**Domain 연동하기**

**준비하기**

- gabia사이트에 회원 가입 후 550원(현금)짜리 계정 만든다.

(<https://www.gabia.com/>)

또는

- “내도메인.한국” 에서 무료로 도메인 계정을 만든다.

**gabia사이트를 이용하는 경우 참고**

|  |
| --- |
| 가비아 ACM(Authority Certification Management) 인증서가 늦게 떨어질 수 있기 때문에 실습하기 전에 개인 도매인 생성부터 먼저 해야 한다.– HTTPS 연결제공  교육장에서 하나의 ip를 타고 나가기 때문에 발급에 시간이 많이 소요될 수 있다.  KOSTA 강의장에서는 3일동안 한 ip당 3개만 가입 가능하므로 주의 – 각자 집에서 생성해오기)  1. 가비아 회원가입  2. 500원 짜리 개인 도메인 만들기(shop 이나 store로 끝나는 도메인을 준다.)  3. My가비아 클릭 - 도메인 생성시 1년 짜리로 해야 500원(부가세포함 550원)이 된다.  4. 네임 서버 신청은 가비아 네임 서버 사용으로 선택하기  5. 안전 잠금 서비스 신청하면 확인 이메일이나 메시지를 보내준다.  6. 동시 신청 가능 서비스에서 웹 호스팅 신청하면 비싸니까 신청하지 않기  7. 마이 가비아>> 서비스 관리에 가면 만든 도메인이 보인다.  8. 카드 결재가 최하가 1000원이므로 무통장 입금으로만 가능하다. |

**실습 – “내도메인한국” 으로 해본다.**

- 회원가입을 진행한다.

- 아래화면처럼 원하는 도메인을 검색해서 등록하기를 한다.

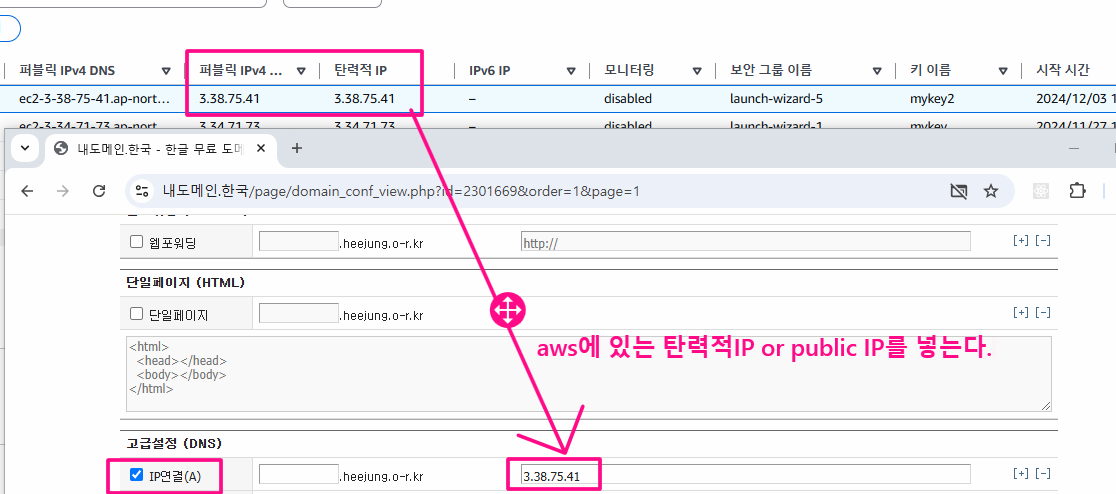


**메뉴 > 도메인 관리 > 수정 클릭**



AWS에 접속 해서 도메인으로 등록 하려는 인스턴스의 IP를 알아본다.

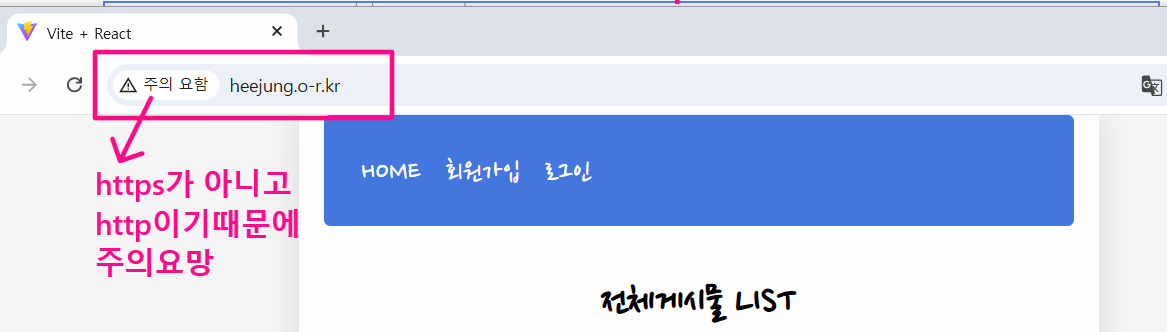
우리는 React를 start 해놓은 인스턴스의 IP를 도메인과 연결한다.



IP입력 후 보안코드 입력 한 후 수정하기를 클릭한다.

브라우져에서 도메인으로 요청해본다.

아래와 같이 도메인으로 잘 연결 된 것을 확인한다.



|  |
| --- |
| 브라우저에서 "주의 요망" 또는 "보안 경고" 메시지가 나타나는 이유는 주로 **HTTP**와 **HTTPS** 간의 차이 때문이다. HTTPS는 보안이 강화된 HTTP 프로토콜로, 데이터를 암호화하여 전송한다. 반면 HTTP는 암호화되지 않아서 보안이 취약하다. 따라서 HTTPS를 사용하는 웹사이트는 데이터 전송 시 암호화와 인증을 제공하며, HTTPS 연결을 통해 웹사이트와의 통신을 보호할 수 있다.  **보안 취약점** : HTTP는 데이터를 암호화하지 않기 때문에, 네트워크 상에서 정보가 탈취될 수 있습니다. 예를 들어, 공용 Wi-Fi에서 HTTP를 통해 웹사이트에 접속하면, 악의적인 사용자가 중간에서 데이터를 가로챌 수 있다.  **브라우저 경고** : 웹사이트가 HTTPS를 사용하지 않고 HTTP로 접속할 경우, **대부분의 최신 웹 브라우저(예: Chrome, Firefox)는 사용자의 보안을 위해 경고 메시지를 표시한다. "이 사이트는 안전하지 않음", "연결이 안전하지 않음" 등의 메시지가 나타나다.**  **사이트 인증 부족** : **HTTPS는 SSL/TLS 인증서를 사용하여 웹사이트의 신뢰성을 검증한다.** 인증서가 유효하고 신뢰할 수 있는 기관에서 발급되었음을 확인하는 절차가 포함된다. 반면 HTTP는 인증서가 없기 때문에, 브라우저는 연결의 신뢰성을 보장할 수 없다.  **검색 엔진 최적화(SEO)** : 구글과 같은 검색 엔진은 HTTPS 웹사이트를 더 신뢰하며, HTTP로만 접속할 수 있는 웹사이트는 SEO 순위에서 불이익을 받을 수 있다.  **따라서** , HTTP 연결을 사용할 경우 보안이 약해지므로, 브라우저는 사용자에게 이를 경고하는 메시지를 표시하고, HTTPS로의 전환을 권장한다. |

**https 프로토콜 적용하기**

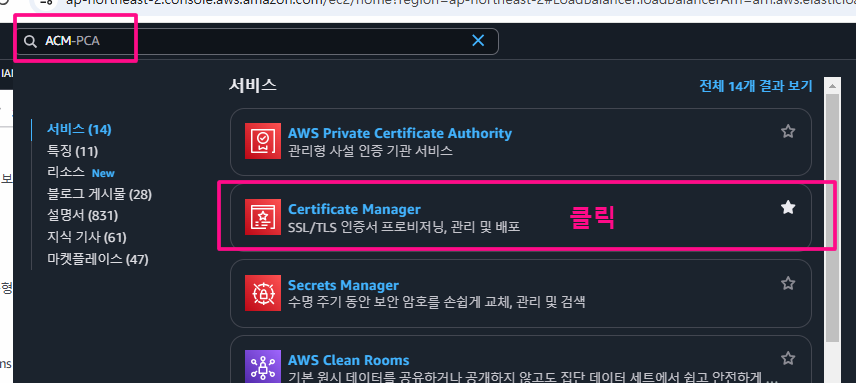
AWS에서 https를 사용하려면 SSL/TLS 인증서를 설정하고, 이를 통해 웹 서버에서 보안 연결을 지원해야 한다. 주요 단계는 AWS에서 인증서를 생성하고, 이를 웹 서버와 연결하여 HTTPS를 활성화한다.

**AWS에서 HTTPS를 설정하는 과정**

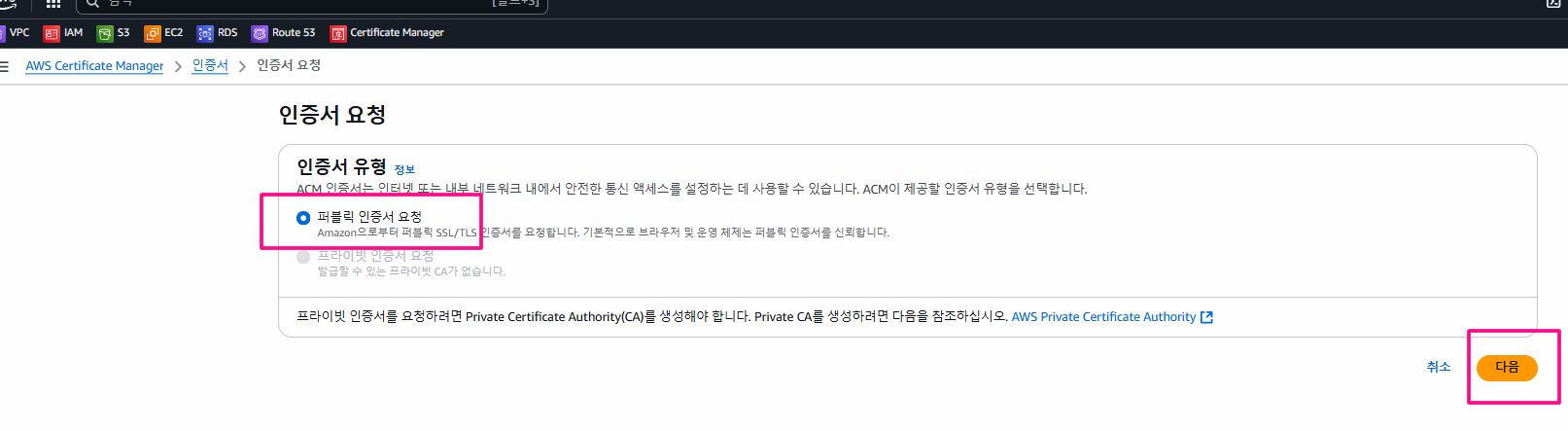
1. ACM(AWS Certificate Manager)에서 SSL/TLS 인증서 요청
2. 타켓그룹 생성
3. ELB(Elastic Load Balancer)생성
4. 보안그룹수정

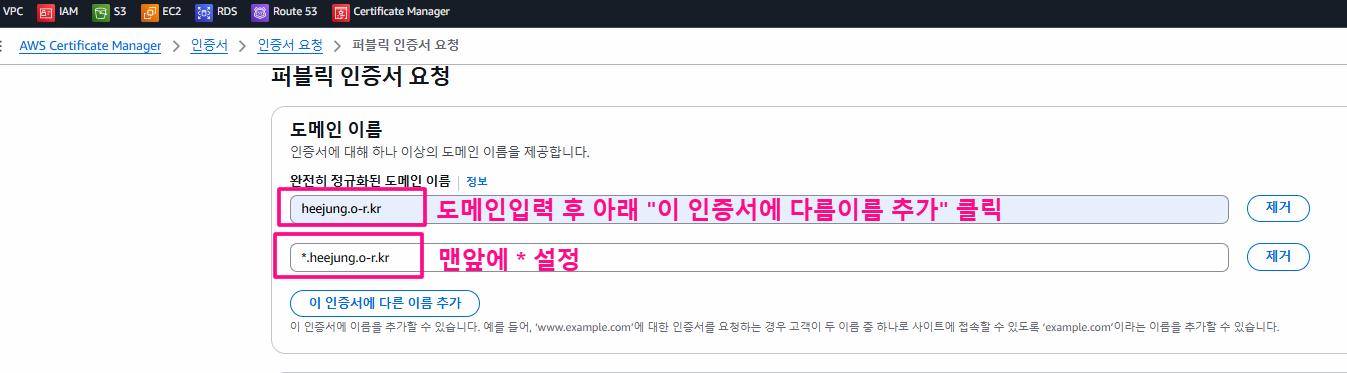
**위 순서대로 진행 해보자.**

Aws 에서 ACM검색

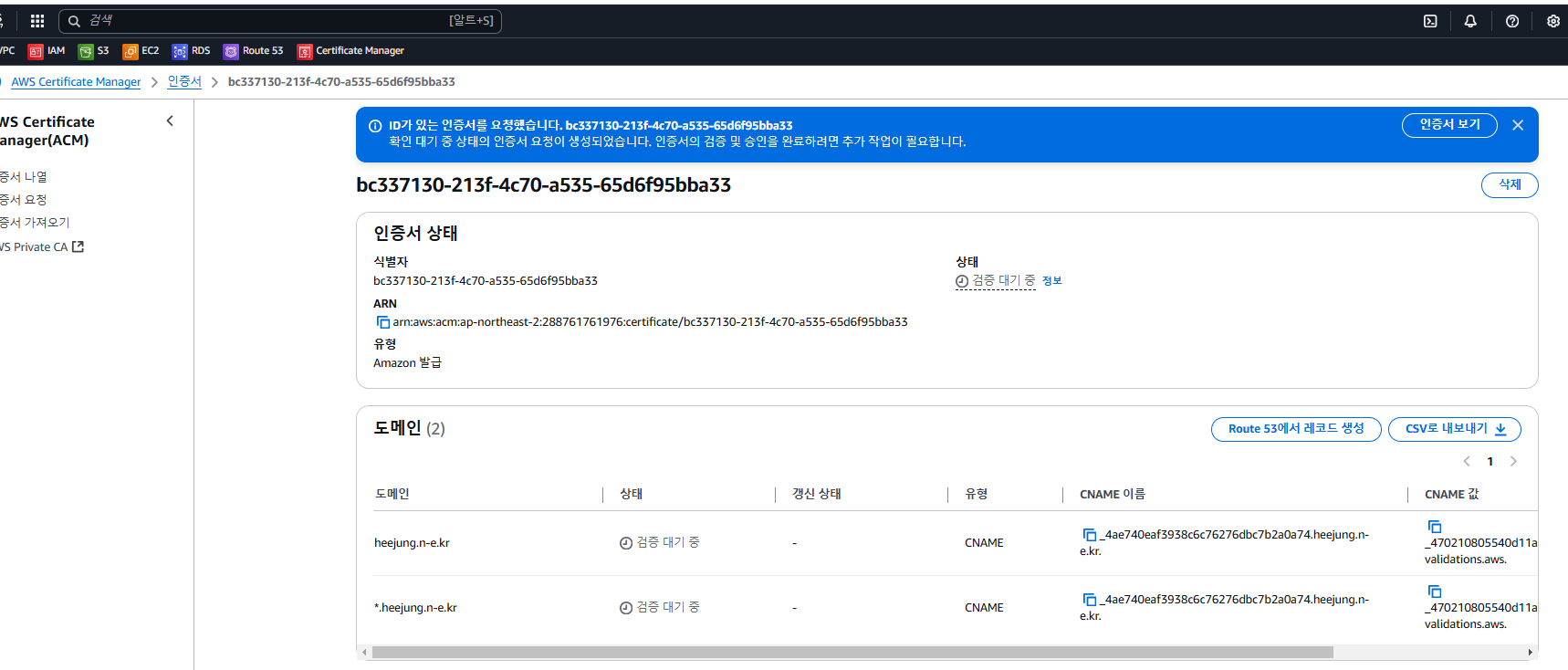


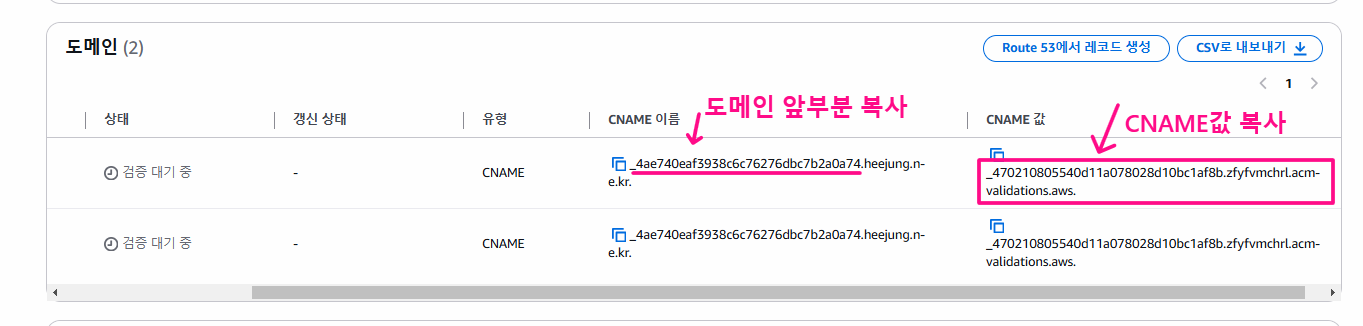






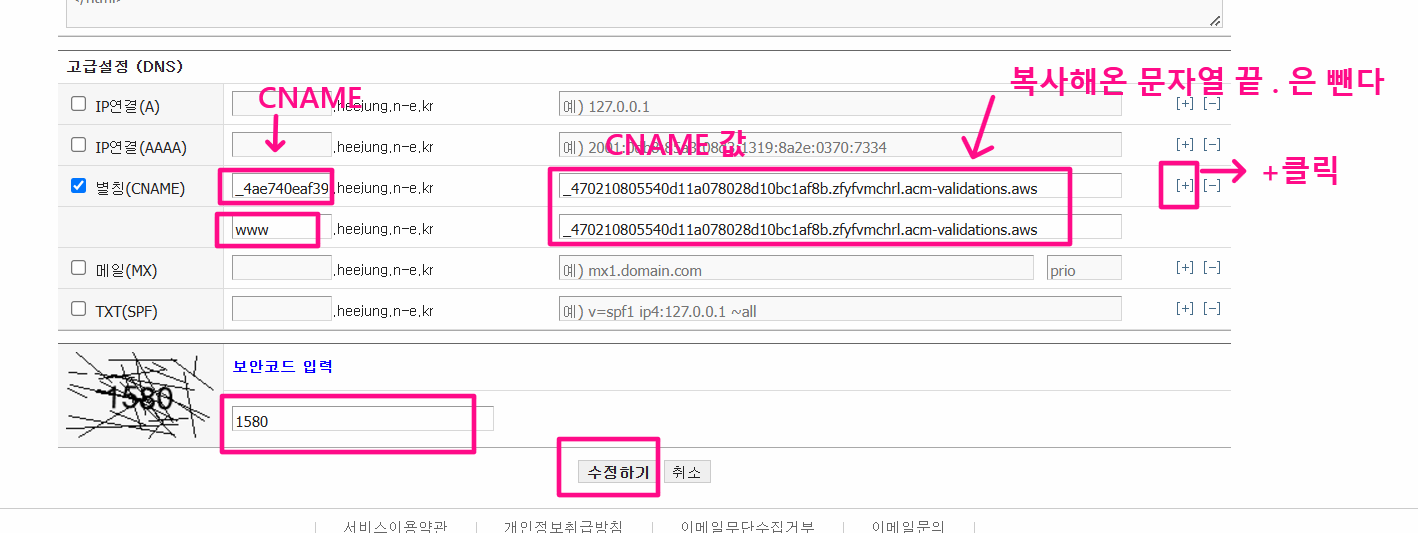




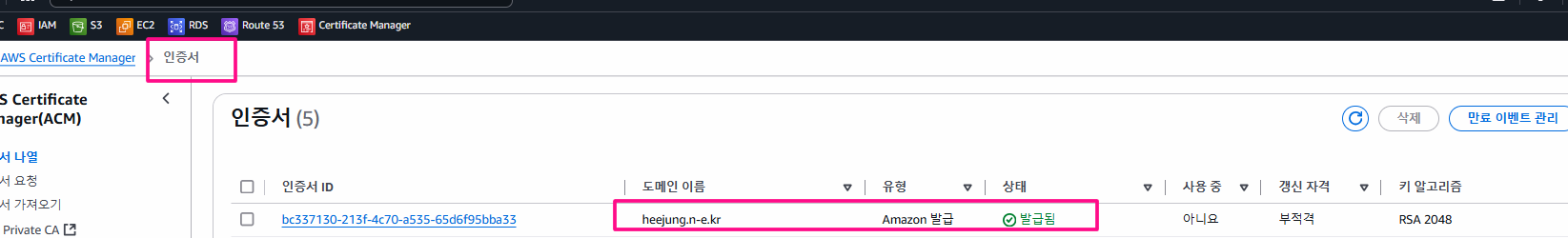


위 CNAME이름과 CNAME값을 “내도메인한국”의 나의 계정에 등록해야 인증서가 발급된다.

기존에 설정해 놓은 IP연결은 해지해도 된다.



**다시 AWS로 돌아가서 잠시 기다리면(5분이내) 인증서가 발급 되었음을 확인한다.**

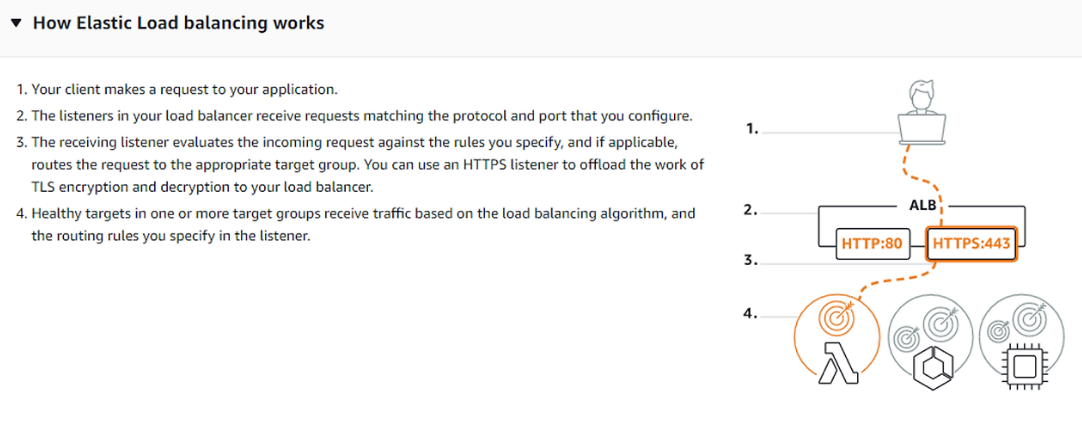


**이제 AWS에서 https를 사용할 때 매우 유용한 ELB를 생성해보자.**

여러 서버에서 HTTPS를 처리하거나, 고가용성과 확장성이 중요한 웹 애플리케이션에서는 ELB를 사용하는 것이 권장되며 ELB는 HTTPS 요청을 처리하고 SSL/TLS 인증서를 관리하는 데 큰 도움이 된다.

**☞ ELB에 대해서**

|  |
| --- |
| AWS ELB(Elastic Load Balancer)는 Amazon Web Services(AWS)에서 제공하는 **로드 밸런싱 서비스**로 여러 서버(인스턴스)에 들어오는 트래픽을 분산하여 서버 간의 부하를 고르게 분배하고, 트래픽 처리 성능을 향상시키며, 애플리케이션의 고가용성 및 확장성을 보장한다.. ELB는 특히 웹 애플리케이션에서 중요한 역할을 하며, **HTTPS**와 같은 보안 트래픽을 처리하는 데 유용하다.. ☞ ELB의 주요 특징:  1. **트래픽 분산**:    * ELB는 여러 EC2 인스턴스(또는 다른 AWS 리소스)로 들어오는 트래픽을 분산시킨다. 이를 통해 서버 하나에 집중되는 과부하를 방지하고, 높은 가용성과 확장성을 유지할 수 있다. 2. **고가용성**:    * 여러 가용 영역(Availability Zones)에서 자동으로 트래픽을 분배하여, 하나의 데이터 센터가 다운되더라도 다른 지역에서 트래픽을 처리할 수 있게 한다. 3. **자동 확장**:    * 트래픽이 급증할 때 EC2 인스턴스를 자동으로 추가하거나 제거하여, 수요에 맞게 자동으로 확장하고 축소할 수 있다.. 4. **다양한 프로토콜 지원**:    * ELB는 HTTP, HTTPS, TCP 등 여러 프로토콜을 지원한다. 따라서 다양한 유형의 트래픽을 처리할 수 있다.  ELB를 사용하여 HTTPS를 설정하는 이유:  1. **SSL 종료 (SSL Termination)**:    * ELB는 **SSL 종료** 기능을 제공한다. 즉, 클라이언트에서 ELB로 오는 HTTPS 요청을 먼저 받아 SSL 암호화를 해제(SSL Termination)하고, 그 후에 내부 트래픽은 **HTTP**로 처리할 수 있다. 이는 내부 시스템에서 HTTPS를 직접 처리할 필요 없이, 트래픽을 효율적으로 관리하는 데 유리하다. 2. **인증서 관리 간소화**:    * SSL/TLS 인증서를 직접 웹 서버(예: Apache, Nginx)에 설치하는 대신, ELB에 인증서를 연결해 관리할 수 있다. AWS Certificate Manager (ACM)을 통해 인증서를 ELB에 쉽게 연결할 수 있다. 3. **고가용성 및 확장성**:    * HTTPS는 보안 연결을 위한 추가적인 처리 능력을 요구하는데, ELB는 이를 자동으로 분산 처리한다. 또한 ELB는 가용성 높은 구조로 여러 인스턴스에 트래픽을 분배하여 높은 트래픽에도 대응할 수 있다. 4. **보안**:    * HTTPS 트래픽을 처리할 때, ELB는 **웹 애플리케이션 방화벽(WAF)**, **DDoS 공격 보호**, **TLS 암호화** 등 추가적인 보안 기능을 통합할 수 있어 보안성을 강화할 수 있다.   다음과 같은 경우에는 ELB를 사용하는 것이 권장한다.   1. **여러 EC2 인스턴스를 사용 중인 경우**:    * 웹 애플리케이션을 여러 EC2 인스턴스에 배포하여 고가용성과 확장성을 요구하는 경우, ELB는 트래픽을 균등하게 분배하여 서버의 부하를 고르게 나눈다. 2. **SSL/TLS 인증서 관리가 필요한 경우**:    * ELB를 사용하면 SSL 인증서를 중앙에서 관리할 수 있고, 인증서 갱신 및 배포가 간편해진다. 3. **트래픽 처리 성능이 중요한 경우**:    * ELB는 자동 확장 및 로드 밸런싱 기능을 통해 트래픽이 증가하더라도 안정적인 성능을 유지할 수 있다 4. **보안이 중요한 경우**:    * ELB는 HTTPS 트래픽을 안전하게 처리하고, Web Application Firewall(WAF)과 통합하여 보안 강화도 가능하게 해준다. |



|  |
| --- |
| . 1. 클라이언트가 애플리케이션에 요청을 보낸다  2. 로드밸런서의 리스너가 설정했던 프로토콜 및 포트에 맞는 요청을 수신한다.  3. 수신 수신기는 사용자가 지정한 규칙과 비교하여 수신 요청이 유효한지 확인하고, 유효한 경우 해당 요청을 적절한 대상 그룹으로 라우팅한다. HTTPS 수신기를 사용하여 TLS 암호화 및 암호 해독 작업을 로드 밸런서로 오프로드할 수 있다.  4. 대상 그룹에 있는 하나 이상의 정상적인 서버는 로드 밸런싱 알고리즘 및 수신기에서 지정한 라우팅 규칙에 따라 트래픽을 수신한다. |

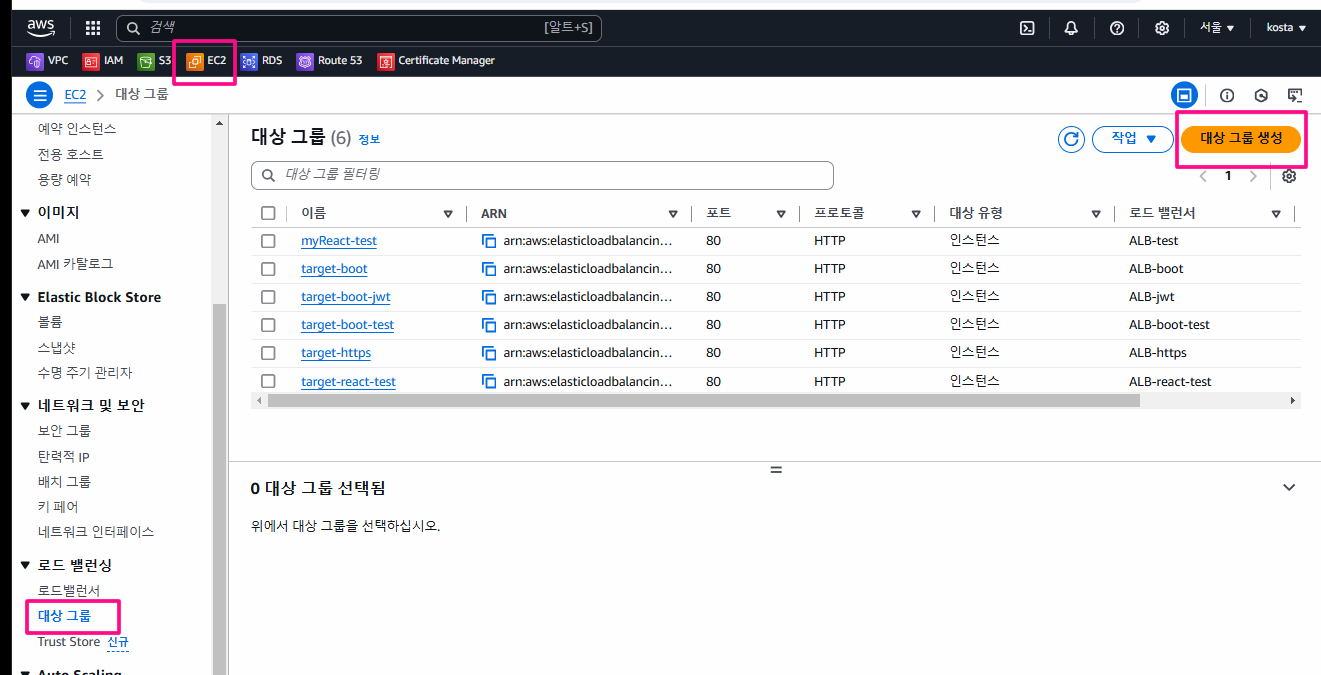
**ELB를 생성하기 전에 먼저 타켓그룹을 생성한다.**

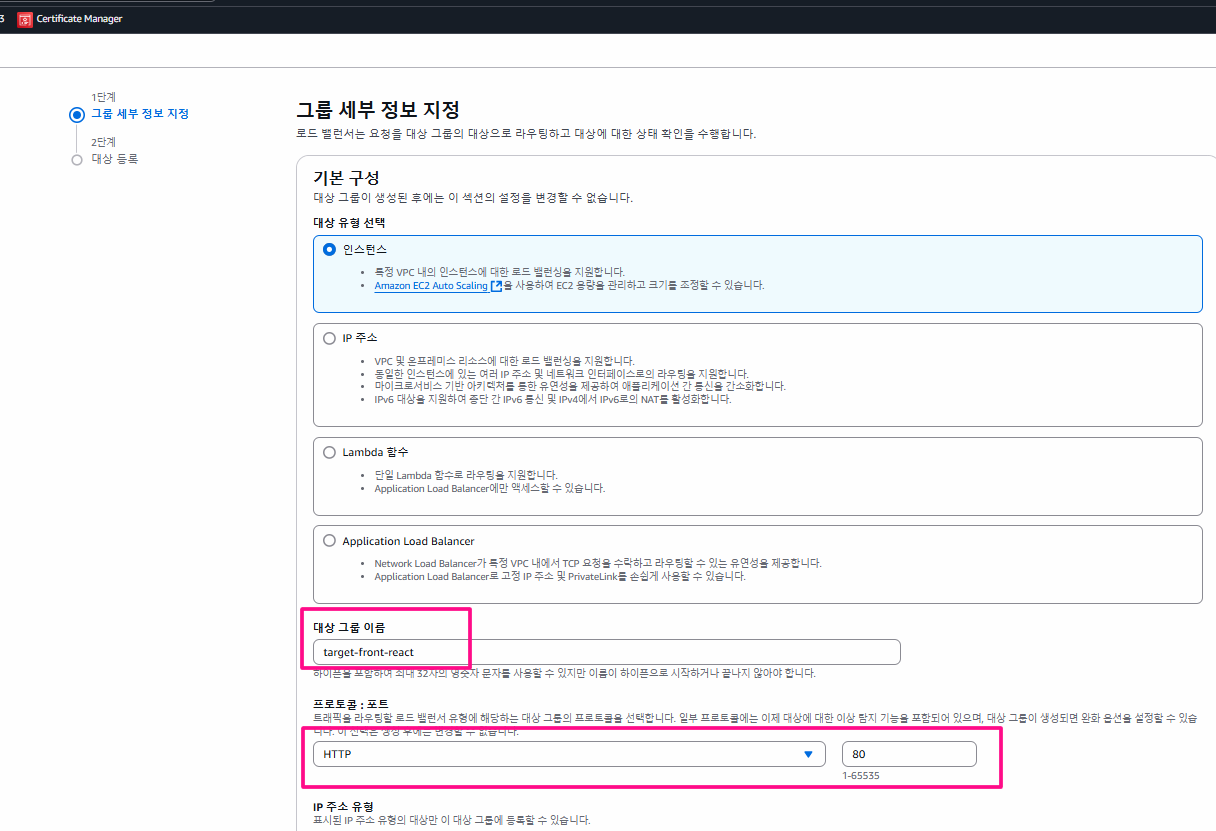
**☞ 타켓그룹에 대해서**

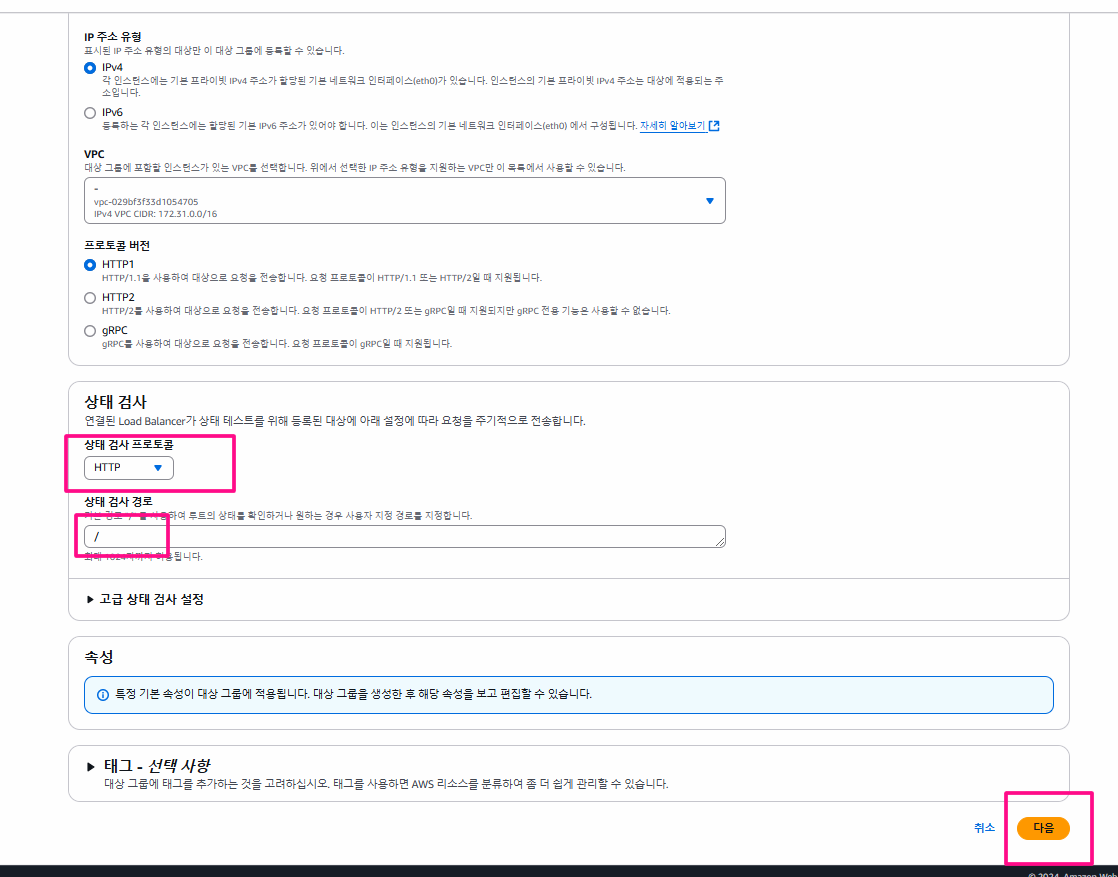
|  |
| --- |
| AWS에서 타겟 그룹(Target Group)은 Elastic Load Balancer(ELB)와 함께 사용되는 구성 요소로, 로드 밸런서가 트래픽을 분배할 대상 서버(인스턴스)들을 그룹화하여 관리하는 단위이다. 타겟 그룹은 트래픽을 분배할 대상 서버(타겟)들을 정의하며, 로드 밸런서가 특정 트래픽을 처리할 때 어떤 인스턴스에 요청을 전달할지 결정하는 데 중요한 역할을 한다..  **타겟 그룹의 주요 개념:**   1. **대상(Target)**:    * 타겟 그룹은 여러 대상을 가질 수 있으며, 대상은 **EC2 인스턴스**, **Lambda 함수**, **IP 주소** 또는 **컨테이너** 등 다양한 형태로 존재할 수 있다. ELB는 타겟 그룹을 기준으로 트래픽을 특정 대상에 전달한다. 2. **타겟 그룹의 역할**:    * 타겟 그룹은 로드 밸런서가 요청을 분배할 때 어떤 서버(인스턴스)로 트래픽을 보내야 하는지 결정하는 단위이다.    * 타겟 그룹에 속한 인스턴스들은 웹 서버 역할을 할 수 있으며, 애플리케이션을 처리하거나 API 요청을 받아 처리하는 등의 역할을 한다. 3. **헬스 체크(Health Check)**:    * 타겟 그룹은 각 인스턴스의 상태를 모니터링한다. 로드 밸런서는 타겟 그룹의 **헬스 체크**를 통해 인스턴스가 정상적으로 작동하는지 확인한다. 헬스 체크에 실패한 인스턴스는 트래픽을 받지 않으며, 로드 밸런서는 다른 정상 인스턴스로 트래픽을 전달한다. 4. **로드 밸런서와의 연결**:    * ELB(예: Application Load Balancer(ALB) 또는 Network Load Balancer(NLB)는 타겟 그룹과 연결된다. 트래픽을 분배할 때, 로드 밸런서는 요청을 받아 타겟 그룹에 설정된 인스턴스들로 분배한다.   **타겟 그룹의 종류:**   1. **Application Load Balancer (ALB)**:    * **ALB**는 주로 HTTP/HTTPS 트래픽을 처리한다. ALB에서 타겟 그룹은 **EC2 인스턴스**, **Lambda 함수**, **IP 주소** 등을 대상으로 설정할 수 있다.    * 예를 들어, 여러 웹 애플리케이션 서버가 있을 때 ALB는 요청을 각 타겟 그룹에 분배한다. 2. **Network Load Balancer (NLB)**:    * **NLB**는 주로 **TCP** 또는 **UDP** 트래픽을 처리한다.. NLB도 타겟 그룹을 사용하지만, 기본적으로 **고속 네트워크 트래픽**을 처리하는 데 최적화되어 있다.    * NLB에서 타겟 그룹은 **EC2 인스턴스**와 **IP 주소**를 대상으로 설정할 수 있다. 3. **Gateway Load Balancer (GLB)**:    * **GLB**는 **네트워크 및 보안 장비**와 연동되는 로드 밸런서로, 보안 장비나 네트워크 장비를 대상으로 하는 타겟 그룹을 설정할 수 있다.   **타겟 그룹을 사용하는 이유:**   1. **로드 밸런싱**:    * 타겟 그룹을 사용하면 로드 밸런서가 트래픽을 여러 인스턴스에 분배할 수 있게 되며 이를 통해 애플리케이션의 성능과 확장성을 향상시킬 수 있다. 2. **고가용성**:    * 여러 인스턴스에 트래픽을 분배함으로써 **고가용성**을 확보할 수 있다. 한 인스턴스가 다운되더라도 다른 인스턴스가 트래픽을 처리할 수 있게 된다. 3. **헬스 체크**:    * 로드 밸런서는 타겟 그룹의 인스턴스 상태를 지속적으로 모니터링한다. 헬스 체크에 실패한 인스턴스는 트래픽을 받지 않게 되어, 장애가 발생한 서버로 트래픽을 전달하지 않도록 할 수 있다. 4. **자동 확장**:    * **Auto Scaling**과 결합하여, 트래픽 부하가 커지면 새로운 인스턴스를 자동으로 타겟 그룹에 추가하고, 부하가 줄어들면 인스턴스를 자동으로 제거할 수 있다. 5. **다양한 리소스 관리**:    * 타겟 그룹을 사용하면, EC2 인스턴스 외에도 Lambda 함수나 IP 주소 등을 타겟으로 설정하여 다양한 리소스를 관리할 수 있다.   AWS에서 **타겟 그룹**은 로드 밸런서가 트래픽을 분배할 대상 인스턴스를 그룹화하여 관리하는 단위로 타겟 그룹을 사용하면 트래픽 분배, 고가용성, 헬스 체크, 자동 확장 등의 기능을 효율적으로 설정하고 관리할 수 있다. ELB는 이러한 타겟 그룹을 통해 여러 서버에 부하를 고르게 분배하며, 안정적이고 확장 가능한 서비스를 제공한다. |

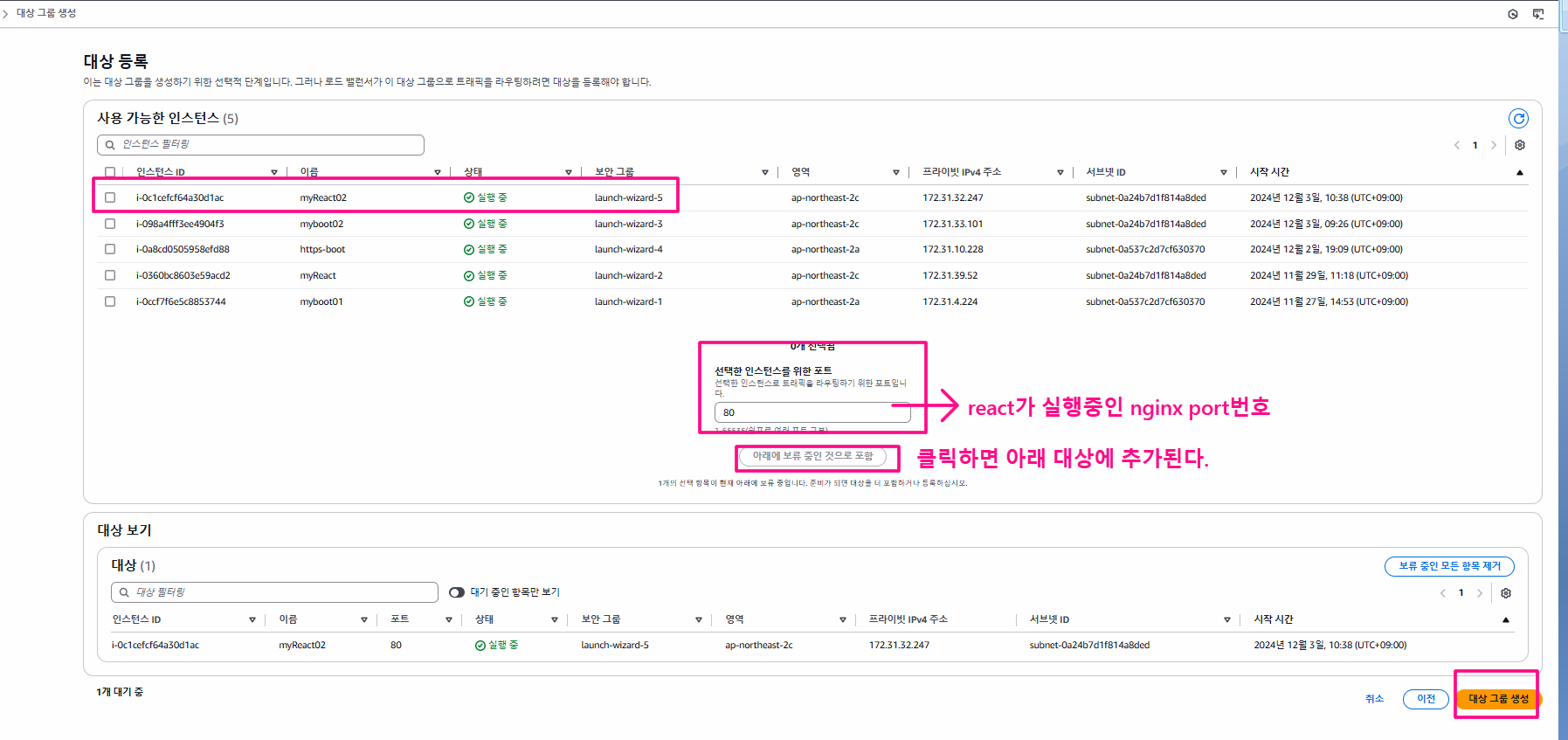
**이제 실습해보자.**

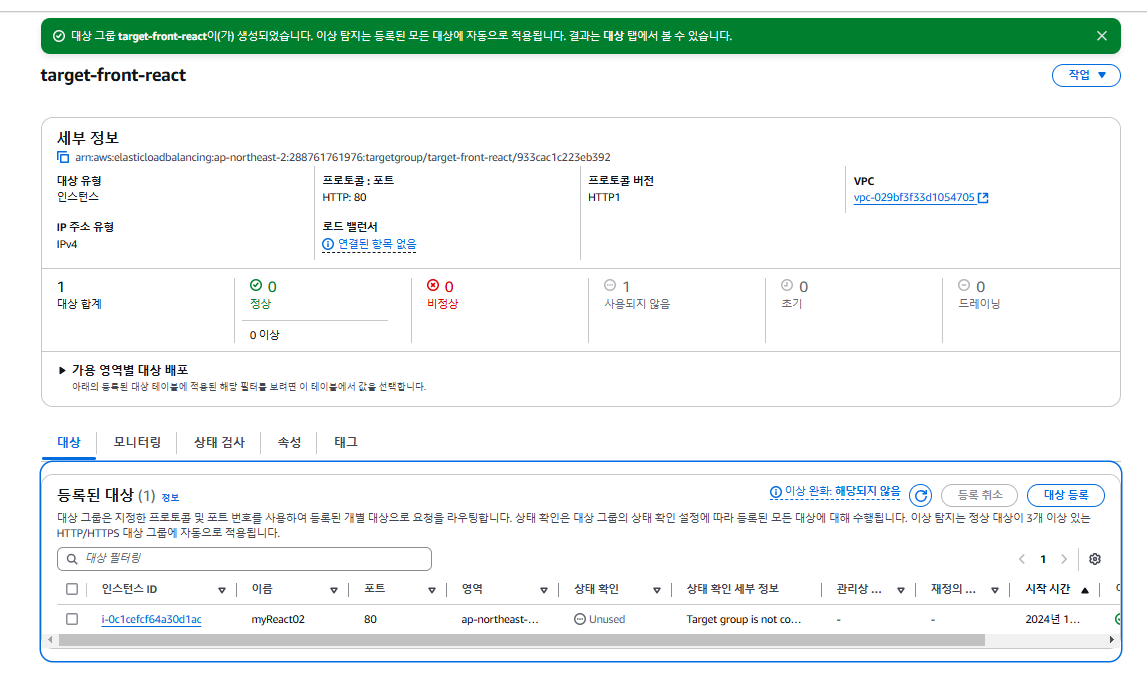
**EC2를 클릭 > 로드밸런싱 > 대상그룹 선택 > 대상그룹 생성 선택**



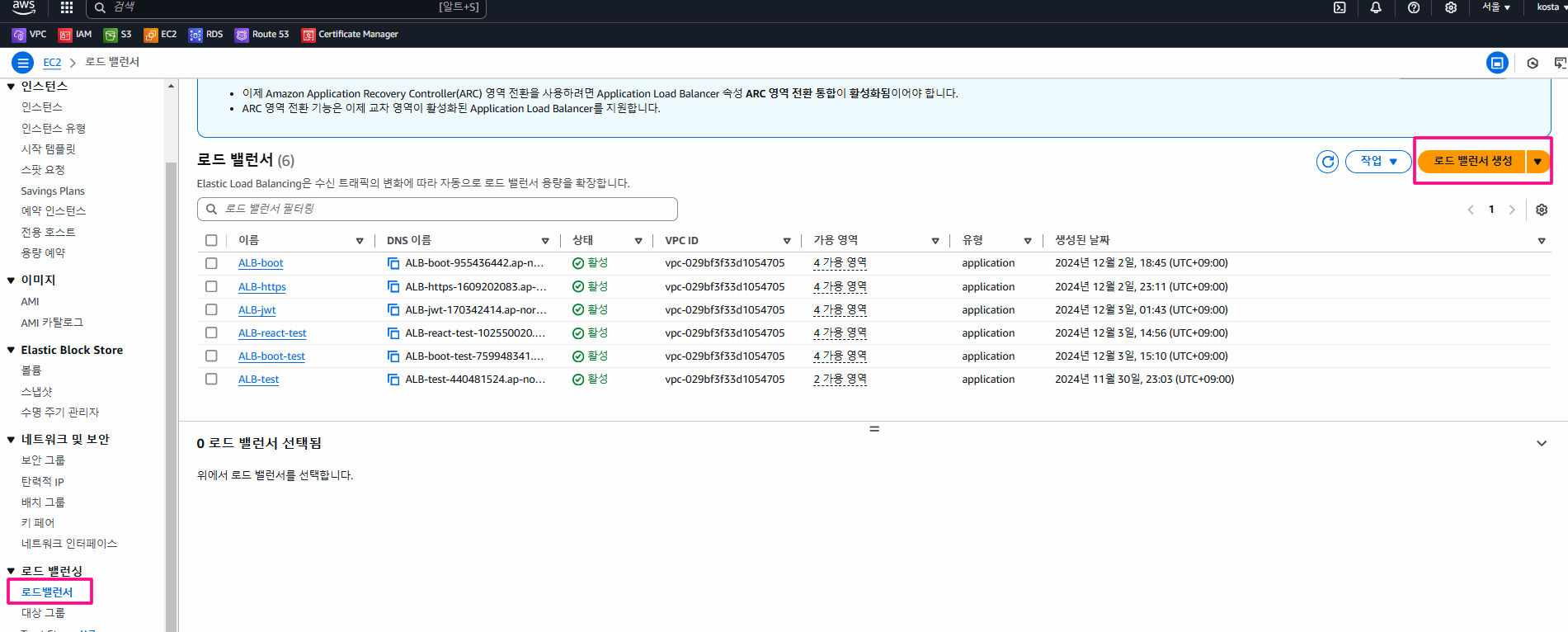


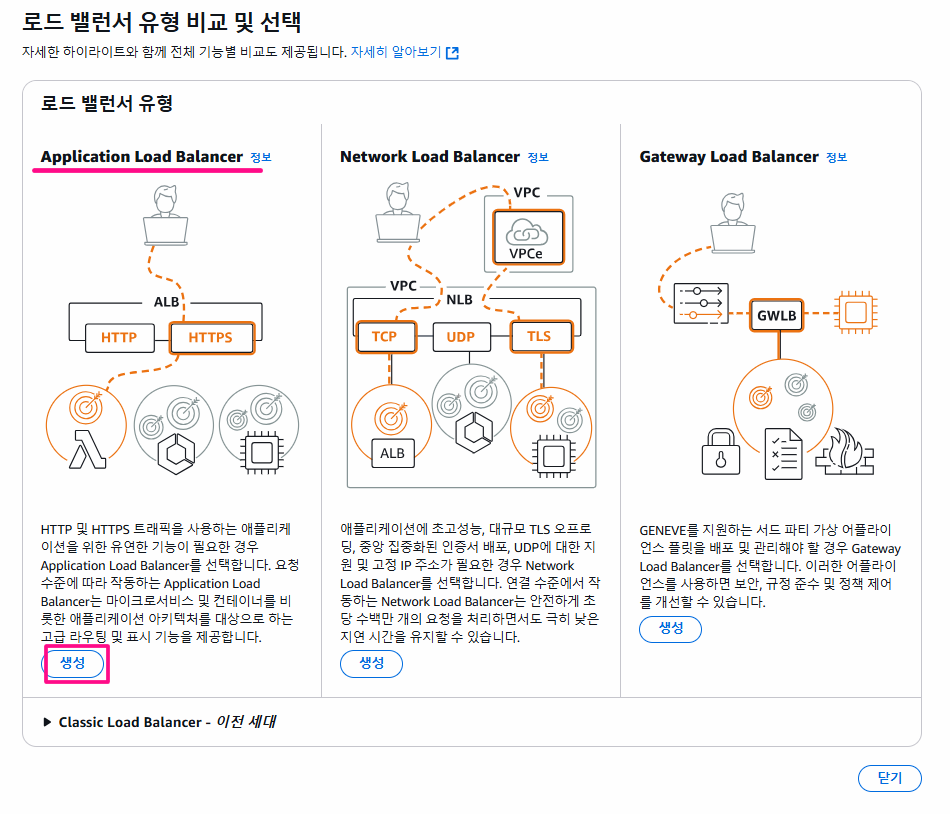


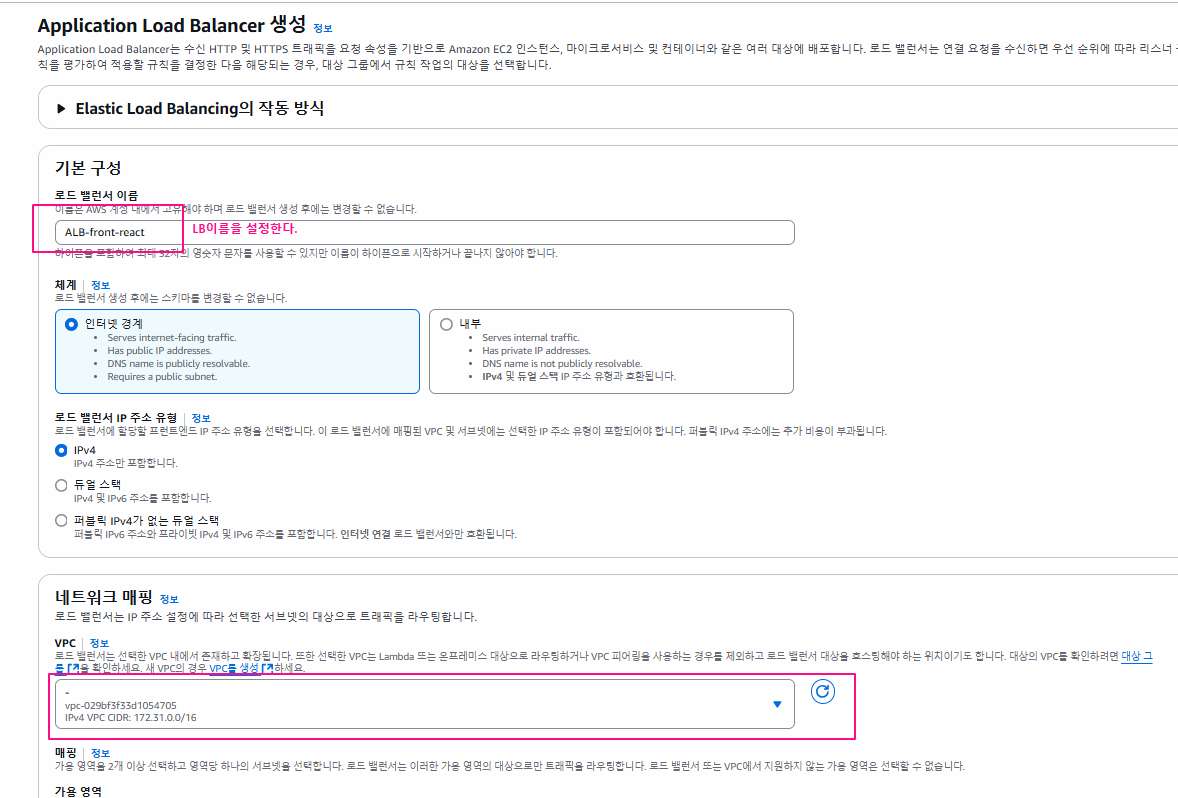


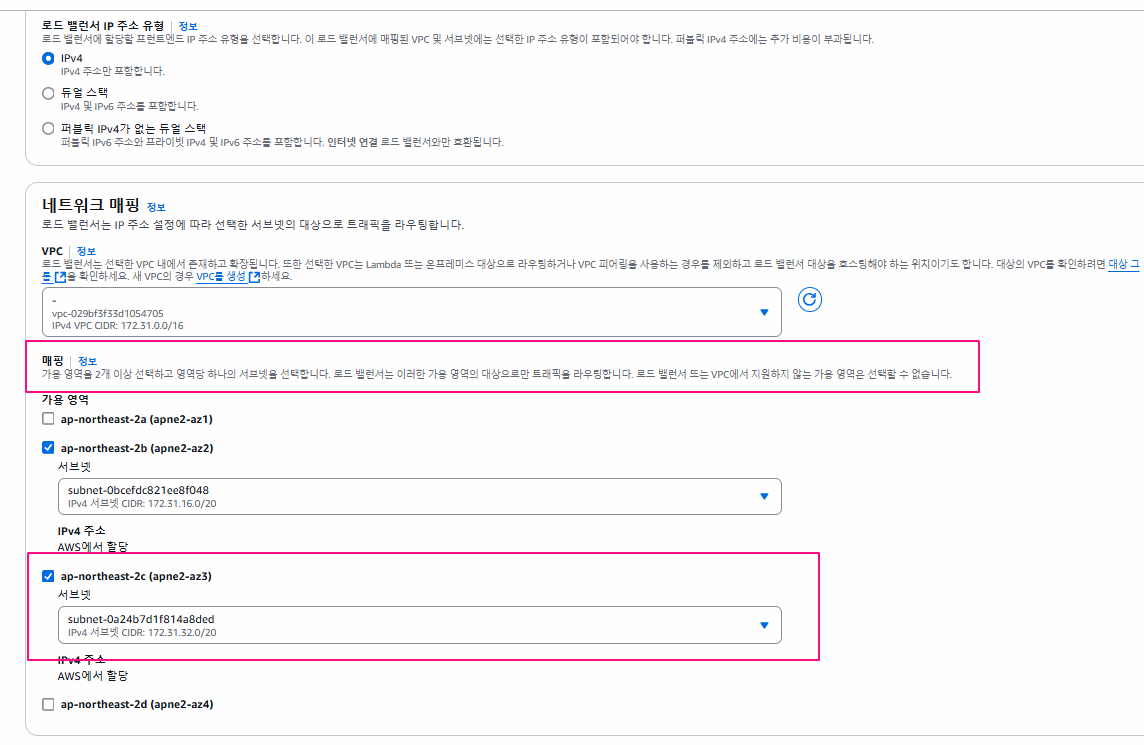


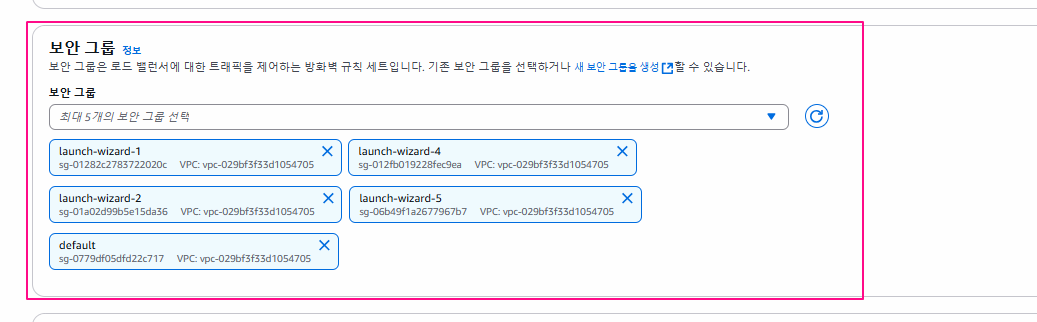
**왼쪽 메뉴에서 로드밸런싱 > 로드밸런서 클릭 > 로드밸런서 생성**

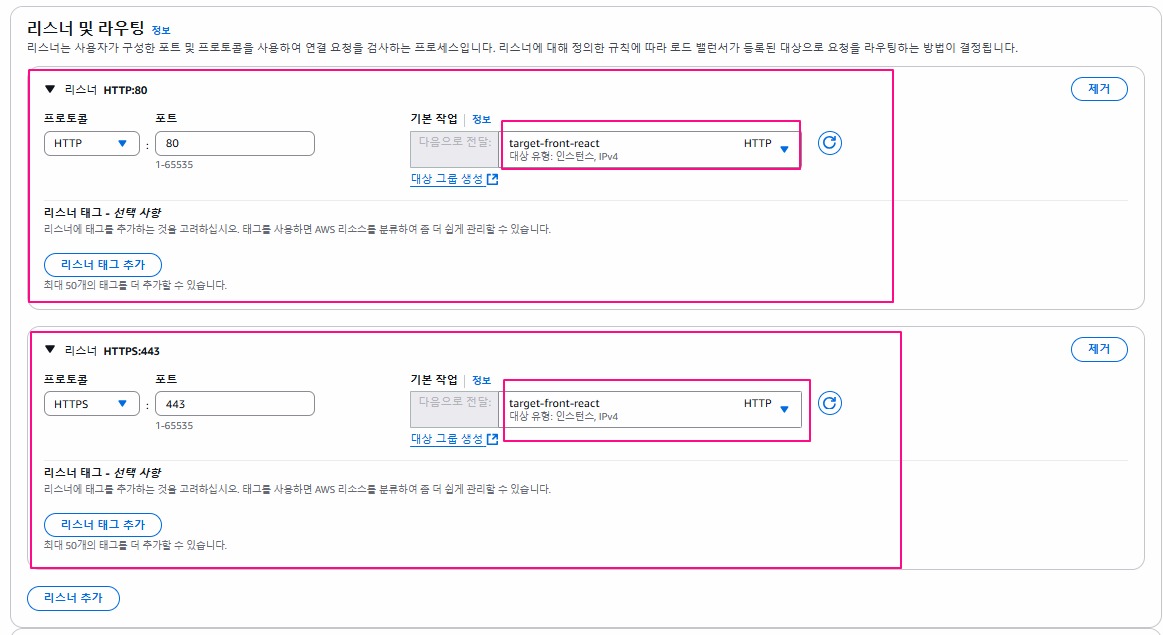


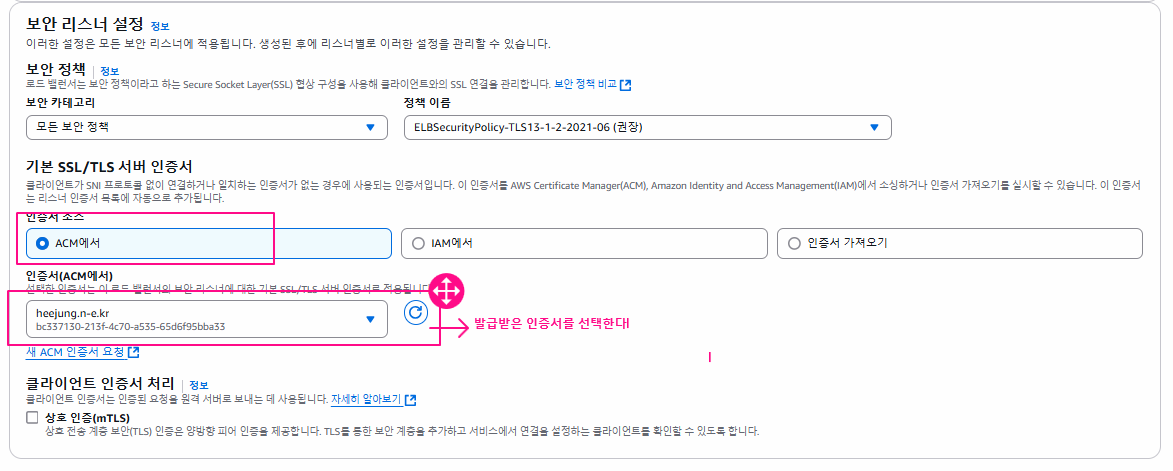


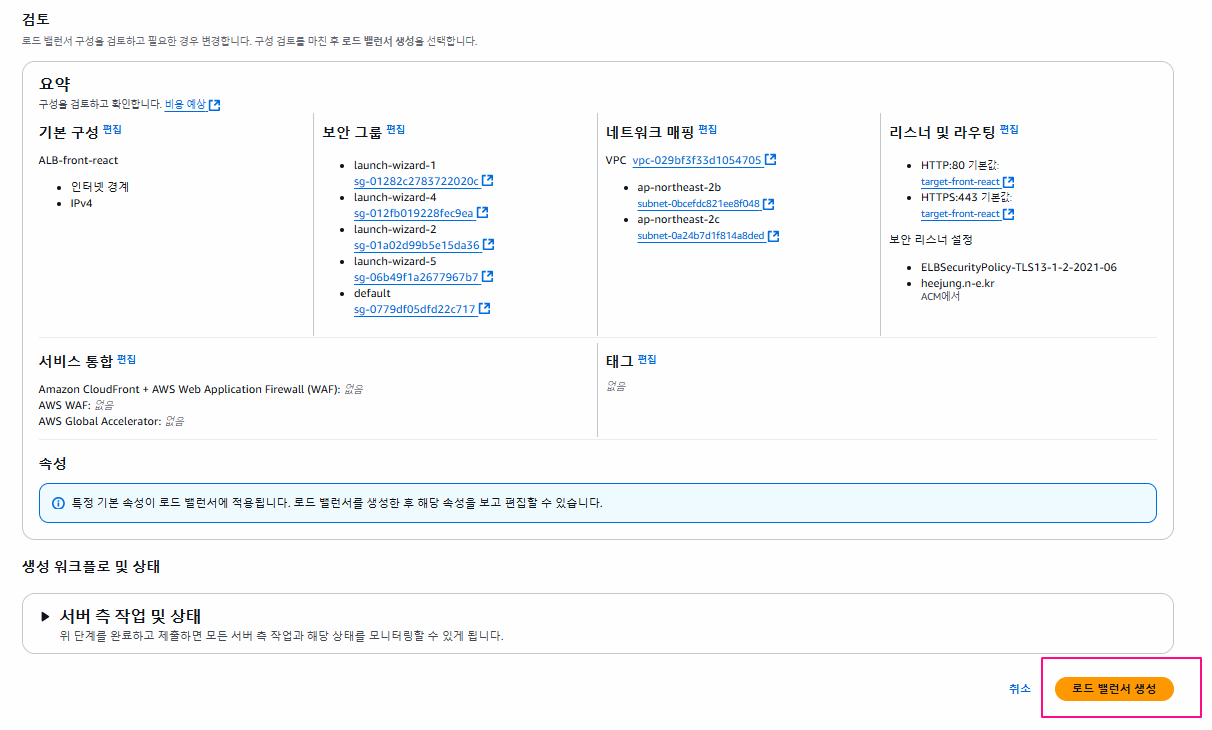


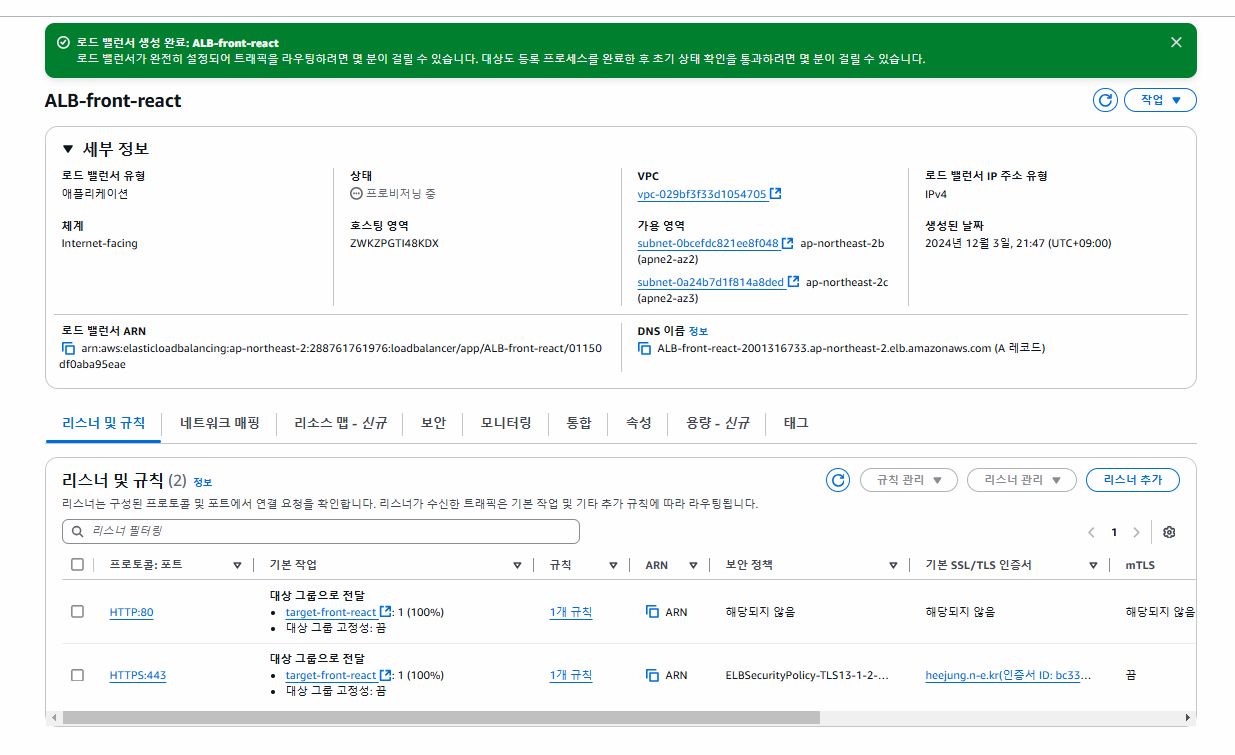






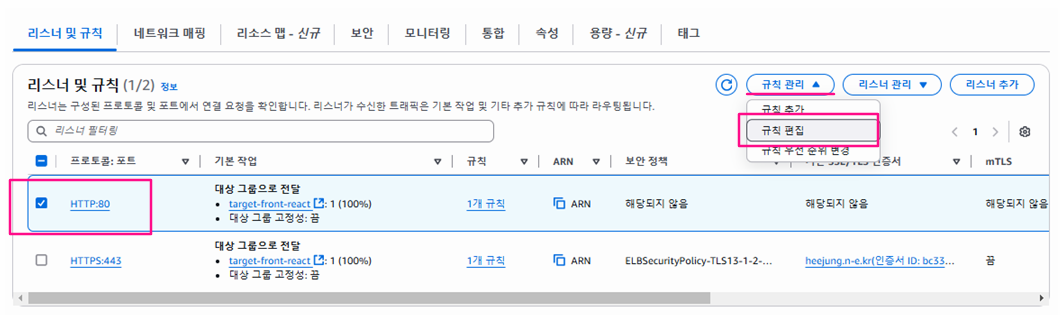


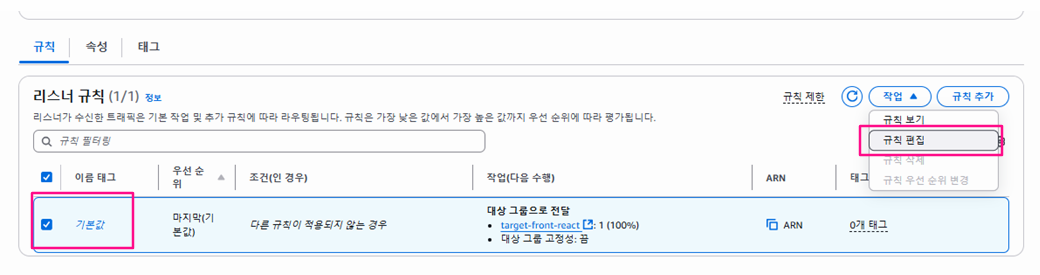


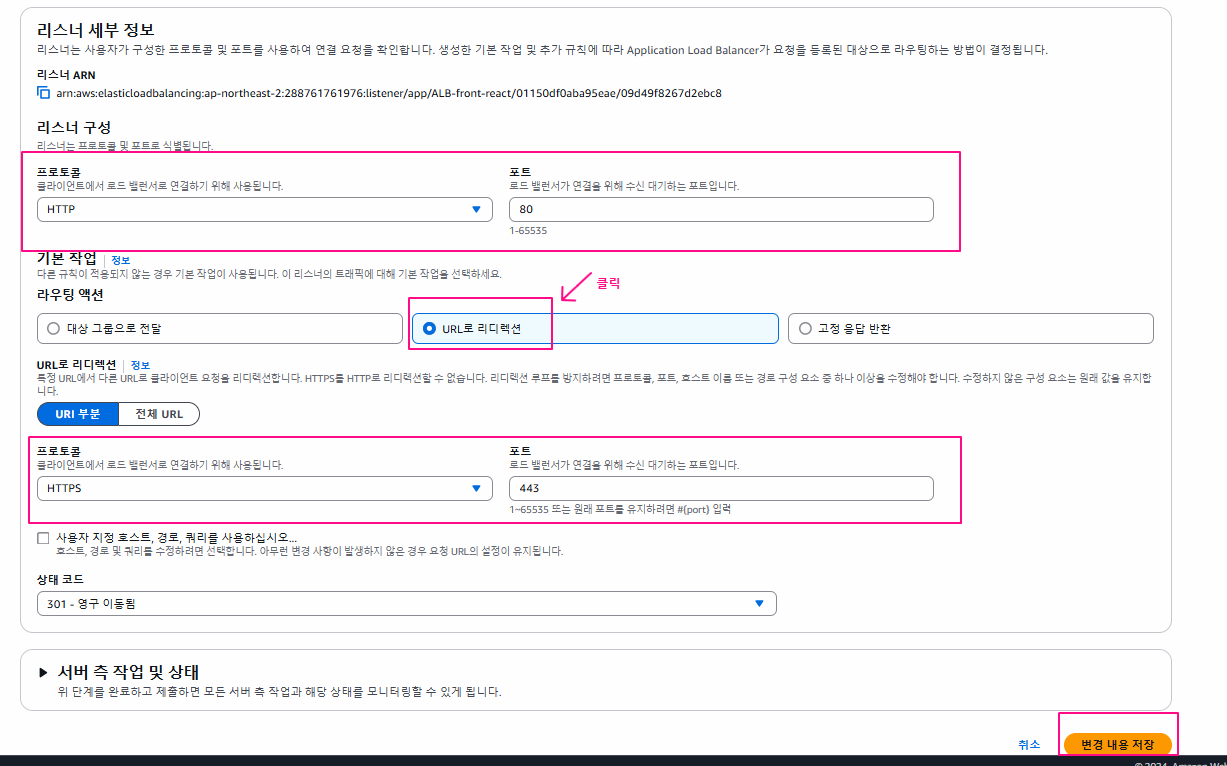


위 화면에서 http:80을 선택 하고 규칙관리 > 규칙편집을 클릭한다.

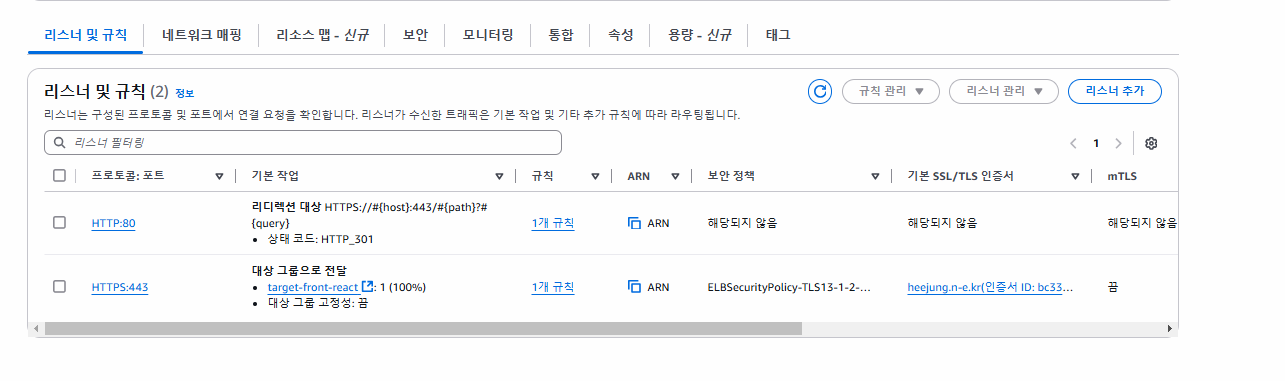
**: http로 요청을 하더라도 https로 요청이 될 수 있도록 설정이 필요하다.**

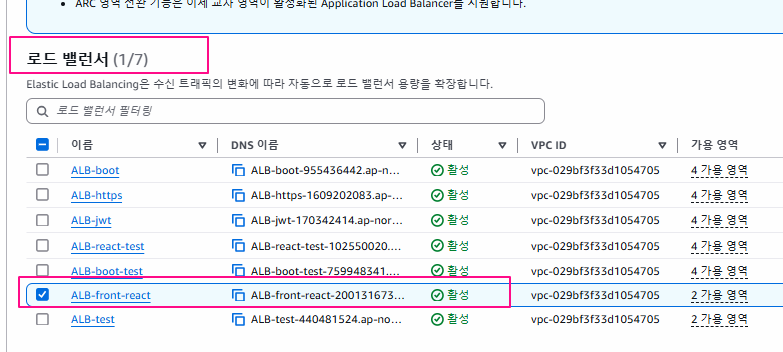




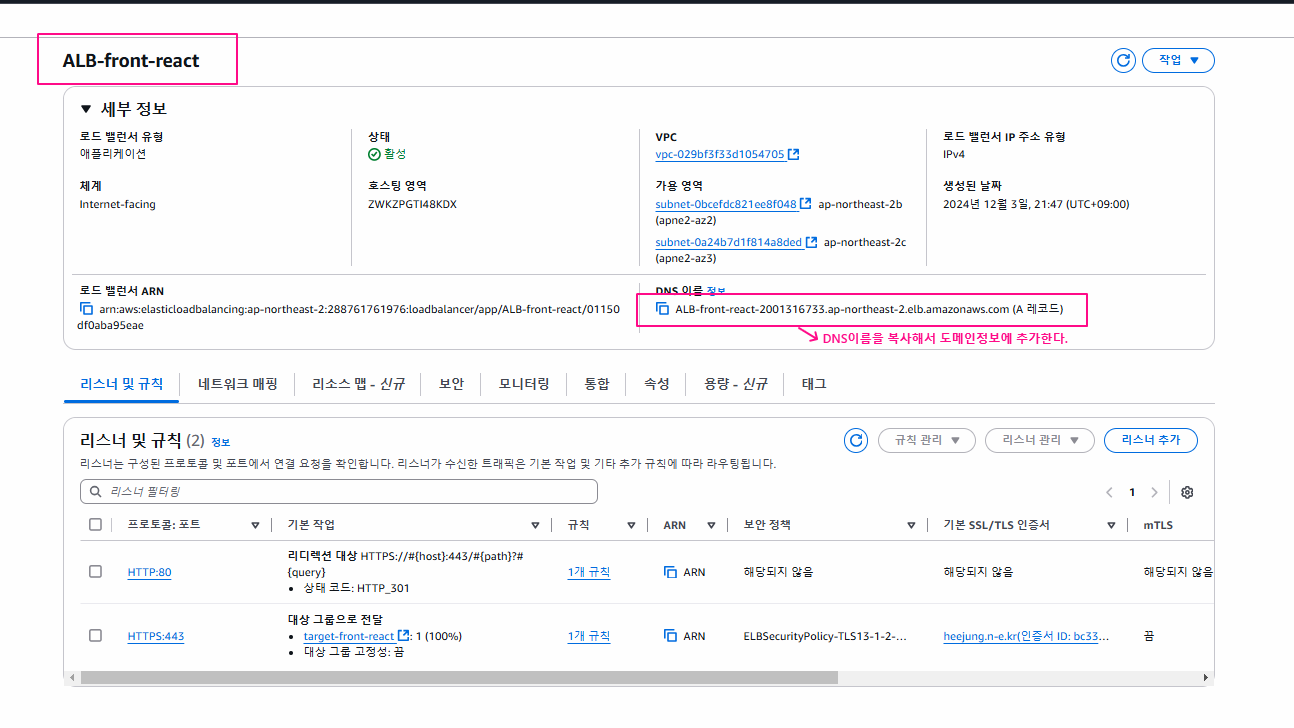


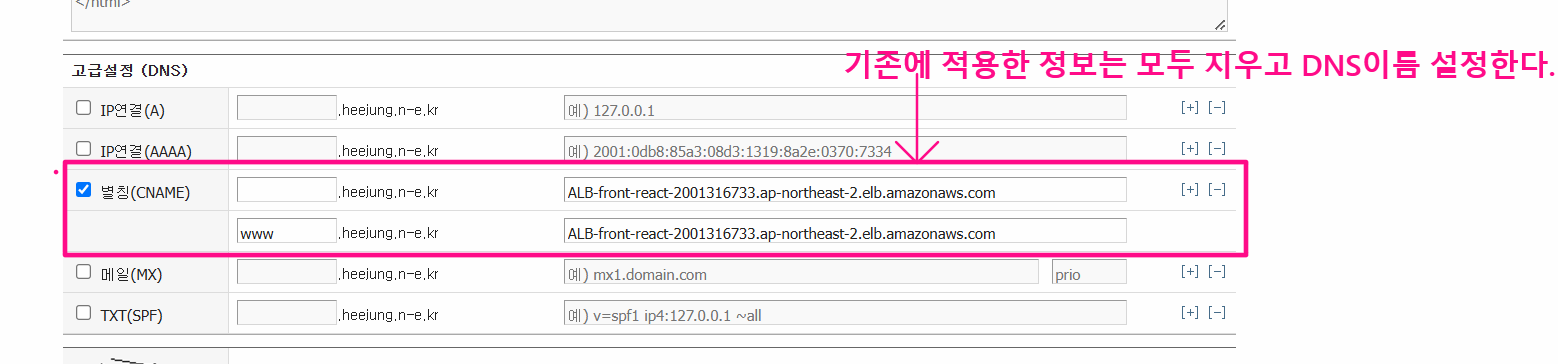
**설정이 완료 된 후에 로드밸런서를 클릭해서 확인해본다.**

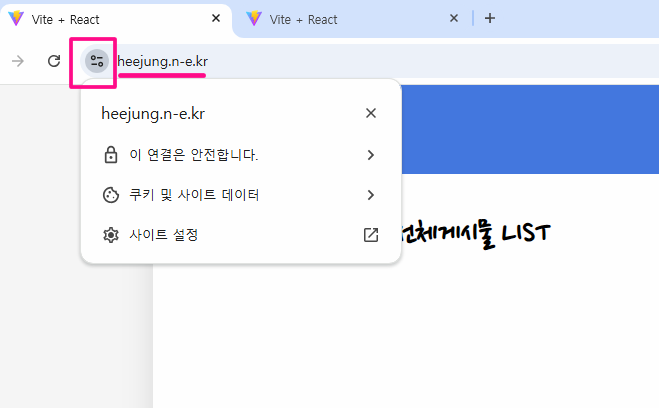




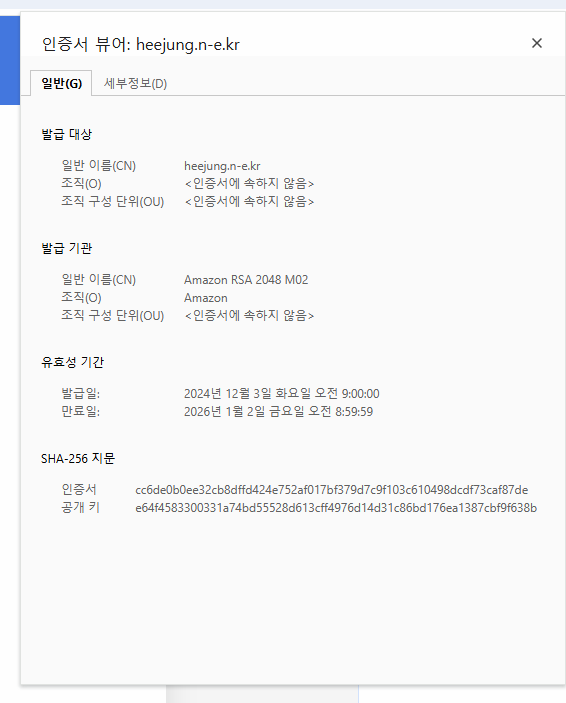
**“내도메인한국” 접속해서 도메인정보를 수정한다.**







**“위의 이 연결은안전합니다.” 클릭 > “인증서가 유효함” 클릭**



url주소를 복사해서 보면 아래처럼 https로 요청된 것을 볼 수 있다.

<https://heejung.n-e.kr/>

**그러나,**

**backend Spring관련 데이터를 가져오지 못했다.!!!**

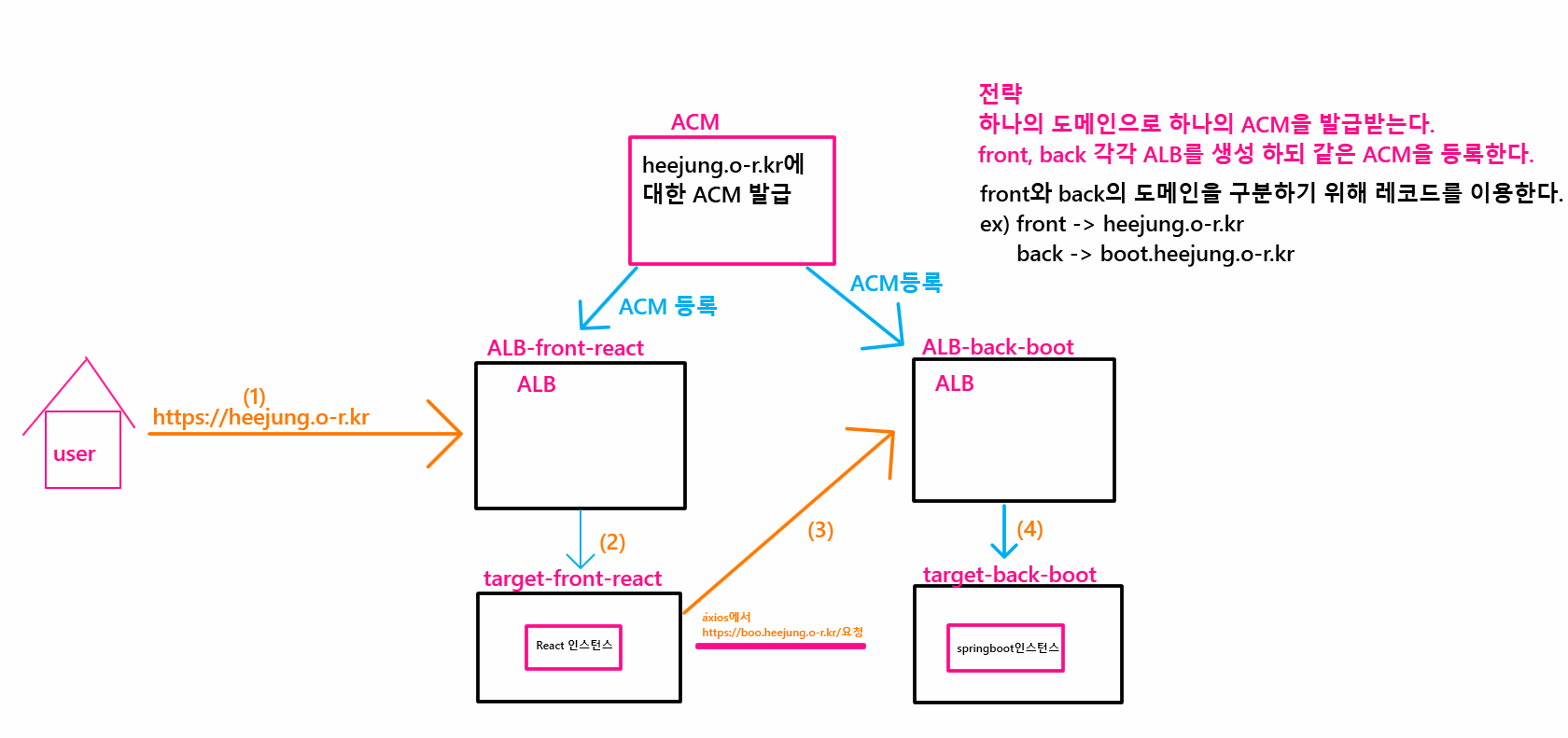
개발자도구를 클릭하여 콘솔에 오류를 확인해보자.

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| 이 에러는 **Mixed Content** 문제로, 웹 페이지가 **HTTPS** 프로토콜을 사용하여 로드되었지만, 페이지 내에서 **HTTP** 프로토콜을 사용하는 리소스를 요청했을 때 발생한다. **브라우저는 보안을 위해 HTTPS 페이지에서 HTTP 리소스를 불러오는 것을 차단한다.**  **문제의 원인**  URL이 https:/heejung.n-e.kr/로 시작하는 웹 페이지는 HTTPS(보안 연결)를 사용해 로드되고 있지만 XMLHttpRequest가 HTTP로 요청되었다. 요청을 보내는 엔드포인트인 http://54.180.0.7:8080/boards는 HTTP로 시작한다. 이는 보안이 없는 연결로, HTTPS 페이지에서 이 리소스를 요청하면 보안상의 이유로 브라우저가 이를 차단한다.  .  **왜 차단되는가?**   * **HTTPS** 웹사이트는 데이터 전송을 암호화하고 보안성을 제공하기 위해 사용된다. * 만약 HTTPS 페이지에서 HTTP로 리소스를 요청하면, 그 요청은 암호화되지 않으며 중간에서 가로채어질 수 있는 위험이 존재한다. * 브라우저는 이러한 보안상의 위험을 막기 위해 **Mixed Content**를 차단한다.   **해결 방법**   1. **HTTP 요청을 HTTPS로 변경**:    * 가능한 경우, http://54.180.0.7:8080/boards엔드포인트를 HTTPS로 변경하는 것이 가장 좋은 해결책이다.    * 서버가 HTTPS를 지원하도록 설정하고, 요청 URL을 https://3.34.71.73:9000/boards로 수정한다. 2. **서버에 SSL 인증서 적용**:    * 엔드포인트인 http://54.180.0.7:8080/boards가 SSL을 지원하지 않는다면, 서버에 SSL 인증서를 설치하여 HTTPS 요청을 처리할 수 있도록 설정해야 한다. 3. **리디렉션 설정**:    * HTTP 요청을 HTTPS로 자동 리디렉션하는 방법도있다. 서버가 HTTP로 들어오는 요청을 HTTPS로 리디렉션할 수 있도록 설정하는 것이다.    * 예를 들어, http://54.180.0.7:8080/boards로 들어오는 모든 요청을 https://54.180.0.7:8080/boards으로 리디렉션하도록 서버 설정을 변경한다. 4. **프론트엔드 수정**:    * 클라이언트 측에서 요청을 보내는 URL을 https://로 수정할 수도 있다. 즉, API 요청을 보내는 코드에서 HTTP 대신 HTTPS를 사용하도록 한다.   **결론**  웹 페이지가 HTTPS로 로드되었을 때, 모든 리소스와 API 요청 또한 HTTPS로 이루어져야 한다. 이를 통해 **Mixed Content** 문제를 해결할 수 있으며, 페이지의 보안성을 유지할 수 있다. |

**우리의 해결 방법**



**이름규칙**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ALB이름** | **target그룹 이름** | **인스턴스** |
| **Front** | ALB-front-react | target-front-react | React 인스턴스 |
| **back** | ALB-back-boot | target-back-boot | Springboot 인스턴스 |

Front의 ALB세팅은 되었으니

1)back에 해당하는 타켓그룹 생성 하고 ALB를 생성한다.

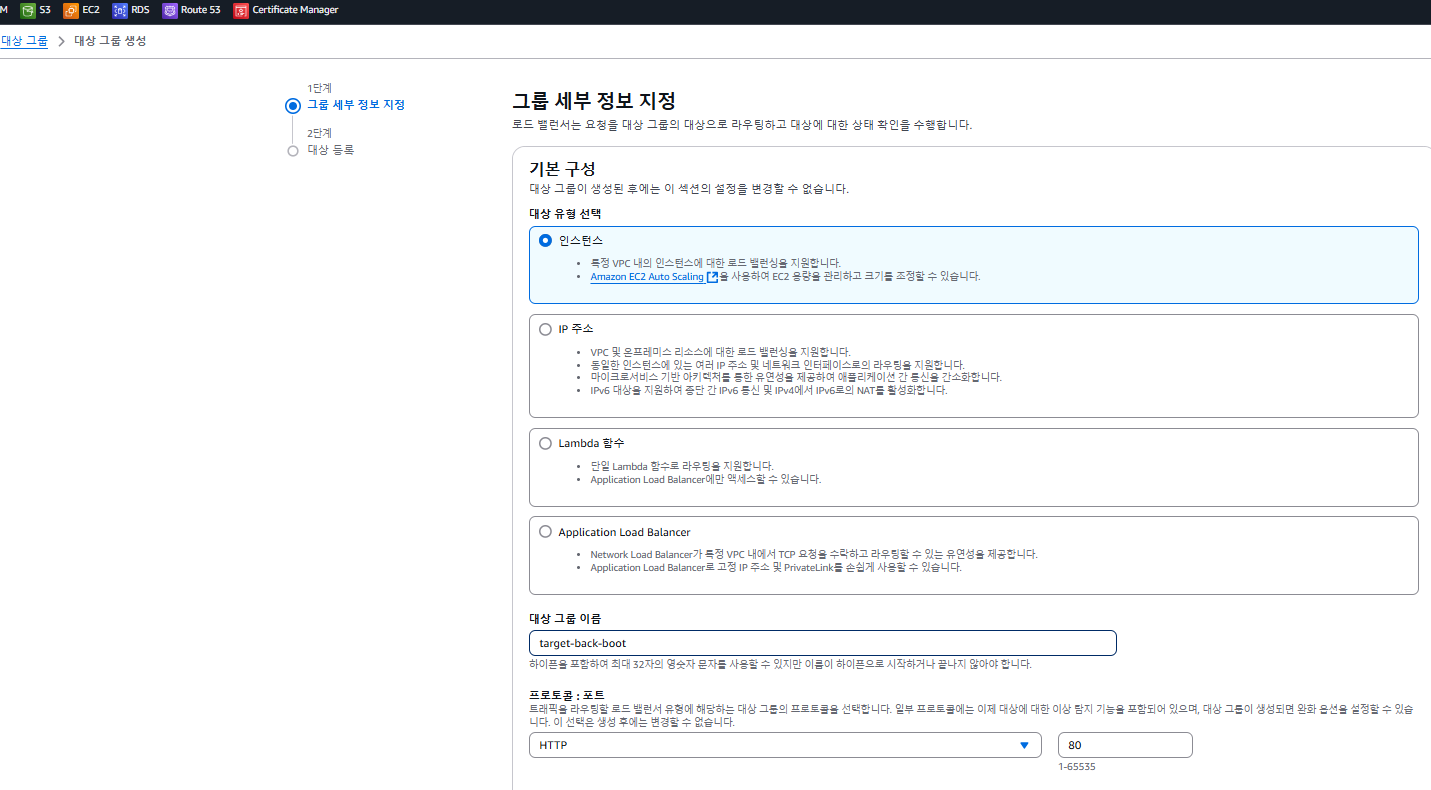
2) “내도메인한국”에 접속해서 도메인수정한다.

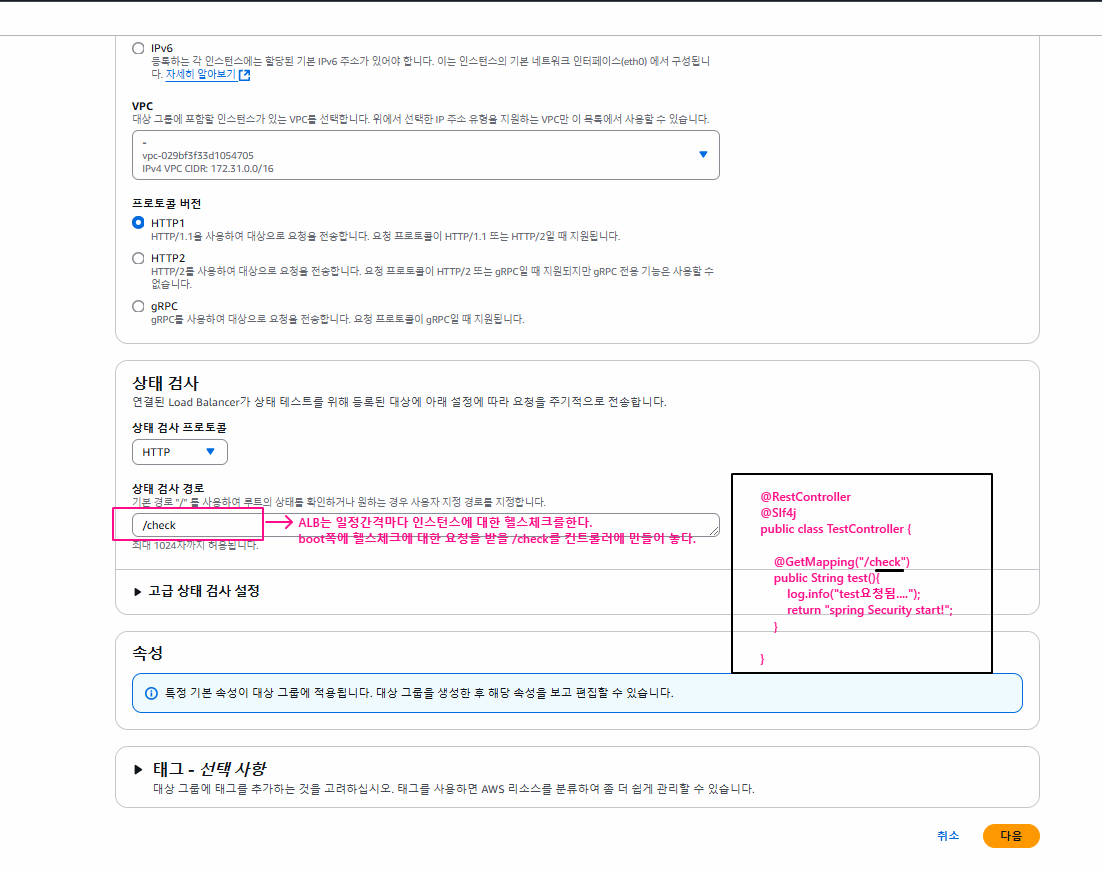
3)React 개발소스에서 .env의 파일을 아래와 같이 수정한다.

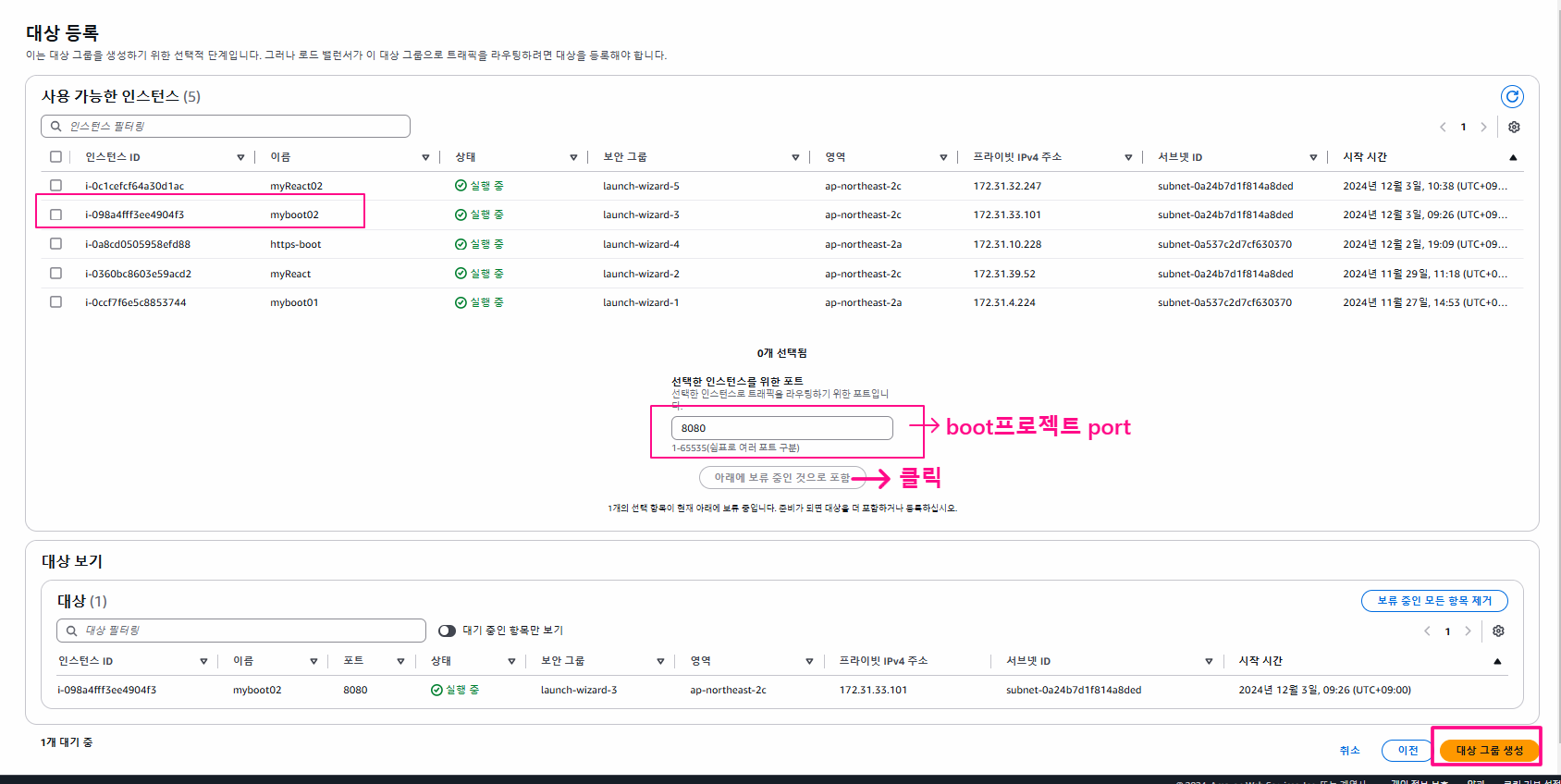
텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

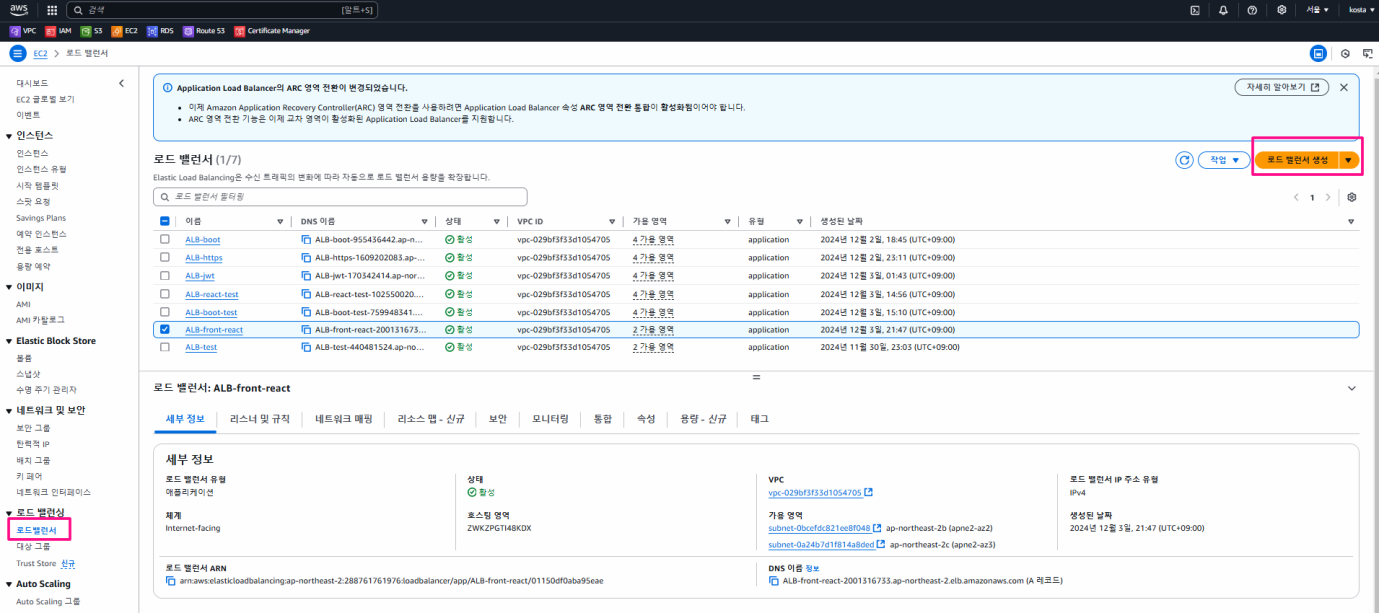
4)개발소스 build > git push > 인스턴스접속 > git clone > 파일이동 > nginx restart 한다.

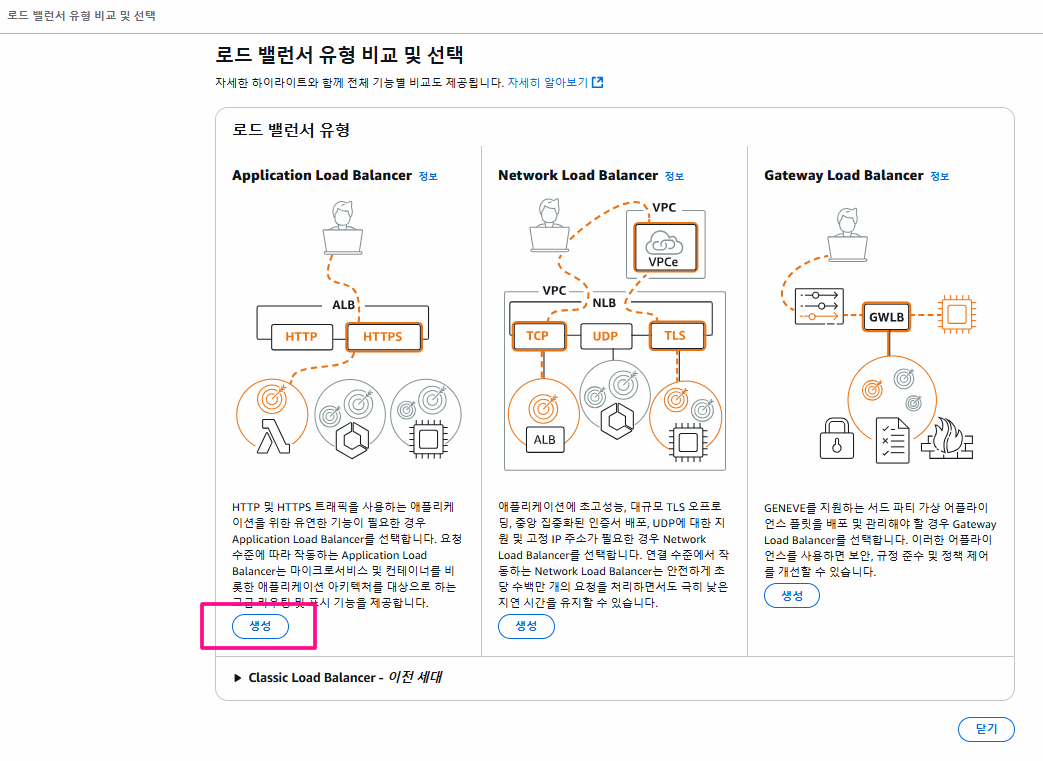


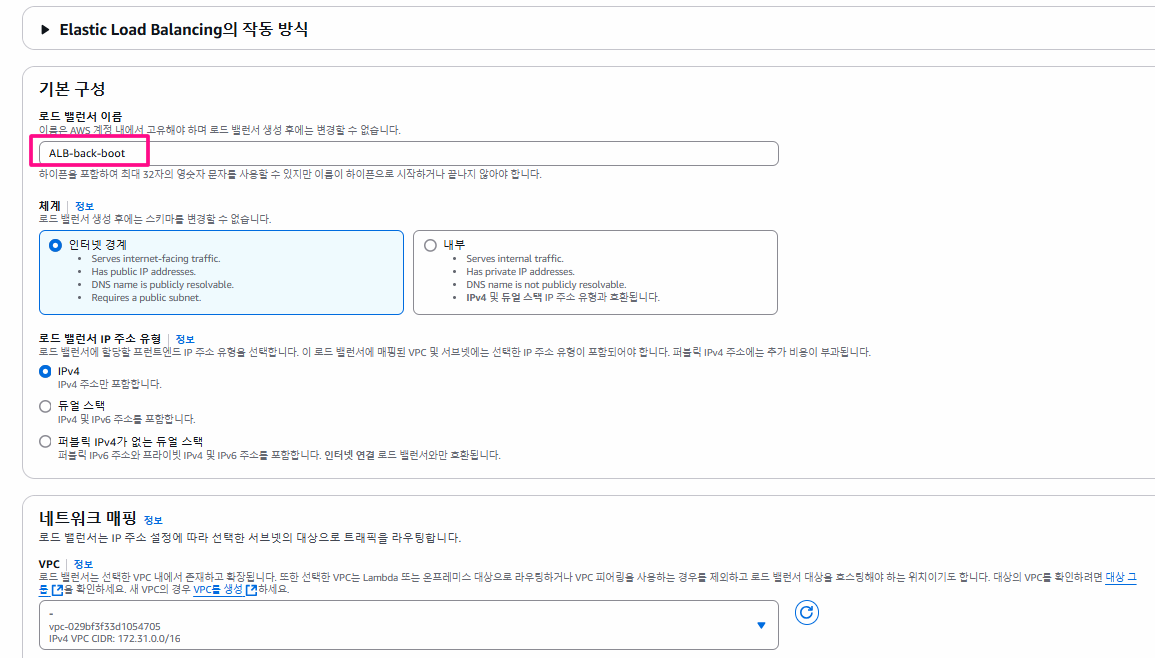


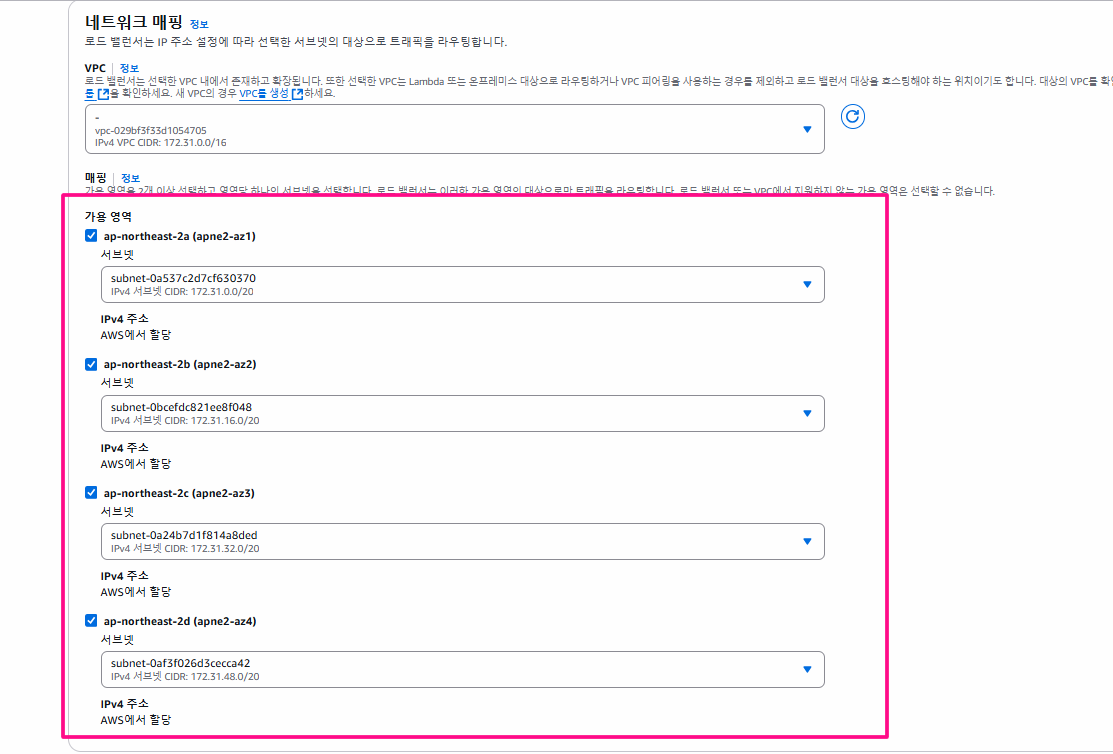


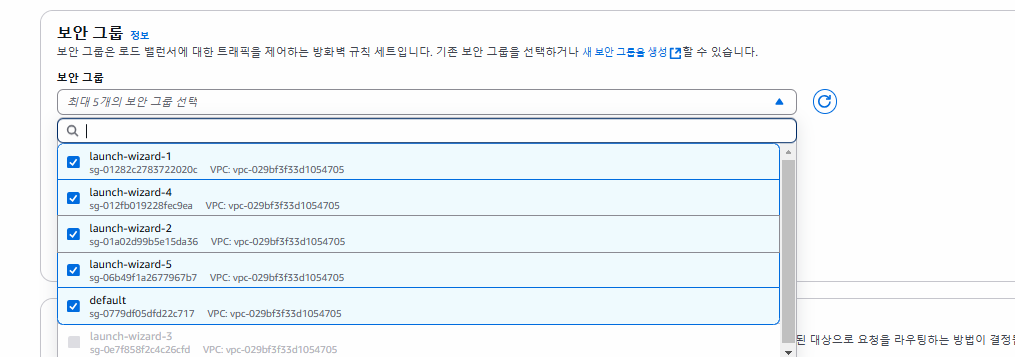
LB생성하기

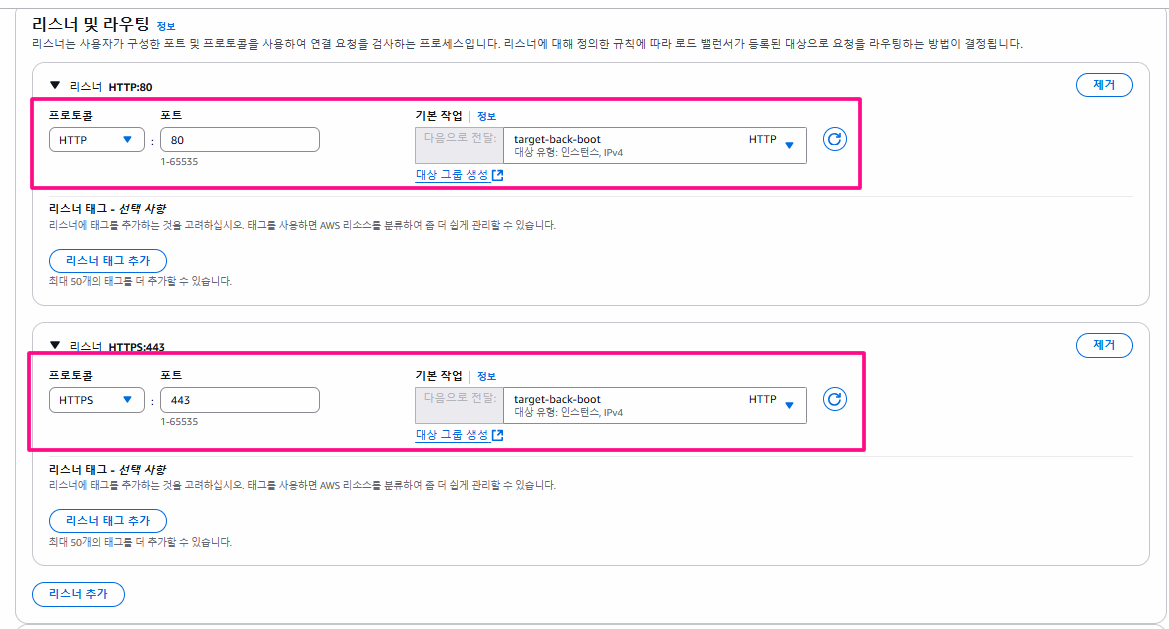




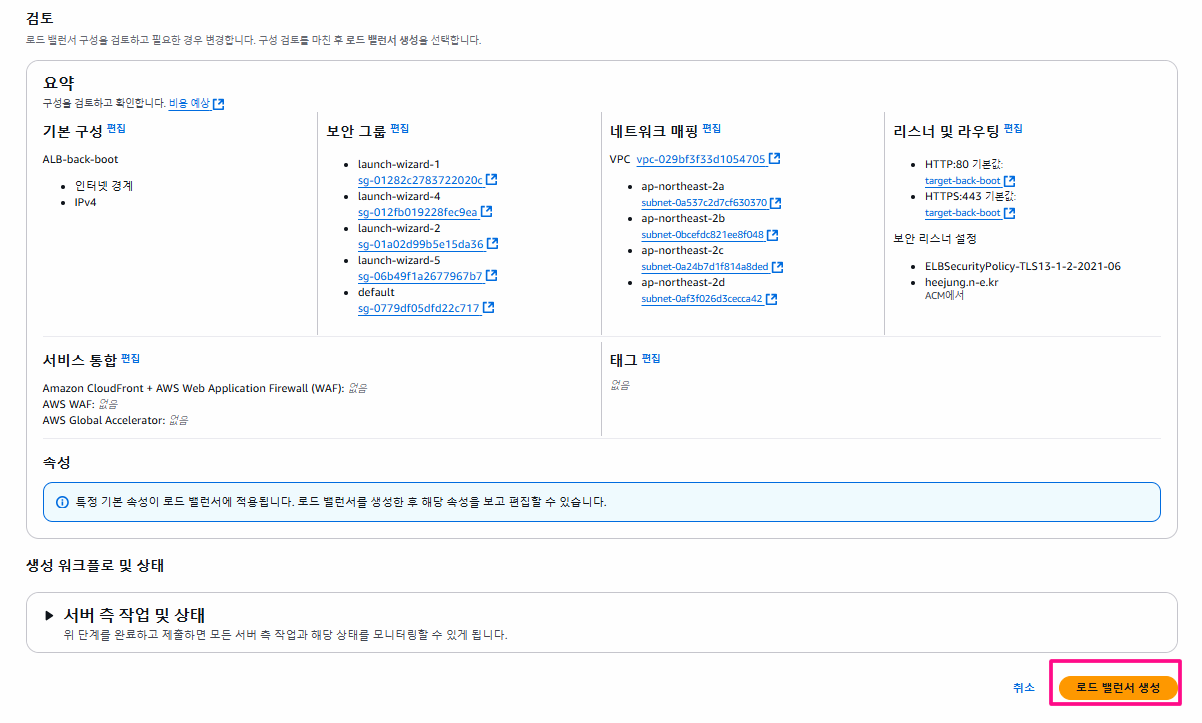




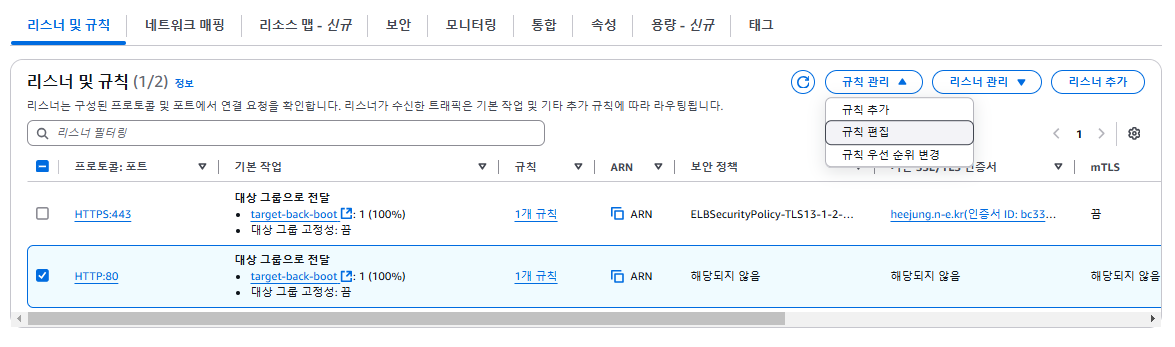


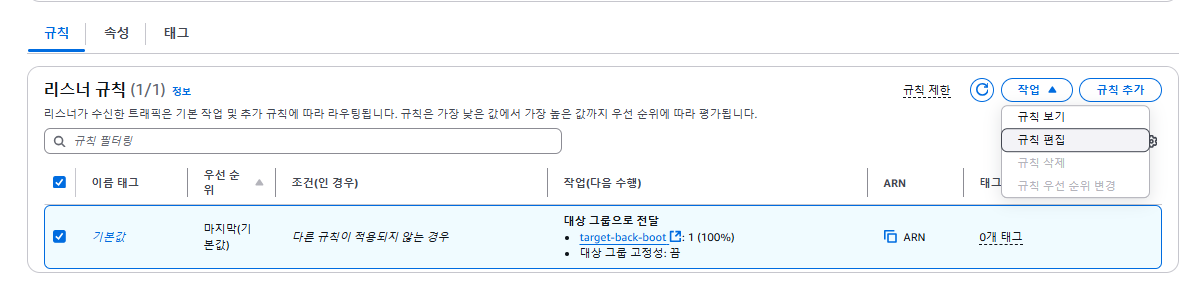


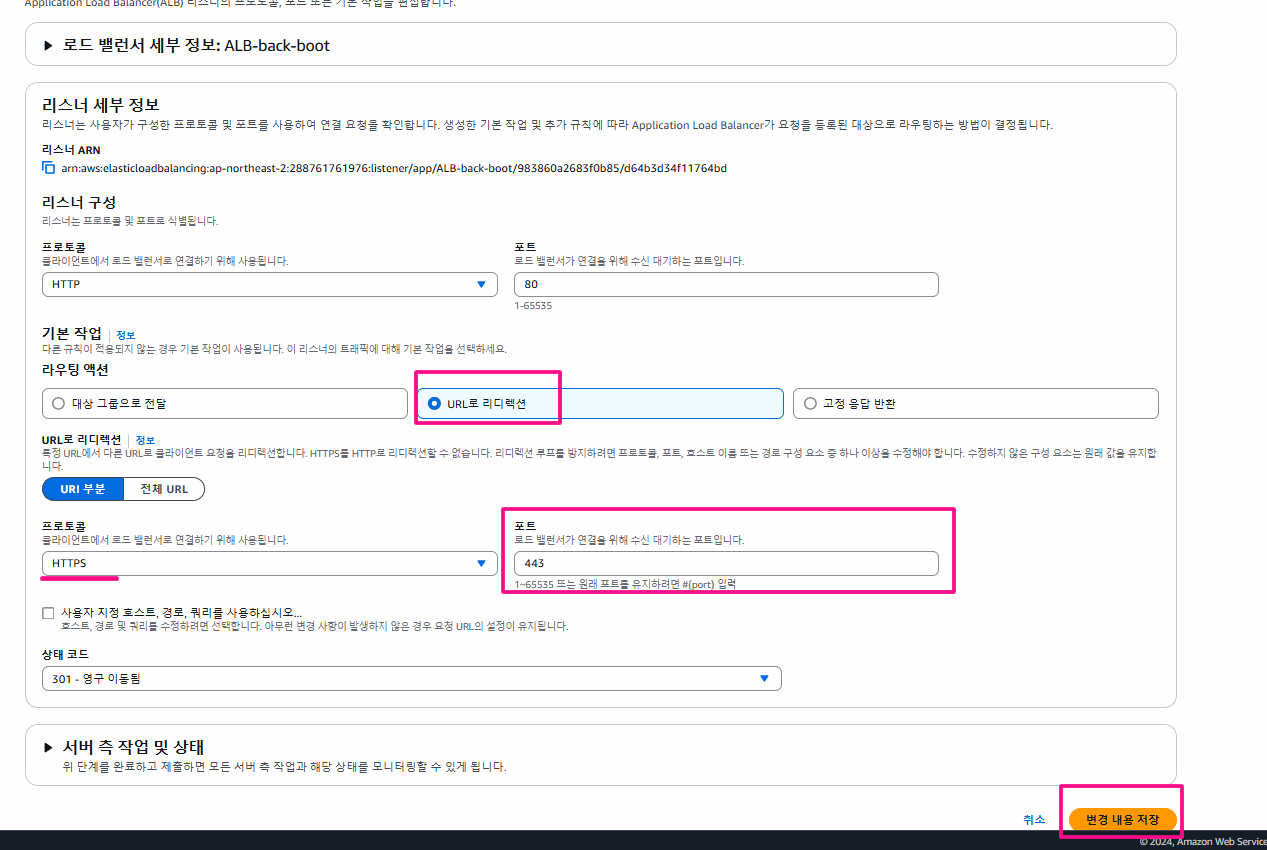


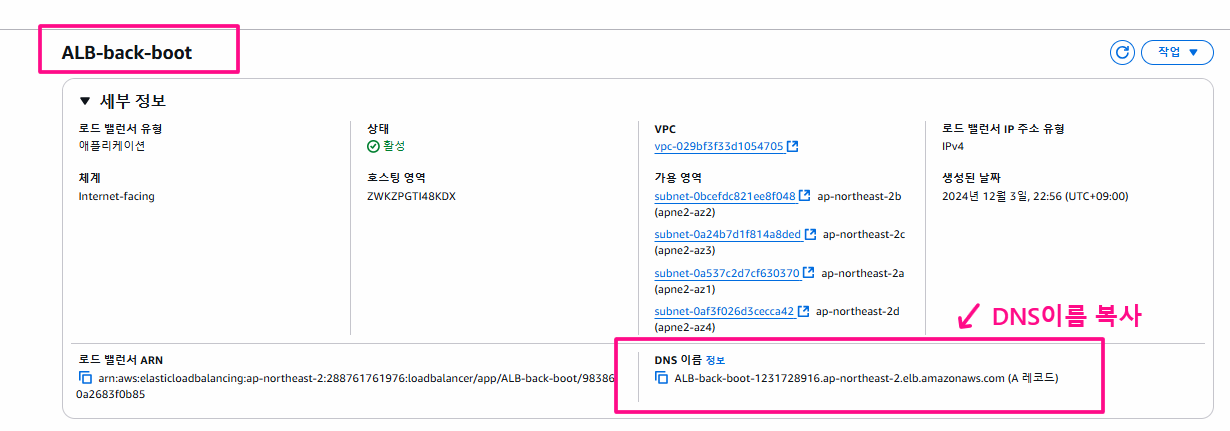


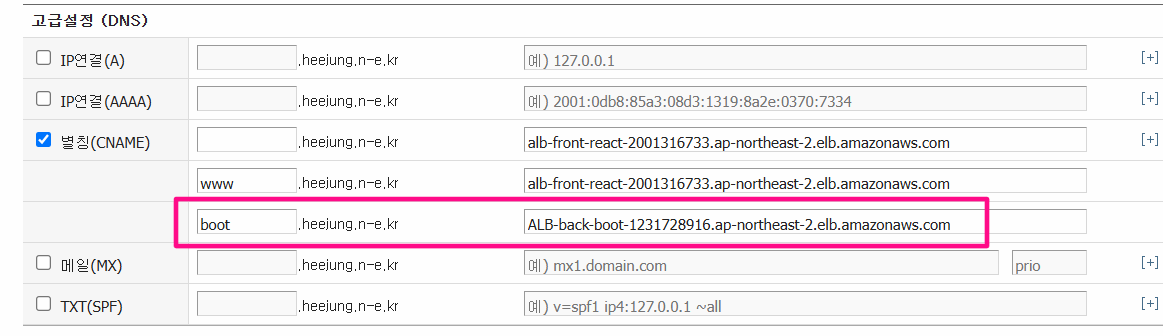
생성완료 후 리스너규칙에서 http:80의 규칙편집을 클릭해서 수정을 한다.











Front소스를 수정한다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| react프로젝트에 있는 dist폴더를 삭제한다.  npm run build  git add ./dist  git commit –m “메시지”  git push origin main  mobaXterm 접속해서 git을 내려 받고 파일을 nginx 폴더에 넣는다.  sudo systemctl restart nginx |

