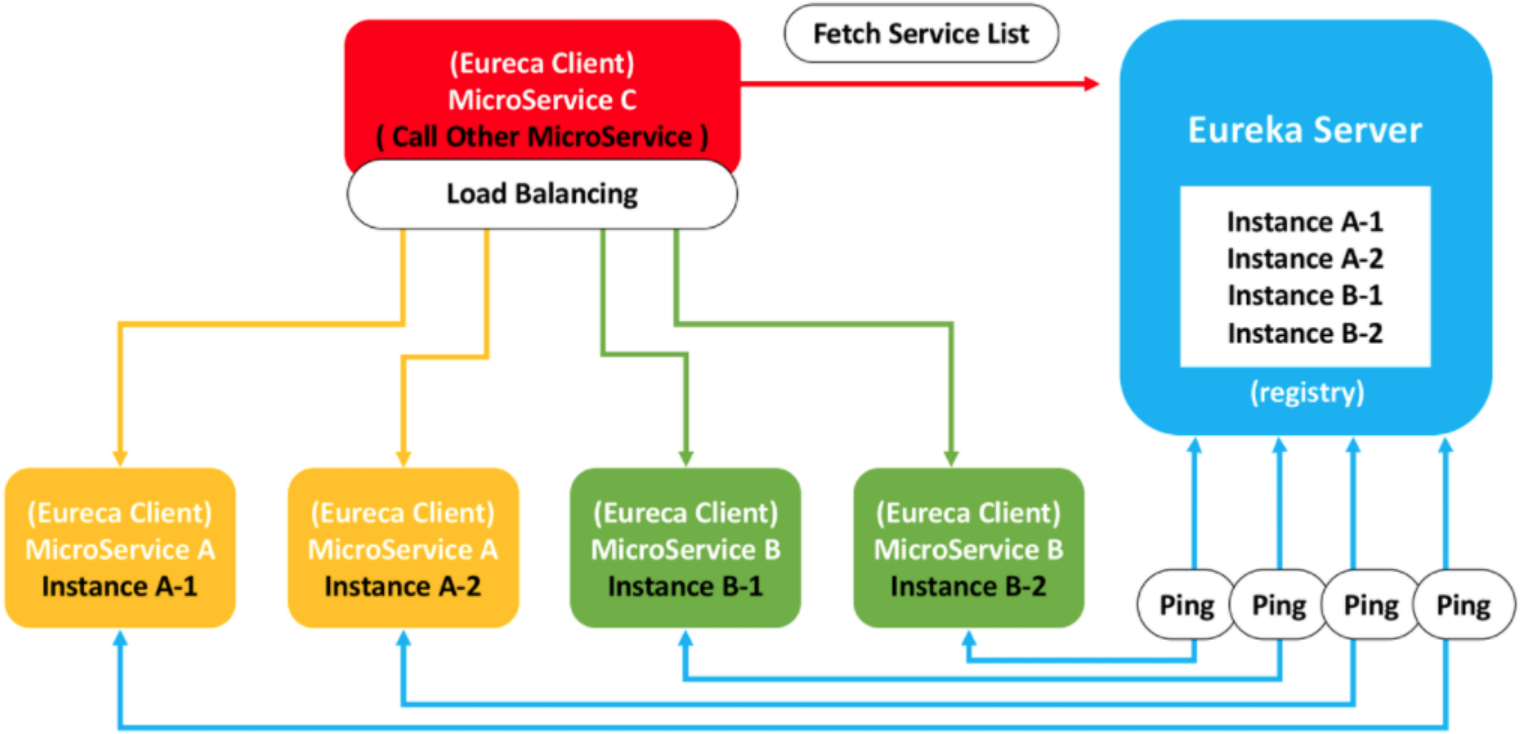
클라우드에서도 많이 사용

00 Session Clustering

» Eureka 세션 클러스터링 오픈 소스 로드밸런싱 지원 오픈소스

넷플릭스가 개발.
MSA형태로 잘만든 기업이
오픈소스로 개발함.
유레카 말고도 많다!

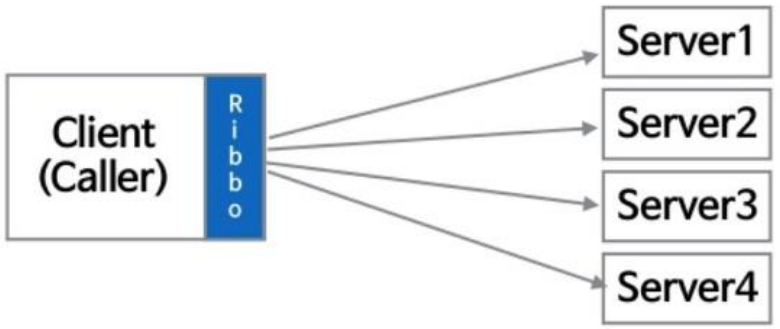
- 마이크로서비스들의 정보를 레지스트리에 등록할 수 있도록 하고, 마이크로서비스의 동적인 탐색과 로드밸런싱을 제공



Session Clustering

» Ribbon

- Client에 탑재된 로드밸런서이며, 서버사이드에서 필요했던 H/W의 부담이 사라지며, 서버 목록의 변경이 쉬워지고, 로드밸런싱 방식도 다양하게 설정할 수 있음



00 이번주 용어

Canary, Blue-Green, Rolling Update????

비슷하게
카나리 업데이트와 블루 그린 업데이트 기법도 있다!
K3s도 뭔지 같이 알아보자

로드 밸런싱되고 세션 클러스터링도 잘되서

각 서버에 잘 분배되고 지정된 상태에서

K3s ?????

업데이트가 된다면 어떻게 될까

한 서버 내부에서 복제를 한다. 복제한 것을 사용자들이 사용하게끔하고
그동안 원본을 업데이트를 하고
업데이트가 완료되면
사용자들을 다시 업데이트된 원본을 사용하게금하여
안끊기고 사용할 수 있게끔 한다.

지속적인 연결을 시킨다.

컨테이너에대한 복제가 쉬워서 가능한 일이다.

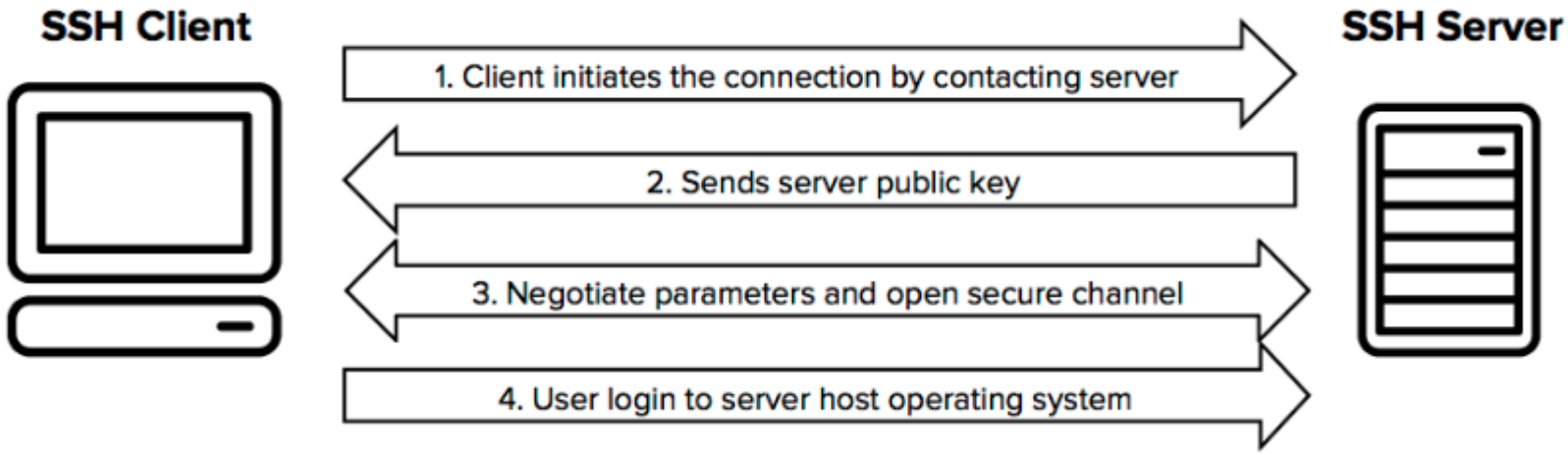
업데이트나 컨테이너 오류를 생겨도
끊기지 않게 업데이트하는 깅법이
롤링 업데이트이다.

SSH

01 SSH

» SSH란

- Secure Shell 의 약어로 원격지 호스트에 접속하기 위해 사용되는 프로토콜
- 암호화 기법이 사용된 통신 프로토콜로 패킷이 노출되더라도 암호화된 내용을 통해 정보를 보호하기 위해 사용
- CLI에서 사용되는 것으로 기본 포트는 22번



01 SSH

» SSH란

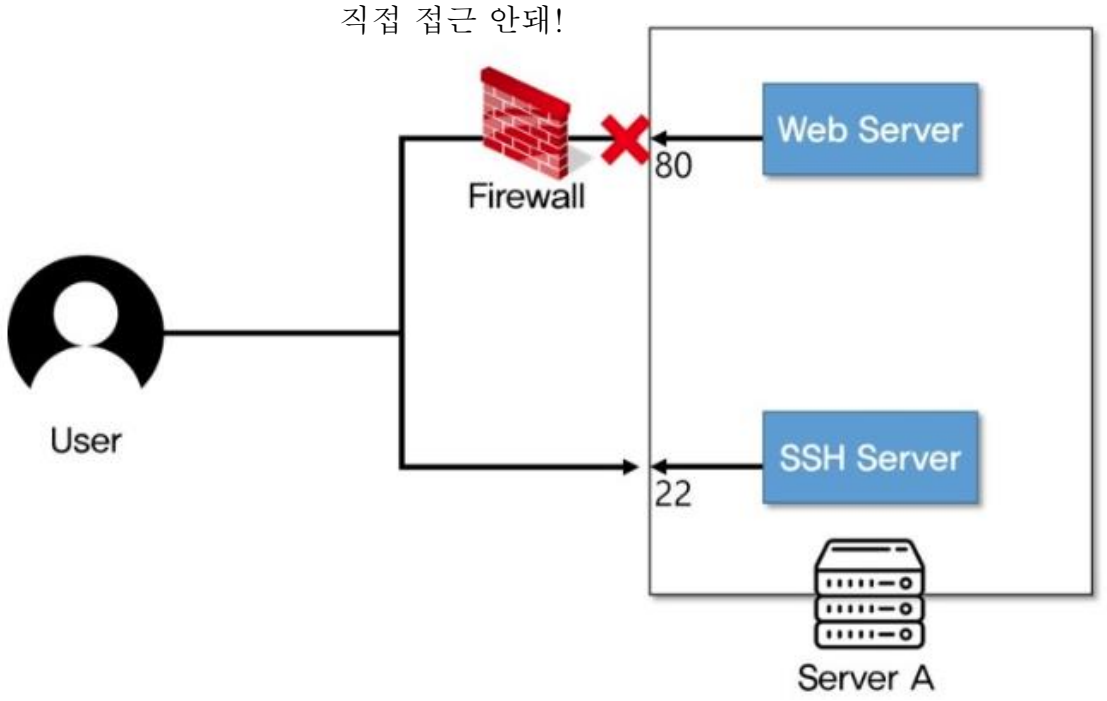
- 주요기능
  - 보안 접속을 통한 rsh, rcp, rlogin, rexec, telnet, ftp 등을 제공
  - IP spoofing 을 방지하기 위한 기능을 제공
  - TCP/IP 패킷 포워딩을 제공

» SSH 포트 포워딩(SSH 터널링)이란

서버 기준 들어오는 패킷 막기

- 서버 A에서 80포트로 바인딩 된 서비스에 접속할 때 방화벽의 인바운드 정책으로 인해 접속이 불가능(방화벽이 80포트 외의 포트를 개방하지 않았기 때문)

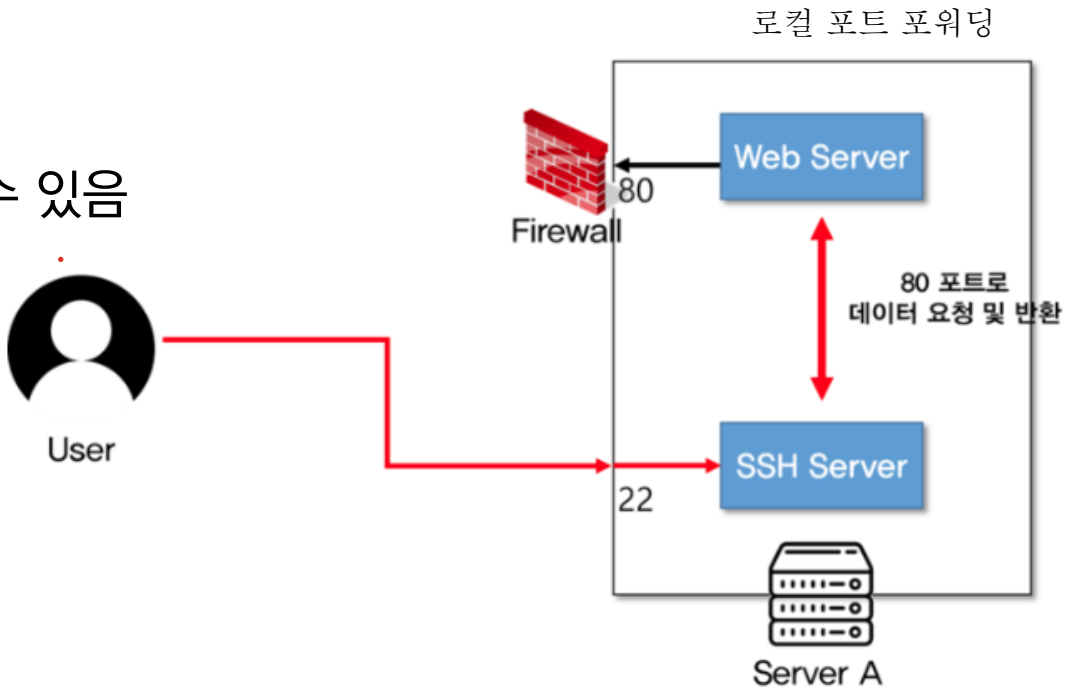
웹 관련 포트빼고 다 폐쇄



» SSH 포트 포워딩(SSH 터널링)이란

- 웹서버와 동일한 서버 내(혹은 동일한 NAT내부) 의 SSH 서버에 접속하고, 이를 통해 웹서버로 접속
- SSH연결 수립 후 외부로부터 데이터를 보호할 수 있는 일종의 통로를 터널

  → 터널링
- 일종의 프록시서버로 볼 수 있음

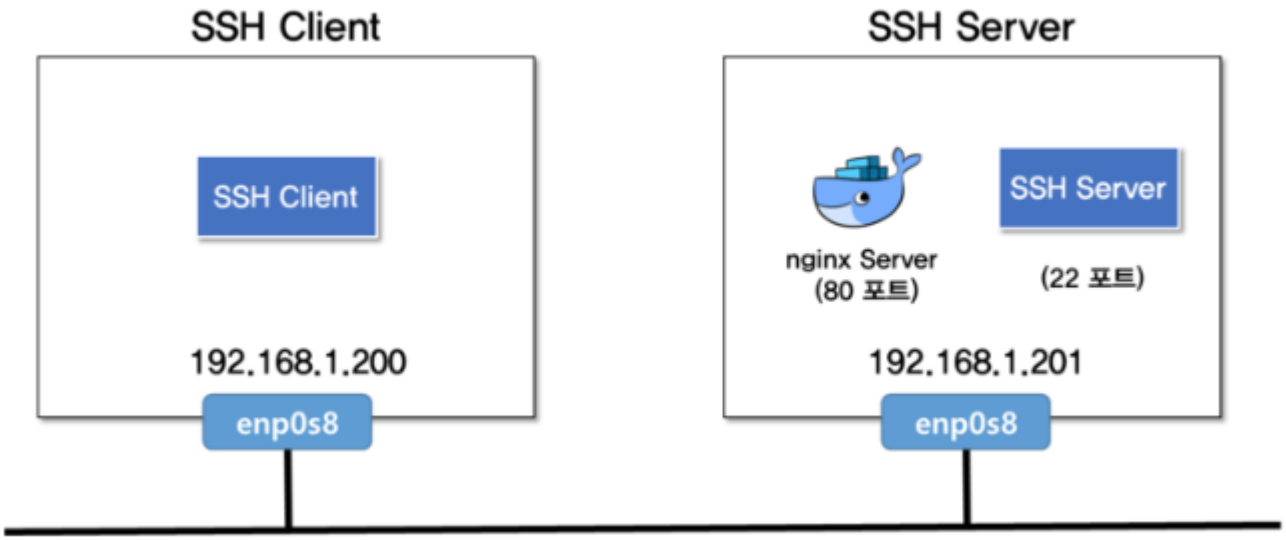


» SSH 포트 포워딩(SSH 터널링) – Local 포트 포워딩

- SSH 포트 포워딩에는 Local, Remote 이 있음
- Local 포트 포워딩은 SSH클라이언트가 SSH서버로 연결을 수립
- Remote 포트 포워딩은 SSH서버가 SSH클라이언트로 연결을 수립

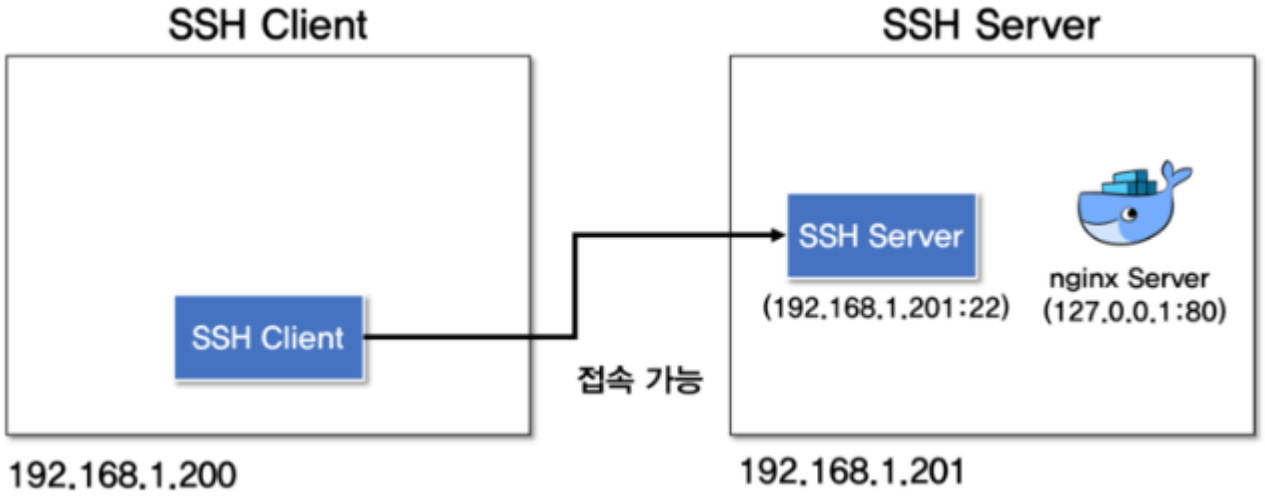
서버가 ㅏ 클라이언트로.

반대 방향



» SSH 포트 포워딩(SSH 터널링) – Local 포트 포워딩

- Local 포트 포워딩



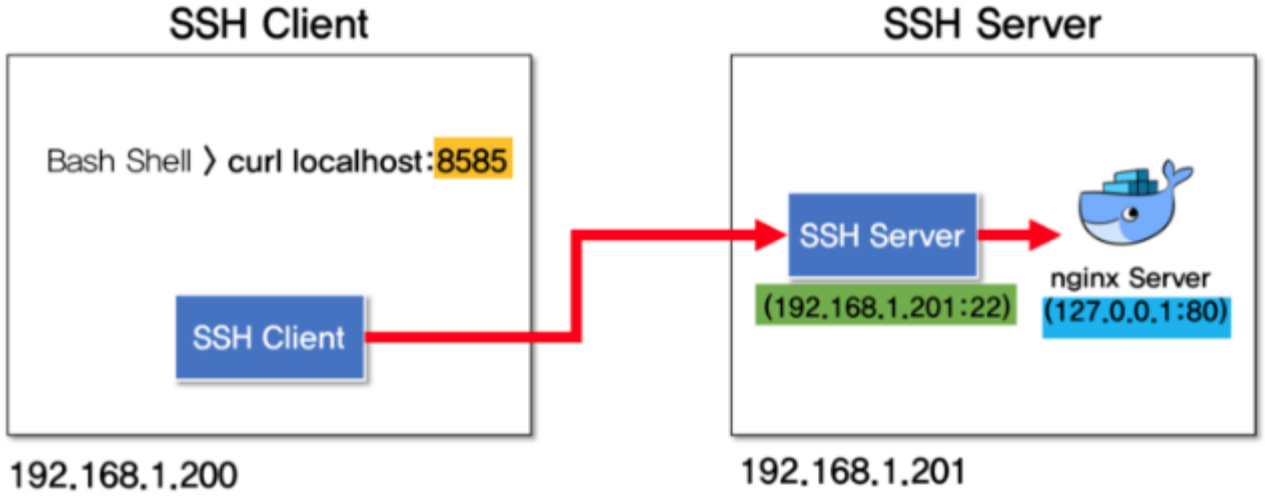
» SSH 포트 포워딩(SSH 터널링) - Local 포트 포워딩

- Local 포트 포워딩
  - ssh -L 8585:127.0.0.1:80 192.168.1.201

    8585 = 로컬에서 사용할 포트

    127.0.0.1:80 = 최종적으로 접근할 곳

    192.168.1.201 = SSH Server 주소

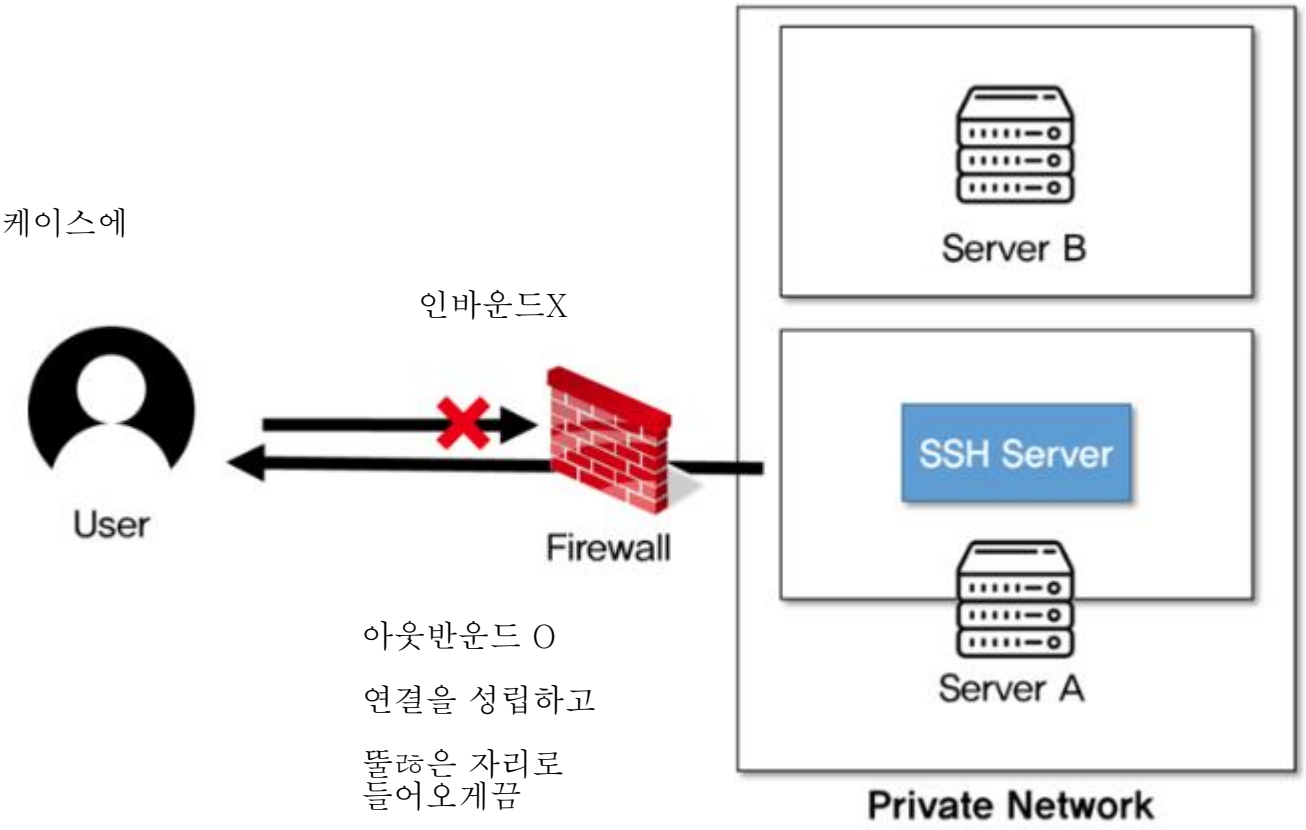


SSH 터널링 명령어 : ssh -L 8585:127.0.0.1:80 192.168.1.201

» SSH 포트 포워딩(SSH 터널링) – Remote 포트 포워딩

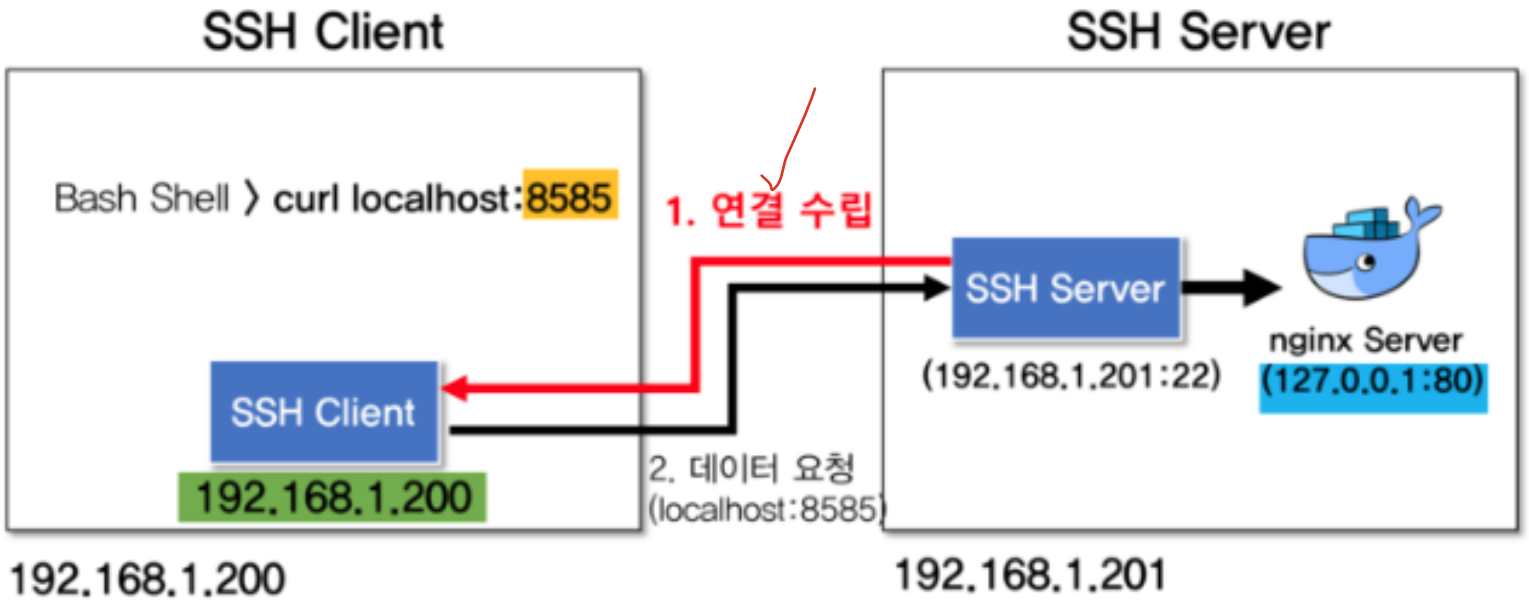
- Remote 포트 포워딩
  - 방화벽에서 외부로 나가는 아웃바운드 패킷은 허용되지만 인바운드 패킷은 허용되지 않을 때 사용

유저가 아예 못들어가는 케이스에



» SSH 포트 포워딩(SSH 터널링) – Remote 포트 포워딩

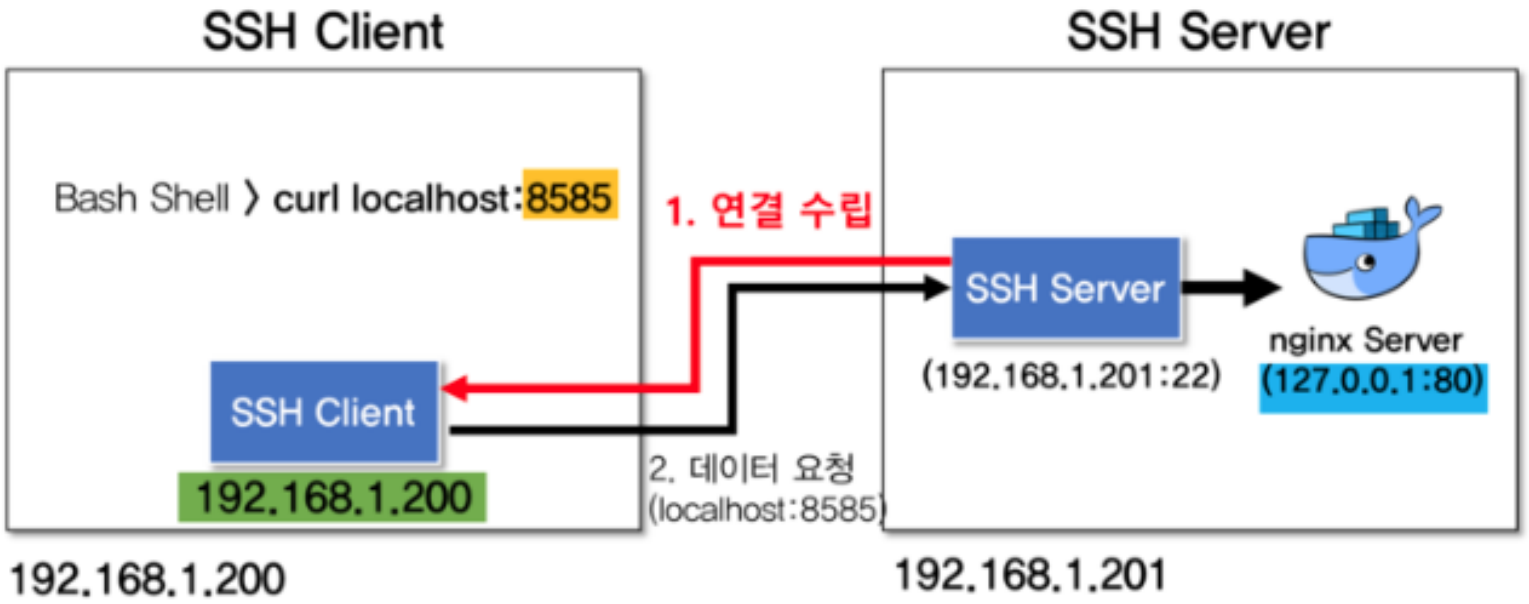
- Remote 포트 포워딩



SSH 터널링 명령어 : **ssh –R 8585:127.0.0.1:80 192.168.1.200**

» SSH 포트 포워딩(SSH 터널링) – Remote 포트 포워딩

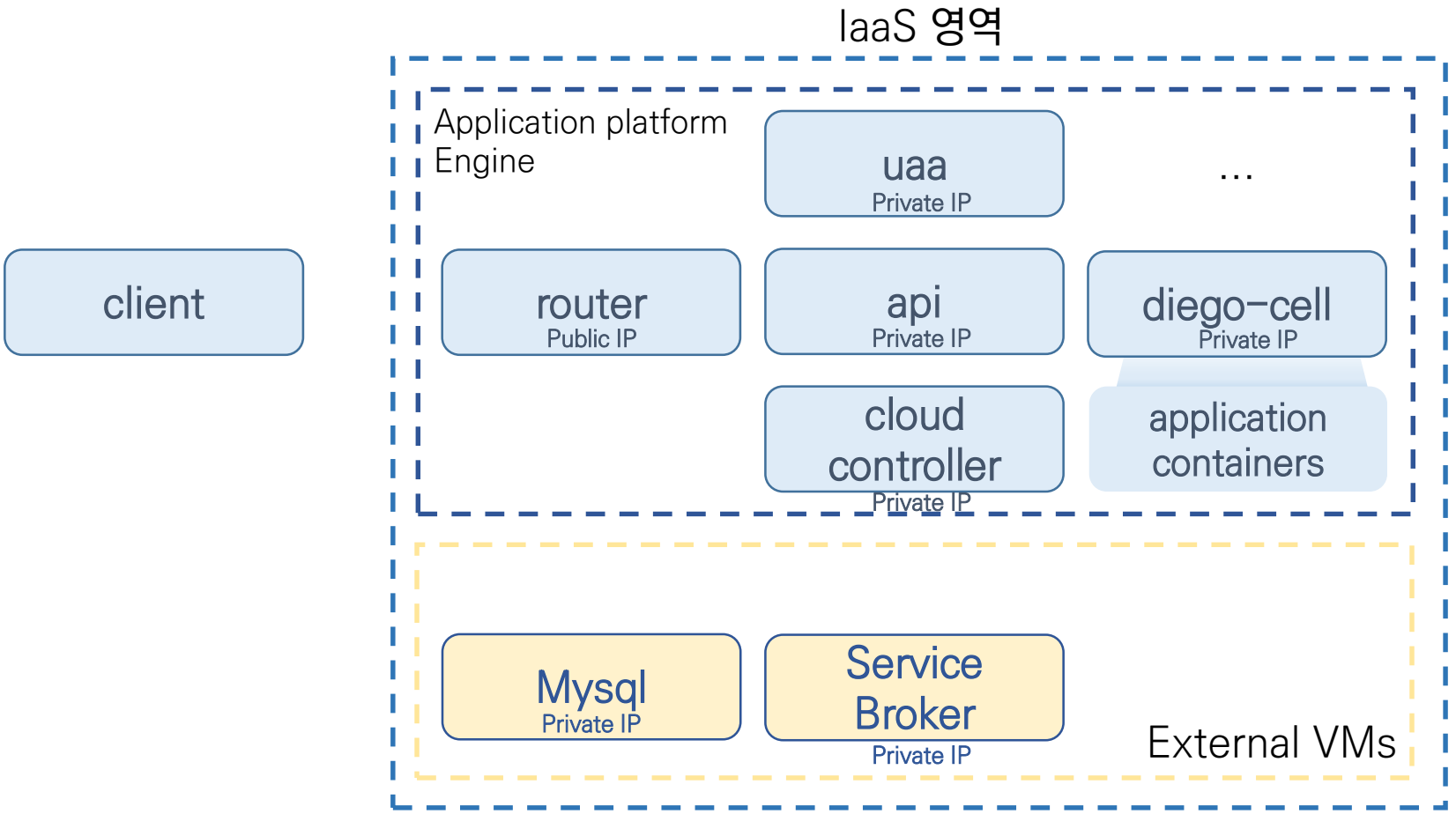
- Remote 포트 포워딩



SSH 터널링 명령어 : **ssh –R 8585:127.0.0.1:80 192.168.1.200**

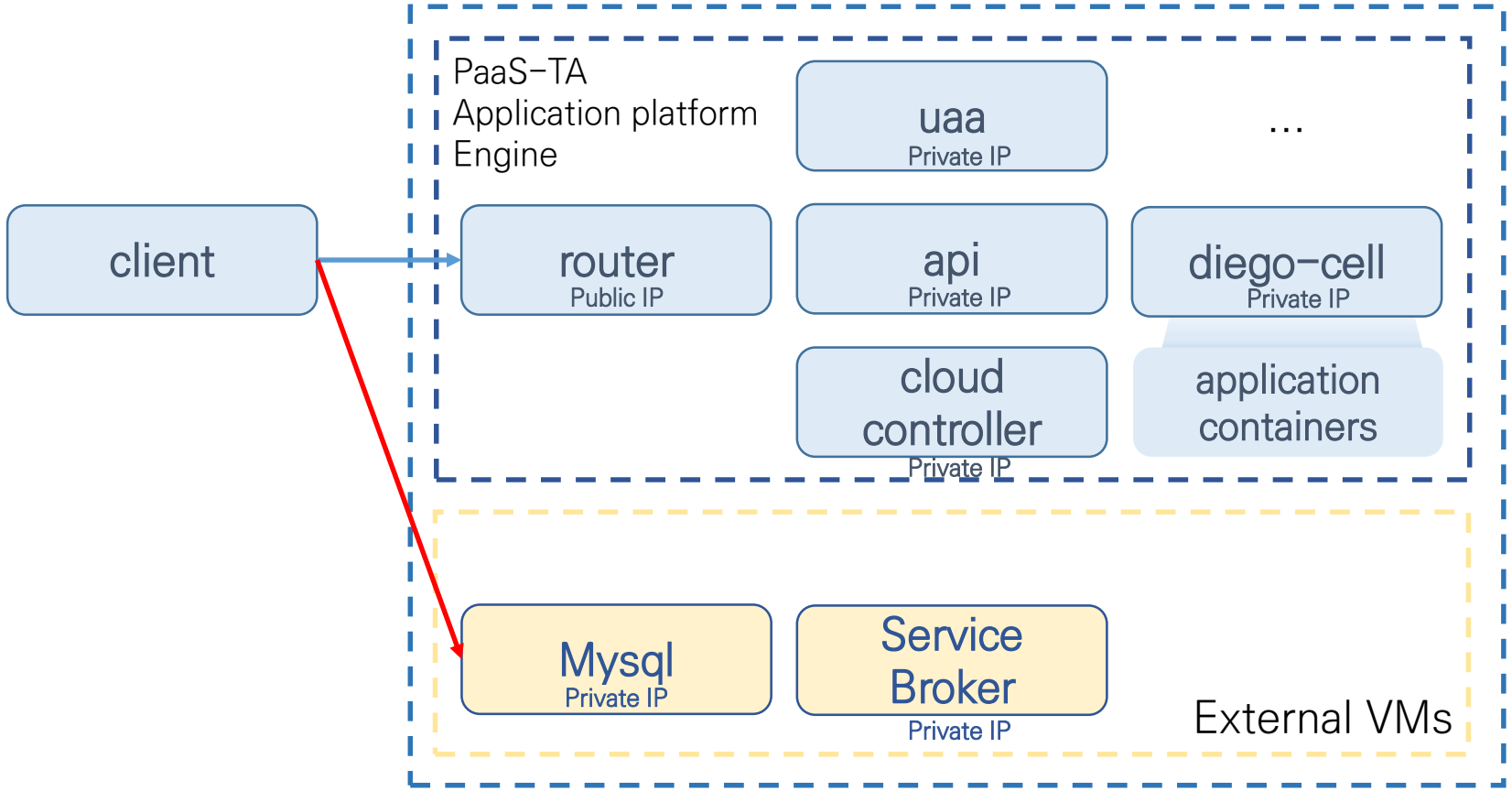
» DB서비스에 Local 포트 포워딩으로 접속하기

- Cloud Foundry 서비스 구조



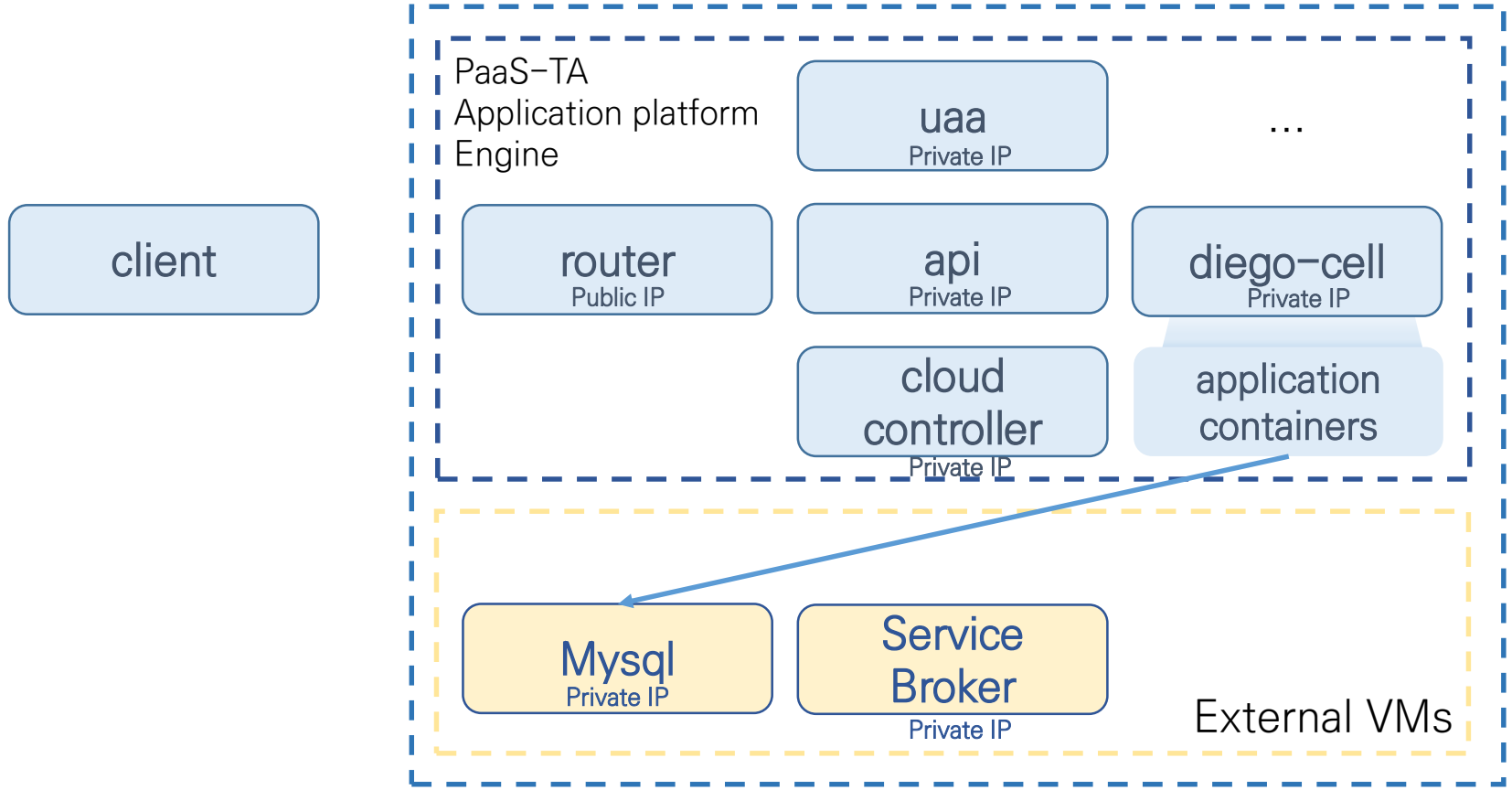
» DB서비스에 Local 포트 포워딩으로 접속하기

- Cloud Foundry 서비스 구조



» DB서비스에 Local 포트 포워딩으로 접속하기

- Cloud Foundry 서비스 구조



» DB서비스에 Local 포트 포워딩으로 접속하기

- Cloud Foundry 서비스 구조