# **07.AWS EKS를 활용해 서버 배포하기**

1. **EC2에서 쿠버네티스를 활용했을 때 발생하는 AWS 예상 비용**

### **✅ EC2**

* EC2 인스턴스 (t4g.small) : 시간당 USD 0.0208 (24시간당 약 700원)  
    
   (비용이 걱정되시는 분들은 학습이 끝나자마자 인스턴스를 종료하시길 권장드립니다.)
* 데이터 전송 비용 : 1 GB당 0.1368 USD (1GB당 약 200원)  
    
   (실습 과정 동안 1GB 이하의 데이터만 전송합니다.)
* Public IPv4 비용 : 시간당 0.005 USD (24시간당 약 200원)

### **✅ RDS**

* RDS 인스턴스 (t4g.micro) : 시간당 USD 0.025 (24시간당 약 800원)  
    
   (프리티어일 경우 월 750시간까지 무료)
* 스토리지 비용 : GB-월당 0.131 USD (20GB-24시간당 약 200원)  
    
   (프리티어일 경우 20GB까지 무료)
* Public IPv4 비용 : 시간당 0.005 USD (24시간당 약 200원)

### **✅ ECR**

* 스토리지 비용 : GB-월당 USD 0.10 (1GB-24시간당 약 10원)  
    
   (실습 과정 동안 1GB 이하의 데이터만 저장합니다.)
* 데이터 전송 비용 : GB-월당 USD 0.126 (1GB-24시간당 약 6원)  
    
   (실습 과정 동안 1GB 이하의 데이터만 전송합니다.)

### **✅ ELB (Classic Load Balancer)**

* 사용 비용 : 시간당 0.025 USD (24시간당 약 800원)
* 처리한 데이터 비용 : GB당 0.008 USD (1GB-약 10원) (실습 과정 동안 1GB 이하의 데이터만 처리합니다.)

### **✅ EKS**

* EKS 클러스터 사용 비용 : 시간당 USD 0.10 (24시간당 약 3,500원) (비용이 걱정되시는 분들은 학습이 끝나자마자 EKS 클러스터를 종료하시길 권장드립니다.)

1. **AWS EKS를 남들보다 빠르게 익히려면?!**

### **✅ AWS EKS를 남들보다 빠르게 익히려면?!**

지금까지의 쿠버네티스 핵심 개념은 다 배웠다. AWS EKS라고 크게 다를 건 없다. 겁먹을 필요 없다. AWS EKS는 단순히 셋팅법만 익히면 나머지는 다 똑같다. EKS를 셋팅하면서 모든 옵션을 다 알 필요가 없다. 딱 필요하고 중요한 부분에 대해서만 알고 있으면 된다. 파레토의 법칙을 잊지 말자.

완벽하게 모든 기능을 익히려는 순간 학습의 효율은 엄청 떨어진다. 우린 내신 시험을 보는 게 아니다. 현업에서 잘 안 쓰는 지엽적인 내용은 뛰어넘을 수도 있어야 한다. 쿠버네티스를 실제 다루는 것에 집중하자.

1. **EKS란?**

### **✅ EKS(Elastic Kubernetes Service)란?**

EKS란 **AWS에서 쿠버네티스를 편하게 관리하고 사용할 수 있게 만든 AWS용 쿠버네티스**이다.

이와 비슷한 예로 MySQL과 같은 DB를 편하게 관리하고 사용할 수 있게 만든 서비스가 RDS이고,

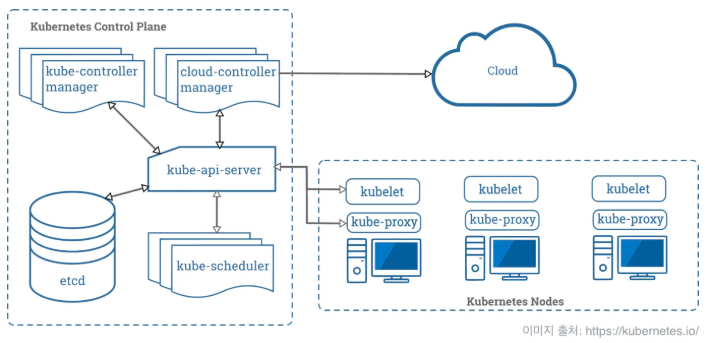
Redis와 같은 캐싱을 편하게 관리하고 사용할 수 있게 만든 서비스가 **ElastiCache**다.

### **✅ 현업에서도 EKS를 많이 사용할까?**

쿠버네티스를 직접 설치해서 관리하는 게 생각보다 손이 많이 간다. 따라서 현업에서는 쿠버네티스를 EC2와 같은 서버에 직접 설치해서 쓰지 않고, AWS에서 제공하는 EKS를 활용하는 경우가 많다.

1. **쿠버네티스와 EKS의 아키텍처 구조**

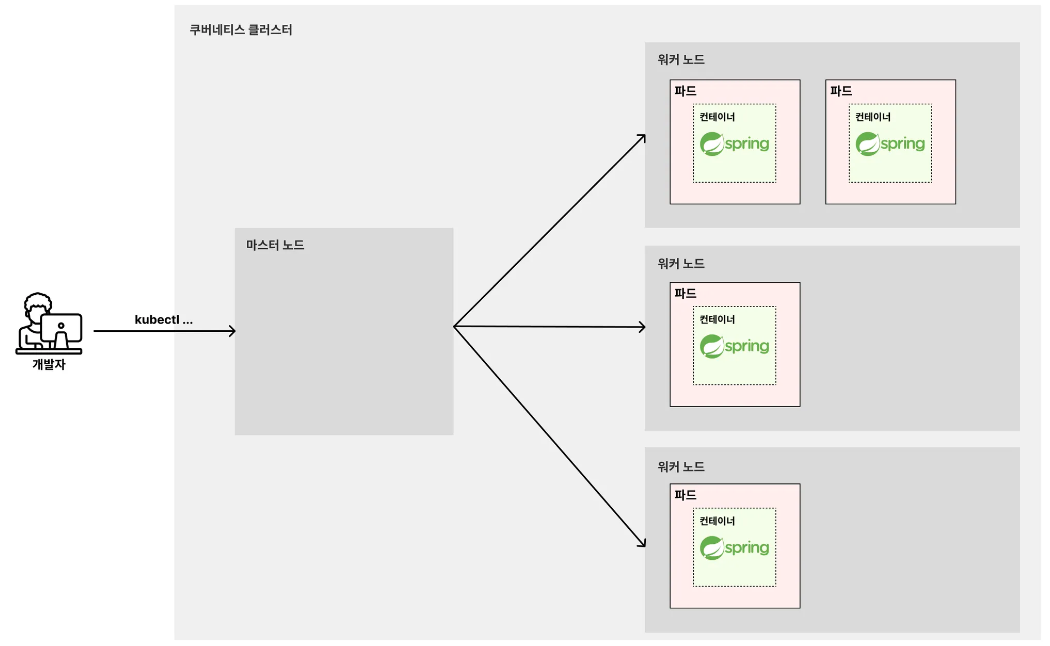
### **✅ 쿠버네티스의 복잡한 아키텍처 구조**



쿠버네티스를 입문하는 입장에서 위와 같은 복잡한 아키텍처 구조를 전부 이해할 필요는 없다. 그런데 대부분의 책과 강의의 초반부를 보면 일일이 다 설명하고 있다. **etcd**가 뭔지, **Control Plane**은 뭔지, **kube-scheduler**가 뭔지 하나하나 다 설명한다. 이렇게 공부하니까 쿠버네티스가 어렵게 느껴지고, 재미도 없고, 진도도 안 나가고, 이해도 안 되는 것이다. 이런 이유 때문에 이 강의의 초반에 쿠버네티스 아키텍처를 굳이 언급하지 않았다. 지금까지 쿠버네티스의 핵심 개념을 이해하는 데 크게 문제가 없었음을 느꼈을 것이다.

하지만 EKS를 다룰 때 아키텍처에서 기본적인 부분을 알아야 할 필요가 있어서 설명하고자 한다. 입문자 입장에서 알면 되는 부분만 간단화시켜서 살펴보자.

### **✅ 간단하게 표현한 쿠버네티스 아키텍처 구조**



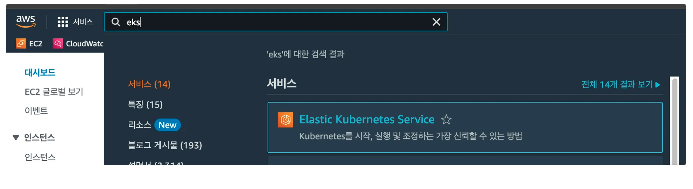
* **쿠버네티스 클러스터 :** 하나의 마스터 노드와 여러 워커 노드들을 한 묶음으로 부르는 단위
* **마스터 노드** : 쿠버네티스 클러스터 전체를 관리하는 서버
* **워커 노드** : 쿠버네티스의 파드를 실행시키는 서버

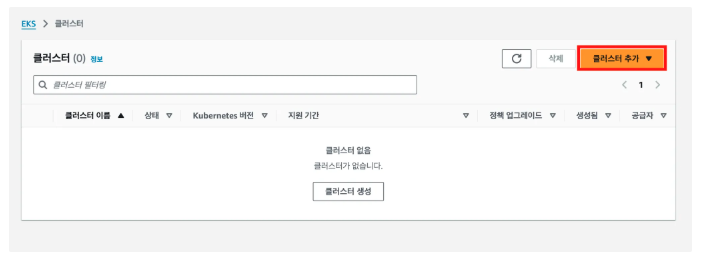
### **✅ EKS를 활용해 구성할 아키텍처 구조**

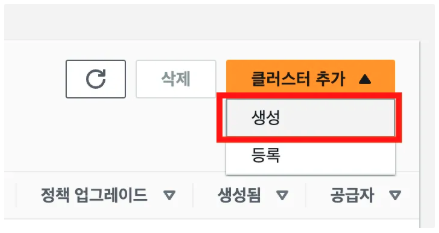
****

1. **EKS 클러스터 생성하기**

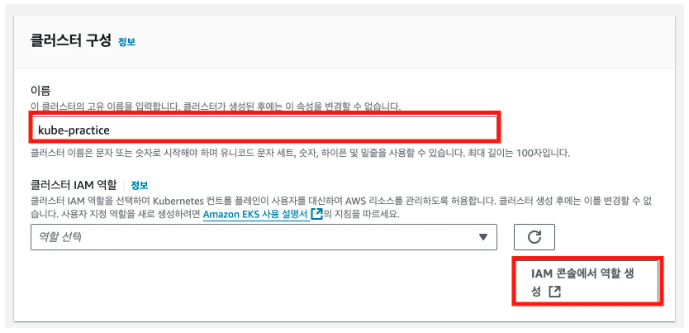
### **✅ EKS 클러스터 생성하기**

1. **EKS 서비스로 들어가기  
   **
2. **클러스터 추가하기**

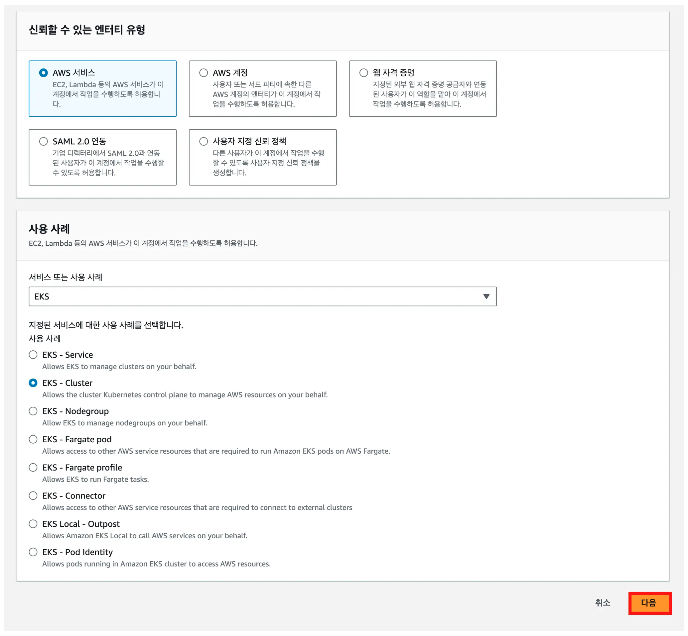
****

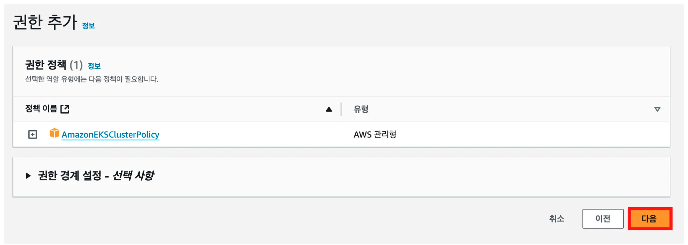
****

1. **클러스터 구성 셋팅하기**

****

1. **EKS 클러스터의 IAM Role 생성하기**

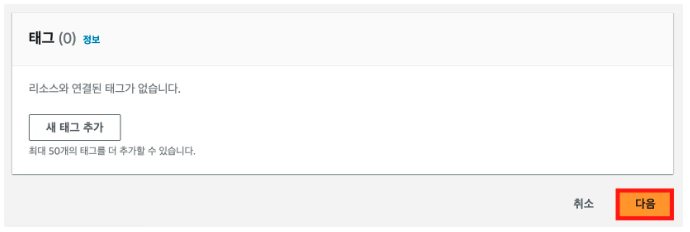
****

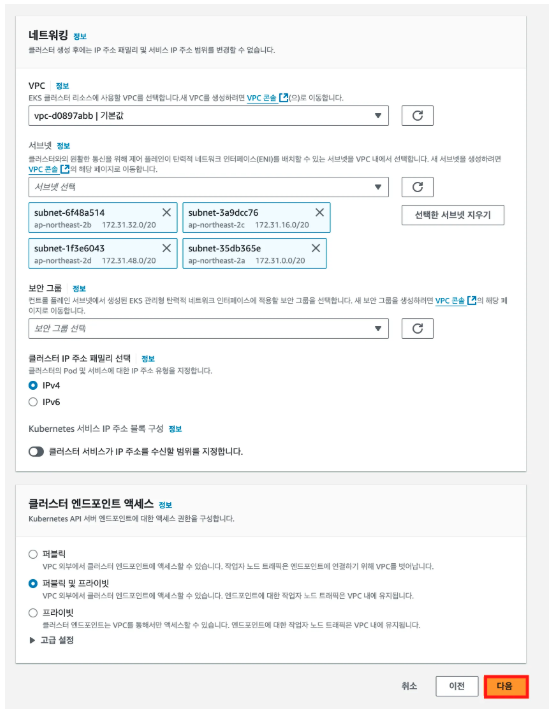
****

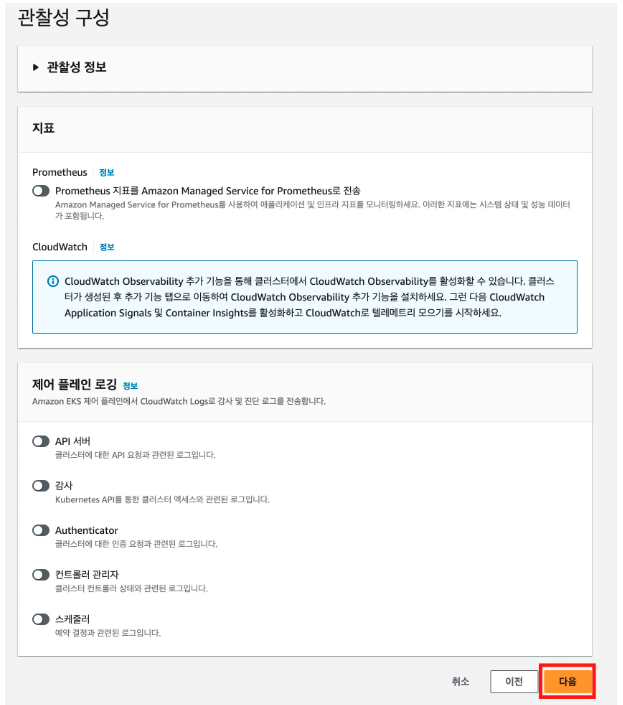
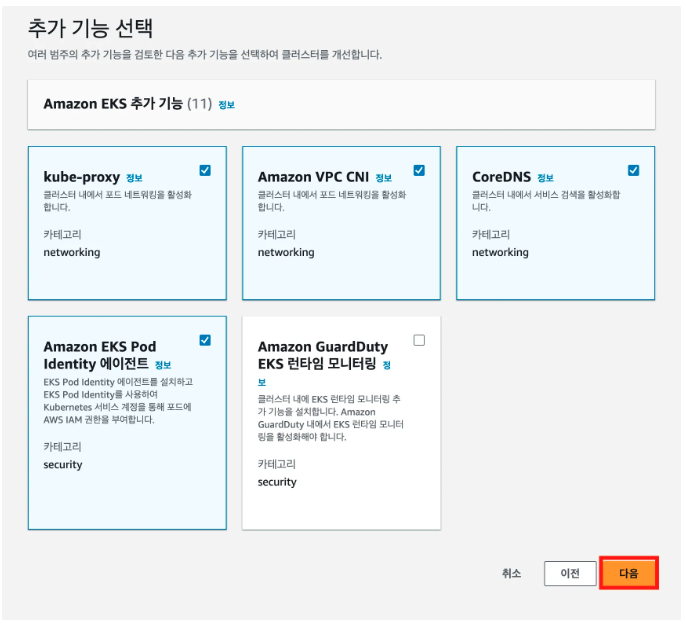
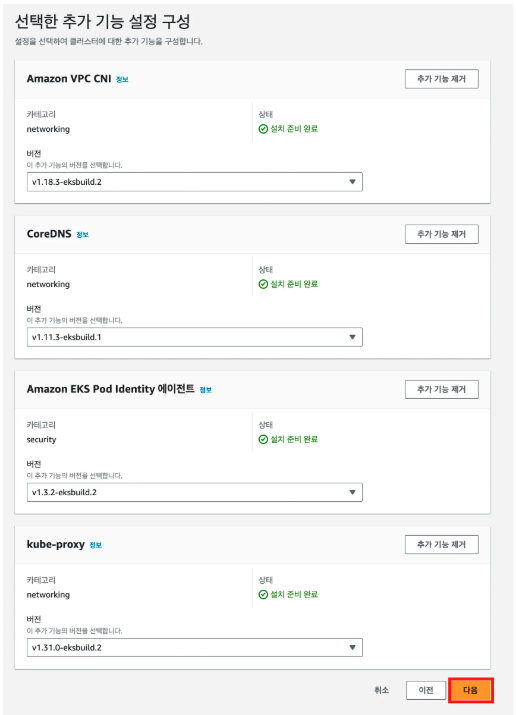
1. **방금 생성한 Role을 선택해 지정하기**

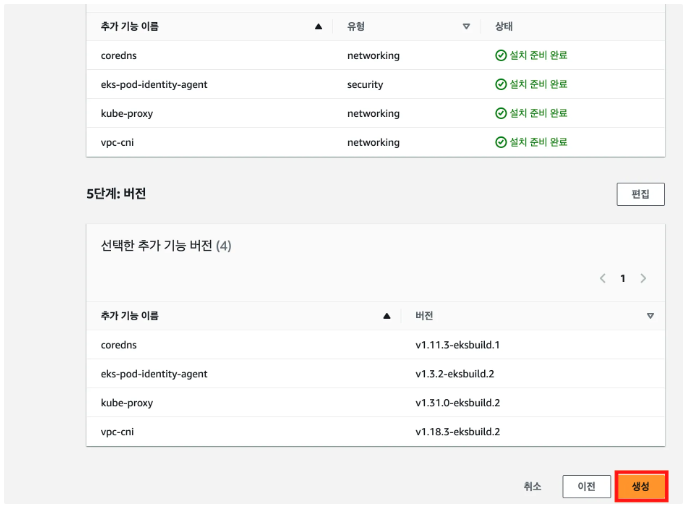
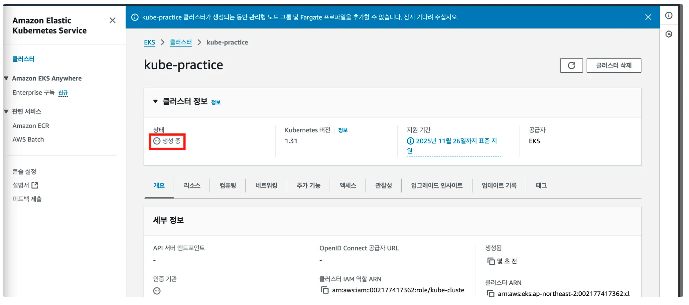
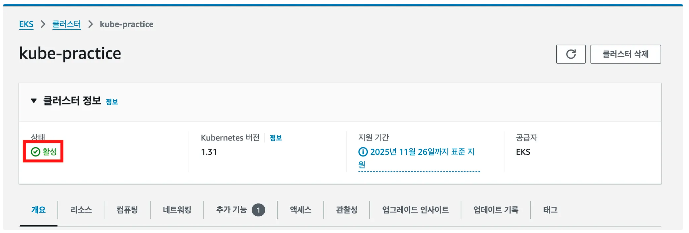
****

1. **나머지 옵션은 그대로 두고 다음 버튼 누르기**

**  
**

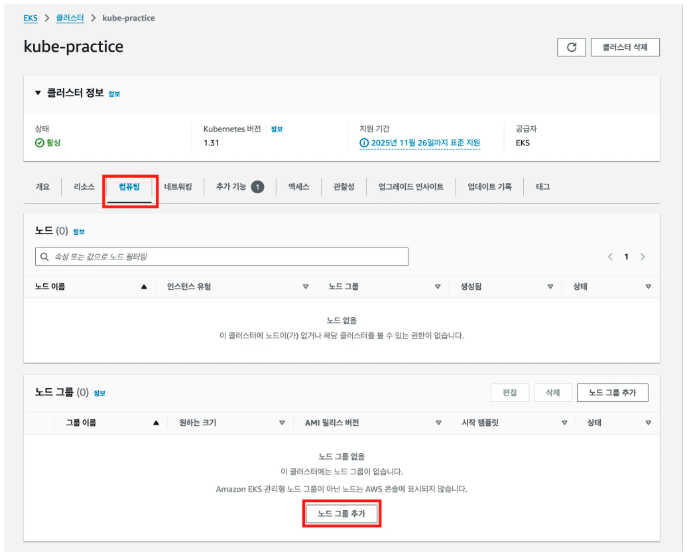
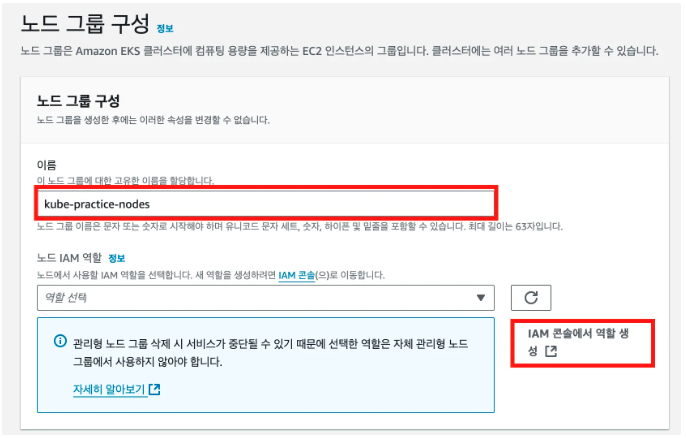
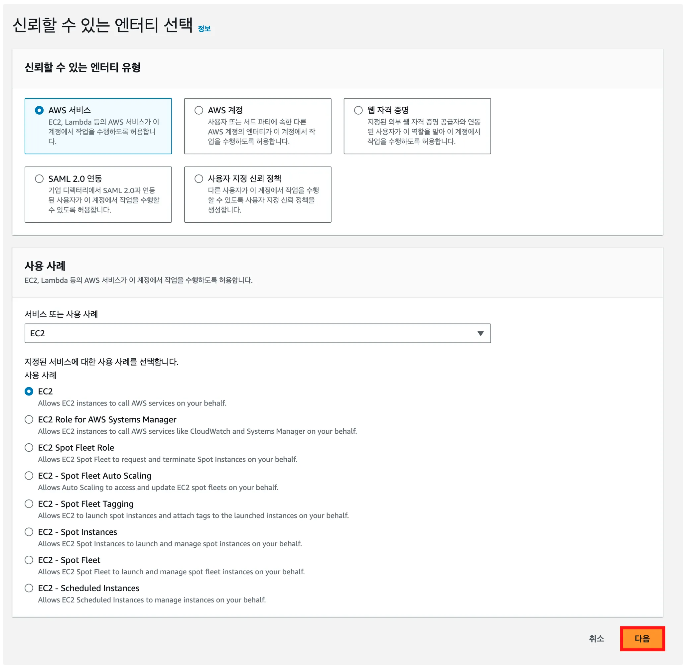
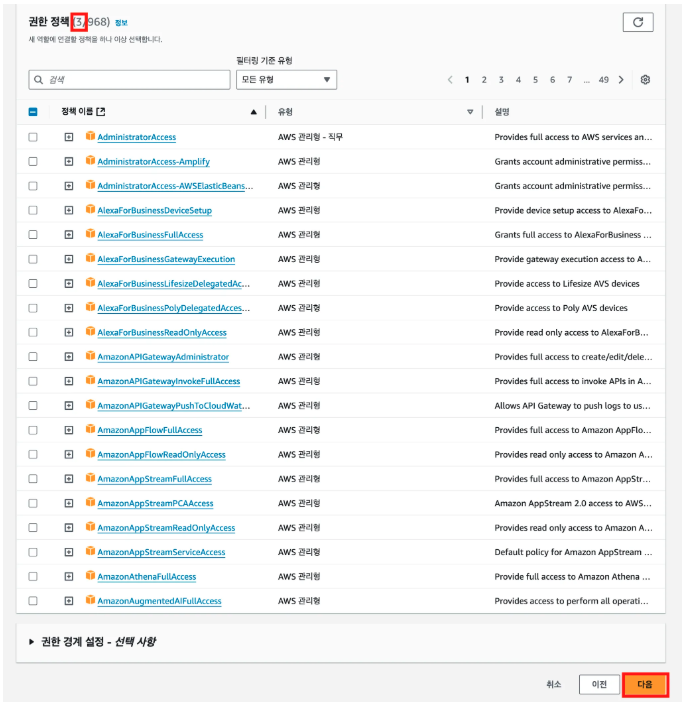
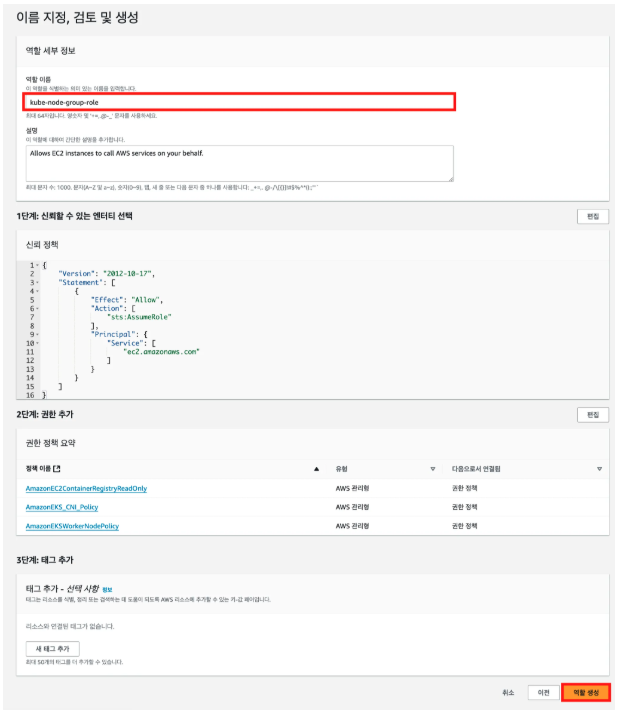
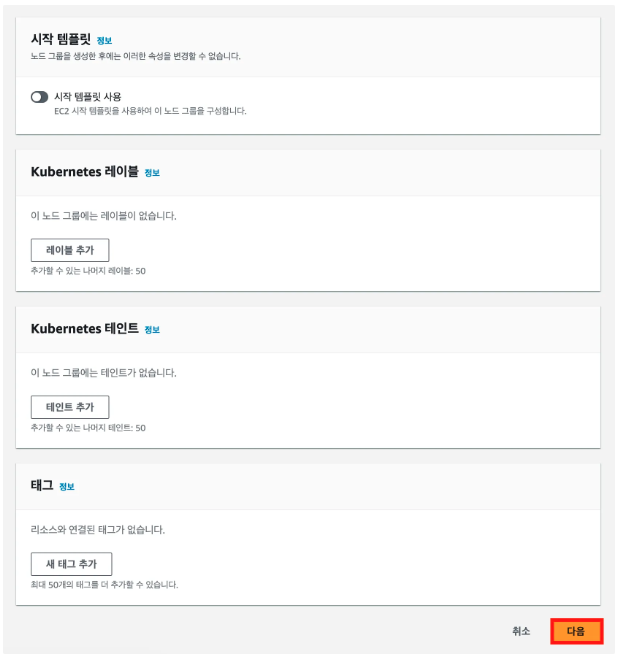
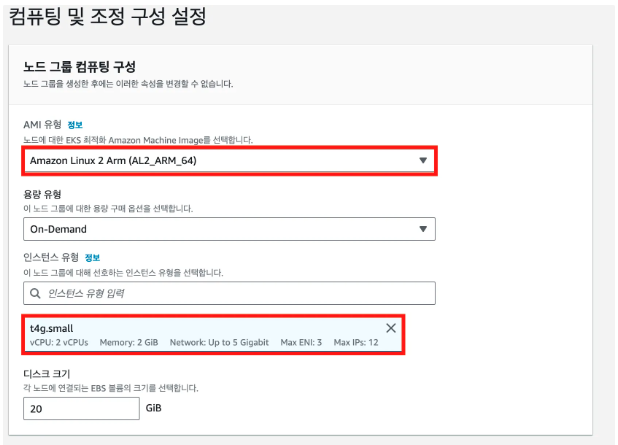
1. **다음 단계에서도 기본 옵션 그대로 두고 다음 버튼 누르기  
   **

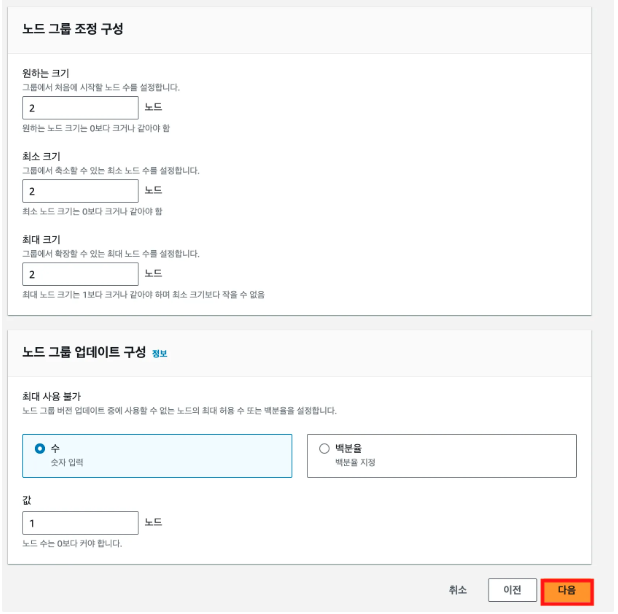
**  
  
**

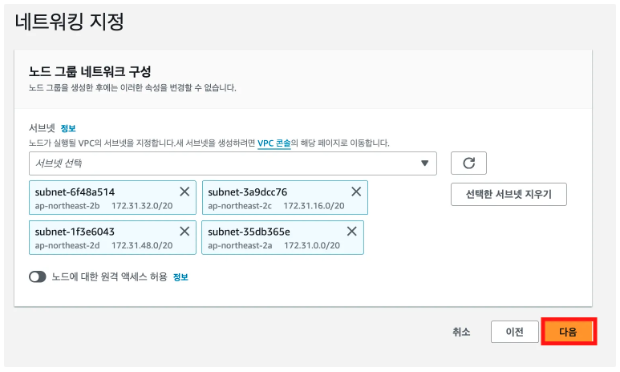
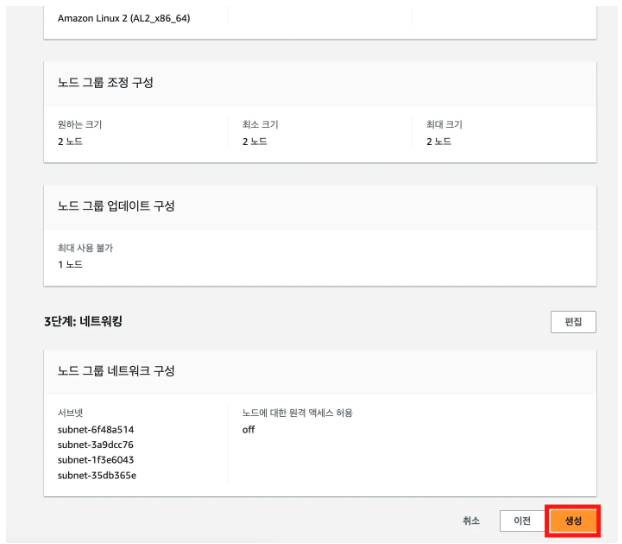
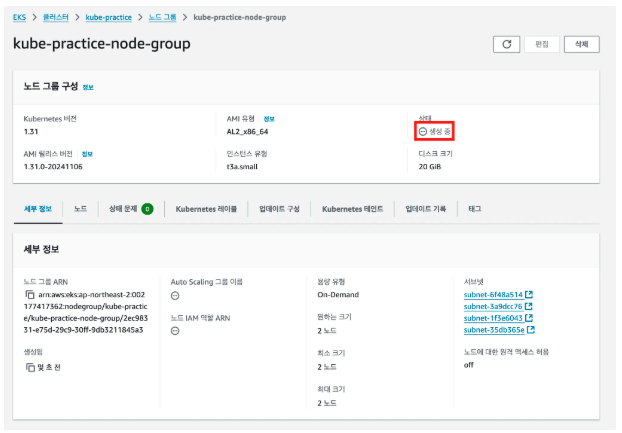
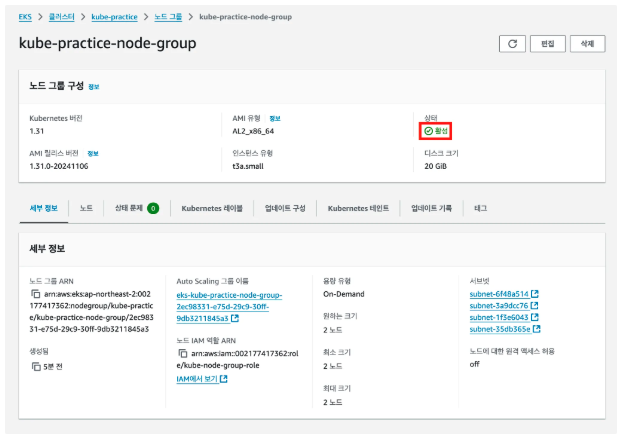
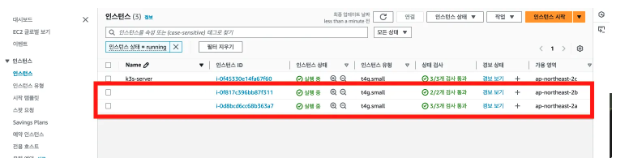
1. **‘검토 및 생성’ 페이지에서 생성 버튼 누르기  
   **
2. **생성이 완료될 때까지 기다리기  
     
    약 10분~15분 정도 걸린다.  
     
   **

**6. EKS 워커 노드 추가하기**

### **✅ EKS 워커 노드 추가하기**

1. **노드 그룹 추가하기**  
   
2. **노드 그룹 구성 셋팅하기**  
   
3. **EKS 노드 그룹의 IAM Role 생성하기**  
     
     
     
    → 이미 권한이 체크된 상태  
   
4. **방금 생성한 Role 선택해 지정하기**  
     
   
5. **컴퓨팅 및 조정 구성 설정하기**  
   



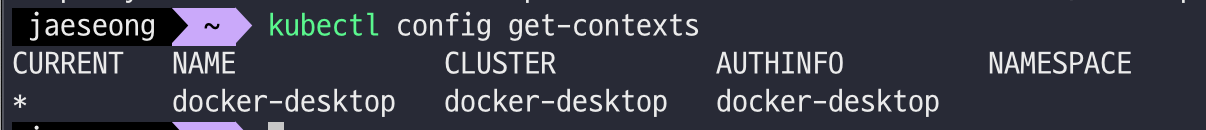
1. **나머지 옵션은 그대로 두기**  
     
   
2. **노드 그룹이 생성될 때까지 기다리기**  
     
     
     
    5분 정도 기다리면 활성화된다.
3. **EC2 인스턴스 확인하기**  
     
    EC2 인스턴스 페이지에 들어가면 새로운 EC2 인스턴스 2개가 생성되어 있는 걸 확인할 수 있다. EKS 클러스터에서 하나의 워커 노드(Worker Node)가 하나의 EC2 인스턴스에서 실행되는 구조이기 때문이다.  
   

**7. 로컬에서 EKS 클러스터 조정할 수 있게 셋팅하기**

#### **✅ 로컬에서 EKS 클러스터 조정할 수 있게 셋팅하기**

1. **현재 kubectl이 어떤 클러스터 환경에서 작동되고 있는 지 확인하기**

| $ kubectl config get-contexts |
| --- |

****

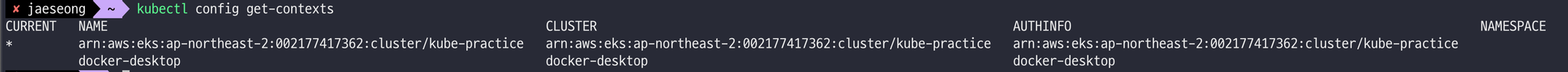
현재는 **kubectl**이 Docker Desktop의 쿠버네티스 클러스터를 작동시키고 있는 걸 알 수 있다 .

1. **kubectl에 EKS 클러스터 추가하기**

| # aws eks --rgeion ap-northeast-2 update-kubeconfig --name <EKS 클러스터 이름>  $ aws eks --region ap-northeast-2 update-kubeconfig --name kube-practice |
| --- |

1. **잘 적용 됐는지 확인하기**

| $ kubectl config get-contexts |
| --- |

****

**ALT**

**[참고]**

| # 다른 클러스터로 전환  $ kubectl config use-context <컨텍스트 이름>  # 특정 컨텍스트 삭제  $ kubectl config unset contexts.<컨텍스트 이름> |
| --- |

**8. EKS에 백엔드(Spring Boot) 서버 배포하기 (+ RDS, ECR)**

### **✅ EKS에 백엔드(Spring Boot) 서버 배포하기 (+ RDS, ECR)**

1. **매니페스트 파일 수정하기**

spring-deployment.yaml

| apiVersion: apps/v1  kind: Deployment  # Deployment 기본 정보  metadata:  name: spring-deployment # Deployment 이름  # Deployment 세부 정보  spec:  replicas: 3 # 생성할 파드의 복제본 개수  selector:  matchLabels:  app: backend-app # 아래에서 정의한 Pod 중 'app: backend-app'이라는 값을 가진 파드를 선택  # 배포할 Pod 정의  template:  metadata:  labels: # 레이블 (= 카테고리)  app: backend-app  spec:  containers:  - name: spring-container # 컨테이너 이름  image: **002177417362.dkr.ecr.ap-northeast-2.amazonaws.com/kube-ecr:2.0** # 컨테이너를 생성할 때 사용할 이미지  ports:  - containerPort: **8080** # 컨테이너에서 사용하는 포트를 명시적으로 표현  env:  - name: DB\_HOST  valueFrom:  configMapKeyRef:  name: spring-config  key: db-host  - name: DB\_PORT  valueFrom:  configMapKeyRef:  name: spring-config  key: db-port  - name: DB\_NAME  valueFrom:  configMapKeyRef:  name: spring-config  key: db-name  - name: DB\_USERNAME  valueFrom:  secretKeyRef:  name: spring-secret  key: db-username  - name: DB\_PASSWORD  valueFrom:  secretKeyRef:  name: spring-secret  key: db-password |
| --- |

spring-secret.yaml

| apiVersion: v1  kind: Secret  type: Opaque # 임의의 사용자 정의 데이터를 저장할 때 사용하는 타입  # Secret 기본 정보  metadata:  name: spring-secret # Secret 이름  # Key, Value 형식으로 값 저장  stringData:  db-username: **admin**  db-password: **password** |
| --- |

spring-config.yaml

| apiVersion: v1  kind: ConfigMap  # ConfigMap 기본 정보  metadata:  name: spring-config # ConfigMap 이름  # Key, Value 형식으로 설정값 저장  data:  db-host: **kube-database.coseefawhrzc.ap-northeast-2.rds.amazonaws.com**  db-port: "3306"  db-name: mydb |
| --- |

spring-service.yaml

| apiVersion: v1  kind: Service  # Service 기본 정보  metadata:  name: spring-service  # Service 세부 정보  spec:  type: **LoadBalancer** # Service의 종류  selector:  app: backend-app # 실행되고 있는 파드 중 'app: backend-app'이라는 값을 가진 파드와 서비스를 연결  ports:  - protocol: TCP # 서비스에 접속하기 위한 프로토콜  port: **80** # 외부에서 사용자가 요청을 보낼 때 사용하는 포트 번호  targetPort: 8080 # 매핑하기 위한 파드의 포트 번호  ~~nodePort: 30000~~ # 외부에서 사용자들이 접근하게 될 포트 번호 |
| --- |

* **NodePort** : 쿠버네티스 내부에서 해당 서비스에 접속하기 위한 포트를 열고 외부에서 접속 가능하도록 한다. ⇒ **들어오는 요청을 여러 Worker Node로 트래픽을 분산시키지 않는다.**
* **ClusterIP** : 쿠버네티스 내부에서만 통신할 수 있는 IP 주소를 부여. 외부에서는 요청할 수 없다.
* **LoadBalancer** : 외부의 로드밸런서(AWS의 로드밸런서 등)를 활용해 외부에서 접속할 수 있도록 연결한다. ⇒ **들어오는 요청을 여러 Worker Node로 트래픽을 분산시켜준다.**

1. **매니페스트 파일을 통해 오브젝트 생성하기**

| $ kubectl apply -f spring-secret.yaml  $ kubectl apply -f spring-config.yaml  $ kubectl apply -f spring-deployment.yaml  $ kubectl apply -f spring-service.yaml |
| --- |

1. **잘 생성 됐는지 확인하기**

| $ kubectl get secret  $ kubectl get configmap  $ kubectl get deployment  $ kubectl get pods |
| --- |

| $ kubectl get service |
| --- |

****

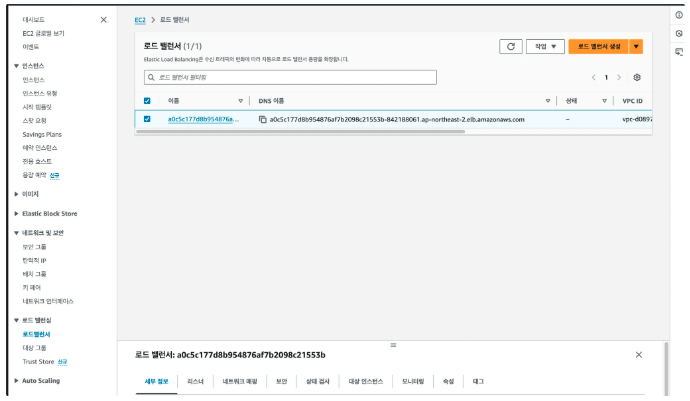
**Service**의 **Type**을 **LoadBalancer**로 했더니 외부에서 접속할 수 있는 주소가 주어졌다.

1. **Service의 주소로 접속해보기\**

****

****

1. **정말 로드밸런서가 생성 됐는지 확인하기**

****

### 

### **✅ 아키텍처 다시 한 번 짚어보기**

****

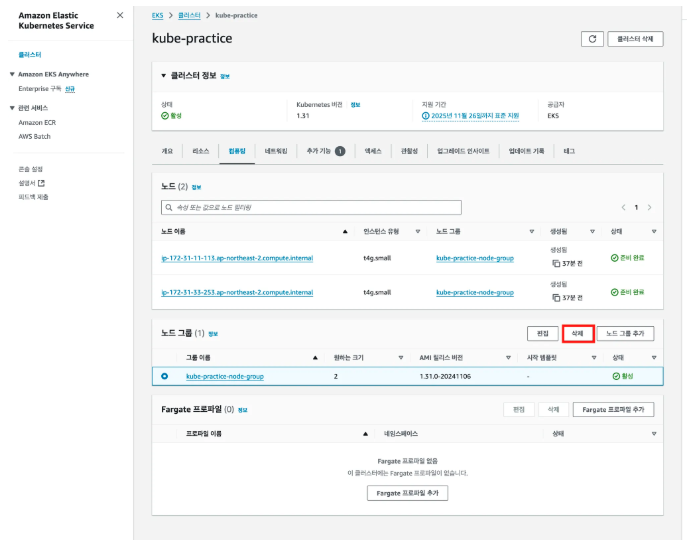
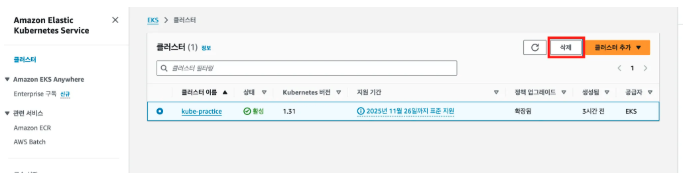
**9. 비용 나가지 않게 EC2, RDS, ECR, EKS 종료하기**

### **✅ 비용 나가지 않게 EKS 종료하기**

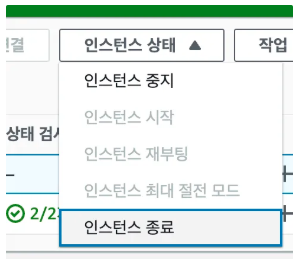
1. **실행 중인 오브젝트 종료하기**

실행 중인 파드가 있으면 EKS의 노드 그룹이 삭제되지 않는다.

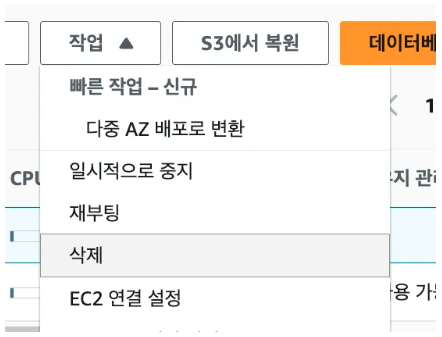
| $ kubectl delete all --all |
| --- |

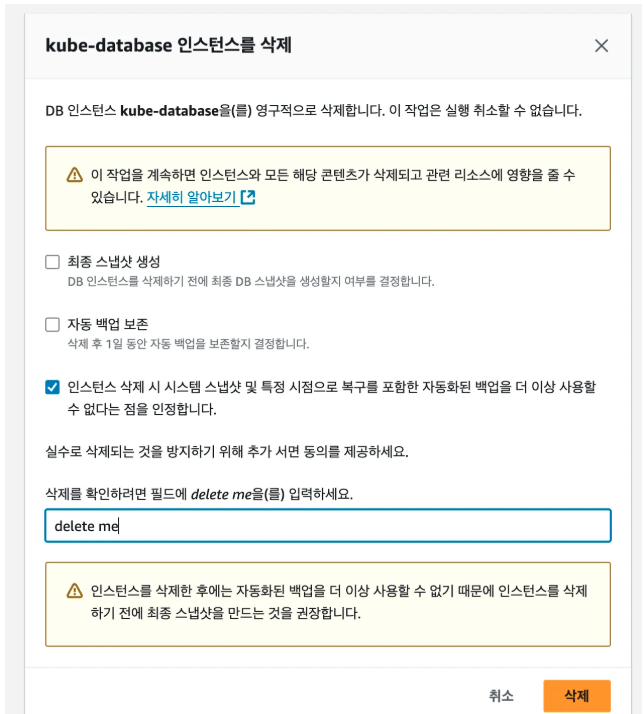
1. **EKS 노드 그룹 삭제하기  
   **
2. **EKS 클러스터 삭제하기**EKS의 노드 그룹이 삭제가 완료돼야만 EKS 클러스터를 삭제할 수 있다. EKS 노드 그룹이 삭제될 때까지 조금만 기다리자. ****

### **✅ 비용 나가지 않게 EC2 종료하기**



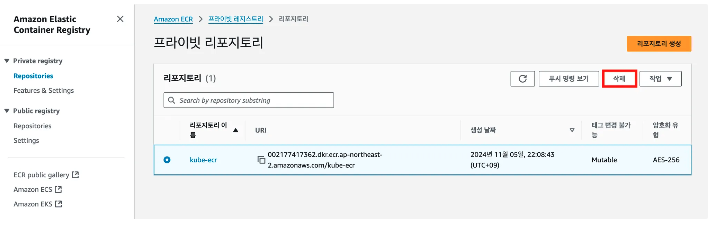
### **✅ 비용 나가지 않게 RDS 종료하기**

****

****

**❗** 최종 스냅샷 생성과 자동 백업 보존을 체크하면 비용이 나간다. 따라서 실제 운영용 데이터베이스가 아니라면 체크를 해제하고 삭제를 하자.

### **✅ 비용 나가지 않게 ECR 종료하기**

****

### **✅ 혹시나 비용 나가는 건 아닌지 체크하는 방법**

