



Running Containers on Amazon EKS

## 모듈 4:

### Amazon Elastic Kubernetes Service(Amazon EKS) 클러스터에 애플리케이션 배포



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.



#### | 강사용 노트

| 이 모듈을 완료하는 데 약 **50**분 정도 걸립니다.

| 이 모듈을 마치면 곧바로 실습 **2**를 진행합니다.

#### | 수강생용 노트

이 모듈은 **Amazon EKS** 클러스터에 애플리케이션 배포입니다.



Running Containers on Amazon EKS

## 모듈 4 개요

- 애플리케이션 배포 방법
- Amazon ECR 사용
- Helm을 사용하여 애플리케이션 배포
- 지식 확인
- 실습 2 준비



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.



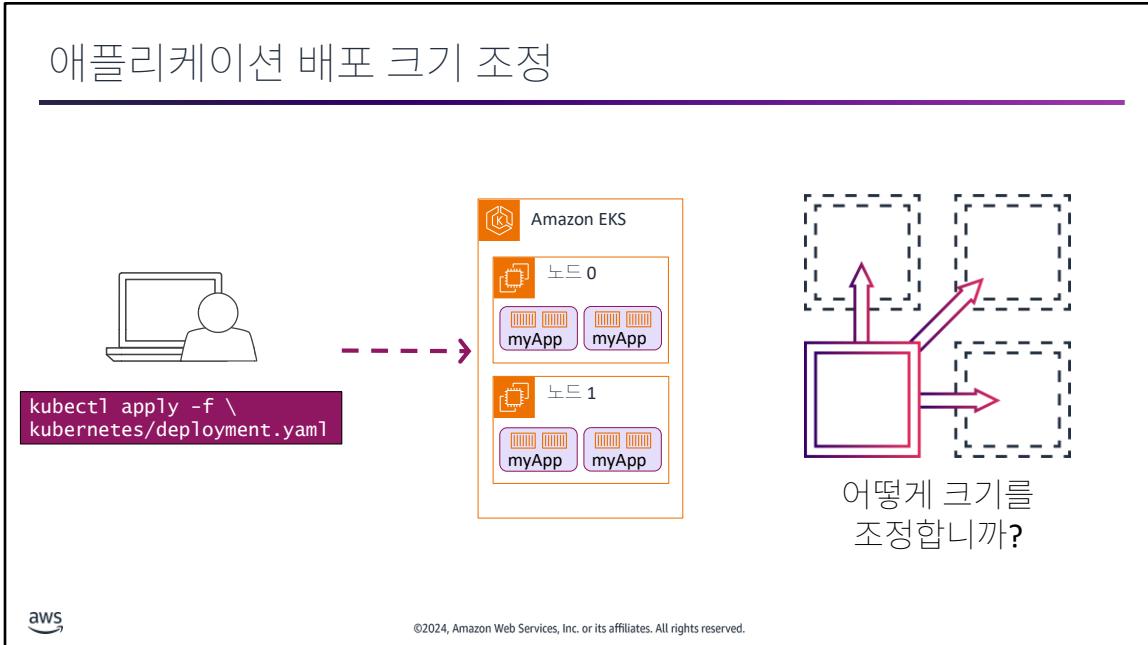
Running Containers on Amazon EKS

# 애플리케이션 배포 방법



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# 애플리케이션 배포 크기 조정



## ~ALT text

- ~ `kubectl`을 사용하여 애플리케이션을 배포하는 개발자.
- ~ **Deployment**는 2개의 노드에서 2개의 복제본을 실행합니다.

| 수강생용 노트

이 예는 Amazon Elastic Kubernetes Service(Amazon EKS)에 애플리케이션을 배포하는 `kubectl` 도구를 보여줍니다. 하지만 이 방법은 확장 가능하지 않습니다. 그렇다면 애플리케이션 **Deployment**를 어떻게 조정해야 합니까?

## Amazon EKS에서 애플리케이션 배포 크기 조정



컨테이너 이미지  
리포지토리를  
설정합니다.

1

패키지 관리자 또는  
애플리케이션 배포  
도구를 선택합니다.

2

CI/CD 파이프라인을  
자동화합니다.

3



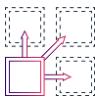
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

Amazon EKS 내에서 애플리케이션 Deployment를 확장 가능한 솔루션으로 만들려면  
다음 3단계를 완료해야 합니다.

먼저 추가 컨테이너 배포를 위해 관리하고 사용할 수 있는 컨테이너 이미지  
리포지토리를 설정합니다. 그런 다음 애플리케이션을 일관되게 배포하고 배포 오류를  
줄이며 롤백을 활성화할 수 있는 패키지 관리자 또는 애플리케이션 배포 도구를  
선택합니다. 마지막으로 컨테이너 이미지 리포지토리를 설정하고 패키지 관리자를  
구성한 후 지속적 통합(CI) 및 지속적 전달(CD) 파이프라인을 자동화하는 방법과 도구  
세트를 선택할 수 있습니다.

## Amazon EKS 애플리케이션 배포 자동화를 위한 도구 선택



Amazon Elastic  
Container Registry  
(Amazon ECR)

컨테이너 이미지  
리포지토리

1



패키지 관리자

2



AWS CodePipeline



ArgoCD

CI/CD 파이프라인  
자동화

3



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

이 과정에서는 실습 요구 사항에 가장 적합한 도구를 확인했습니다.

컨테이너 이미지 리포지토리에는 **Amazon Elastic Container Registry(Amazon ECR)**을, 패키지 관리자에는 **Helm**을, CI/CD 파이프라인 자동화에는 **AWS CodePipeline**을 사용합니다. **AWS CodePipeline**을 사용하는 '기존' CI/CD 파이프라인의 예도 살펴봅니다. 또한 **ArgoCD**와 **GitOps** 방법론을 추가하는 더욱 정교한 파이프라인의 예도 살펴봅니다.



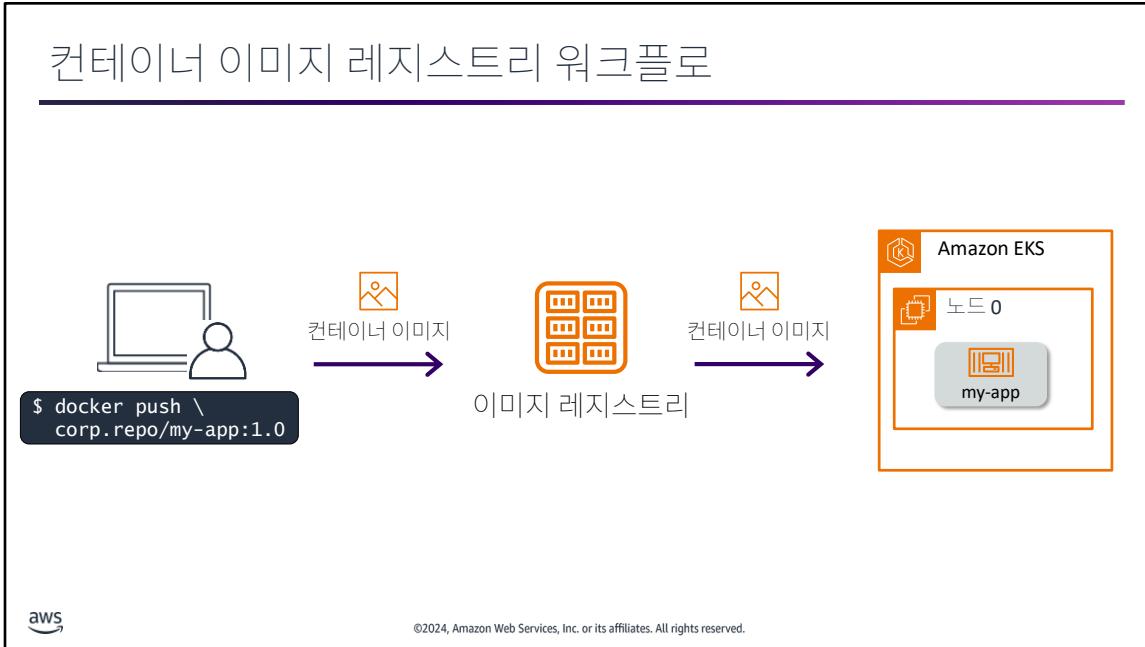
Running Containers on Amazon EKS

## Amazon ECR 사용



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# 컨테이너 이미지 레지스트리 워크플로



## ~Alt text

~ 컨테이너 이미지를 이미지 레지스트리에 푸시하는 개발자.

~ Amazon EKS 클러스터로 가져온 컨테이너 이미지.

~

## | 수강생용 노트

애플리케이션 개발 팀은 코드 리포지토리가 필요한 것처럼 컨테이너 이미지를 중앙에서 저장하고 제공할 수 있는 방법이 필요합니다. 컨테이너 이미지 레지스트리는 컨테이너 이미지의 신뢰할 수 있는 정보 소스가 됩니다. 이는 대상 환경(예: 개발, 테스트 또는 프로덕션)에 컨테이너식 애플리케이션을 배포하는 데 중심적인 역할을 제공합니다. 이점은 어디서든 동일한 코드를 실행할 수 있으면서도 코드가 모든 대상 환경에서 실행될 것을 확신할 수 있다는 것입니다.

이 다이어그램은 컨테이너 이미지가 개발자 워크스테이션에서 이미지 레지스트리로 이동하는 간단한 워크플로를 보여줍니다. **kubebuilder**은 컨테이너 런타임에게 컨테이너 이미지를 가져오도록 지시하며, 최종적으로 **Pod**에서 컨테이너가 실행됩니다. 조직이 **CI/CD** 자동화 파이프라인과 같은 워크플로를 선택하면 컨테이너 이미지 워크플로는 현저히 달라지지만 이미지 레지스트리의 일반 원칙은 여전히 유지됩니다.

## 컨테이너 이미지 저장

### 퍼블릭 레지스트리

- 퍼블릭 리포지토리
- 오픈 소스 소프트웨어
- 커뮤니티 지향적
- 공식 컨테이너 이미지
- 간단한 사용

### 프라이빗 레지스트리

- 프라이빗 리포지토리
- 독점적이고 권한이 있는 컨테이너 이미지
- 이미지 보안 검사
- 인증
- 역할 기반 액세스 제어



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

컨테이너 이미지를 레지스터리에 저장할 때 퍼블릭과 프라이빗 중에서 선택할 수 있습니다. 퍼블릭 레지스트리는 컨테이너 이미지용 퍼블릭 리포지토리를 제공합니다. 퍼블릭 리포지토리의 기본적 목적은 컨테이너 이미지를 누구와도 공유하는 것입니다. 오픈 소스 소프트웨어를 위한 커뮤니티 참여를 높이는 것이 목적이라면 퍼블릭 리포지토리가 가장 좋은 방법입니다. 또한 조직이 컨테이너가 포함된 제품과 서비스를 제공하는 경우 퍼블릭 리포지토리는 공식 컨테이너 이미지를 공유할 수 있는 방법을 고객에게 제공합니다.

프라이빗 레지스트리는 컨테이너 이미지용 프라이빗 리포지토리를 제공합니다. 프라이빗 레지스트리에는 일반적으로 조직의 보안 및 규정 준수 요구 사항(예를 들어 조직의 소프트웨어가 독점이고 공개 배포가 허용되지 않는 경우)을 충족하기 위한 기업 옵션이 있습니다. 프라이빗 레지스트리는 컨테이너 이미지에 대한 액세스를 권한이 있는 사용자로 제한할 수 있는 수단을 제공합니다. 또한 조직은 소프트웨어 개발 팀이 조직의 프라이빗 레지스트리의 권한이 있는 컨테이너 이미지(즉 ‘골든 이미지’)만을 개발 목적으로 사용하도록 허용할 수 있습니다. 프라이빗 레지스트리에 국한된 것은 아니지만 이미지 검사는 많은 조직에서 요구 사항입니다. 이미지 스캐닝은 취약성 탐지와 컨테이너 이미지에서 발견된 취약성 수정을 위한 보고를 제공합니다.

## Amazon ECR



Amazon ECR



- 컨테이너 이미지(Docker) 및 Open Container Initiative(OCI) 아티팩트를 위한 완전관리형 레지스트리

- 고가용성 및 확장성
- 안전함
  - 저장 데이터 암호화
  - 취약성 스캐닝(선택 사항)
- IAM 통합
- 이미지 서명

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

애플리케이션을 배포할 때 가장 먼저 고려해야 할 사항 중 하나는 컨테이너 이미지를 저장할 위치입니다. 컨테이너 이미지는 일반적으로 퍼블릭 또는 프라이빗 레지스트리에 저장됩니다. 이 모듈의 뒷부분에 설명되어 있는 것처럼 컨테이너 이미지를 구축하고 필요에 따라 가져오는 프로세스는 대개 자동화되어 있습니다.

레지스트리를 선택할 때는 다양한 선택지가 있습니다. 이 예에서는 **Amazon ECR**을 사용합니다. **Amazon ECR**은 완전관리형이고 다른 AWS 서비스와 잘 통합되며 안전합니다. 각 AWS 계정에는 기본적으로 프라이빗 및 퍼블릭 **Amazon ECR** 레지스트리가 각각 하나씩 제공됩니다. 프라이빗 레지스트리를 사용하면 레지스트리 내에 하나 이상의 리포지토리를 만들어 컨테이너 이미지를 저장할 수 있습니다. 생성하는 각 리포지토리는 **AWS Identity and Access Management(IAM)**에서 서로 다른 권한을 설정하여 액세스를 제어할 수 있습니다. **Amazon ECR Public**을 사용하면 누구나 전 세계에서 컨테이너 이미지를 검색하고 다운로드할 수 있도록 컨테이너 이미지를 저장, 관리, 공유, 배포할 수 있는 퍼블릭 레지스트리를 보유할 수 있습니다.

이 주제에서는 **Amazon ECR**에 컨테이너 이미지를 저장하는 데 중점을 둡니다. **Helm** 차트와 같은 OCI 아티팩트를 **Amazon ECR**에 저장할 수도 있습니다. 이 모듈의 뒷부분에서 **Helm** 차트에 대해 자세히 알아봅니다.

**Amazon ECR**에 대한 자세한 내용은 '**Amazon Elastic Container Registry**' (<https://aws.amazon.com/ecr/>)를 참조하십시오.

# ECR의 이미지 스캐닝

## 기본 스캐닝

- **Clair** 사용
  - 오픈 소스 이미지 스캐닝 솔루션
  - 푸시 또는 온디맨드로 호출
  - 24시간마다 한 번

## 고급 스캐닝

- **Amazon Inspector**와 통합
- 운영 체제 및 프로그래밍 언어 패키지 취약점을 모두 검사
- 이벤트를 **Amazon EventBridge**로 내보냄
- 일부 제한 사항 및 추가 비용



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 강사용 노트

| 시간이 허락한다면 이 슬라이드 다음에 **Amazon ECR**을 시연하면 좋습니다. 원하는 경우 다음 슬라이드를 숨김 해제하여 가이드로 사용할 수 있습니다.

### | 데모 단계

- |리포지토리를 생성합니다.
- |레지스트리에 **Docker** 클라이언트를 인증합니다.
- |이미지를 작성하고 태그를 지정합니다.
- |이미지를 리포지토리로 푸시합니다.
- |리포지토리에서 이미지를 사용합니다.

### | 추가 데모 노트:

| **Amazon ECR**에서 리포지토리를 구성하는 데 필요한 개략적인 프로세스입니다.

### | 수강생용 노트

컨테이너 이미지의 바이너리 및 애플리케이션 라이브러리에 취약성이 있거나, 시간이 지나면서 취약성이 발생할 수 있습니다. 이미지 스캐너로 이미지를 정기적으로 스캔하여 악용으로부터 보호하십시오. **Amazon ECR** 프라이빗 레지스트리에 저장된 이미지에 대해 기본 또는 고급 스캐닝을 구성할 수 있습니다.

기본 스캐닝은 오픈 소스 이미지 스캐닝 솔루션인 **Clair**를 사용합니다. **Clair**에 대한 자세한 내용은 GitHub의 **clair** 리포지토리(<https://github.com/quay/clair>)를 참조하십시오. 기본 스캐닝은 푸시 또는 온디맨드로 호출할 수 있습니다(24시간에 한 번).

고급 스캐닝은 **Amazon Inspector**와 통합되어 운영 체제 및 프로그래밍 언어 패키지 취약성을 모두 스캔합니다. 스캔 필터를 설정하여 자동 연속 스캐닝을 위해 구성한 리포지토리와 푸시 스캔을 위해 구성한 리포지토리를 선택합니다. 스캔 결과는 **Amazon ECR**와 **Amazon Inspector** 모두에서 확인할 수 있습니다. 결과에는 **Common Vulnerabilities and Exposure(CVE)** 데이터베이스를 바탕으로 발견된 소프트웨어 취약성이 심각도 기준으로 나열됩니다. 또한 **Amazon Inspector**는 초기 스캔이 완료되고 이미지 스캔 결과가 생성, 업데이트 또는 해결될 때 이벤트를 **Amazon EventBridge**에 내보냅니다. 고급 스캐닝과 관련해 제한 사항 및 추가 비용이 있습니다. 자세한 내용은 **Amazon ECR** 사용 설명서의 ‘고급 스캔 설명서’ (<https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning-enhanced.html>)를 참조하십시오.

**Amazon ECR**이 이미지를 스캔하고 나면 결과가 **Amazon EventBridge**의 **Amazon ECR**에 대한 이벤트 스트림에 로깅됩니다. **Amazon ECR** 콘솔에서도 스캔 결과를 볼 수 있습니다. 취약성이 높거나 심각한 이미지는 삭제하거나 다시 구축합니다. 배포된 이미지에서 취약성이 발생하는 경우 가능한 한 빨리 이미지를 교체하십시오.

**Amazon ECR** 구성의 세부 단계는 **Amazon ECR** 사용 설명서의 '프라이빗 리포지토리 생성' (<https://docs.aws.amazon.com/AmazonECR/latest/userguide/repository-create.html>)을 참조하십시오.

## 지식 확인 1

Amazon ECR에 대한 설명으로  
옳은 것은 무엇입니까? (2개  
선택)

보기	응답
A	각 AWS 계정에 단일 프라이빗 레지스트리와 단일 퍼블릭 레지스트리가 제공됩니다.
B	각 AWS 계정에 단일 프라이빗 리포지토리와 단일 퍼블릭 리포지토리가 제공됩니다.
C	AWS 계정 하나에 여러 개의 레지스트리를 생성할 수 있습니다.
D	레지스트리 하나에 여러 개의 리포지토리를 생성할 수 있습니다.
E	리포지토리 하나 안에 여러 개의 레지스트리를 생성할 수 있습니다.

## 지식 확인 1: 정답은 A와 D입니다.

Amazon ECR에 대한 설명으로 옳은 것은 무엇입니까? (2개 선택)

보기	응답
<b>A</b> 정답	각 AWS 계정에 단일 프라이빗 레지스트리와 단일 퍼블릭 레지스트리가 제공됩니다.
<b>B</b>	각 AWS 계정에 단일 프라이빗 리포지토리와 단일 퍼블릭 리포지토리가 제공됩니다.
<b>C</b>	AWS 계정 하나에 여러 개의 레지스트리를 생성할 수 있습니다.
<b>D</b> 정답	레지스트리 하나에 여러 개의 리포지토리를 생성할 수 있습니다.
<b>E</b>	리포지토리 하나 안에 여러 개의 레지스트리를 생성할 수 있습니다.

 ©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

정답은 **A**와 **D**입니다. 레지스트리에는 리포지토리가 포함되어 있습니다. 각 AWS 계정에는 프라이빗 레지스트리와 퍼블릭 레지스트리가 하나씩 있습니다. 이러한 레지스트리 안에 여러 리포지토리를 생성할 수 있습니다.



Running Containers on Amazon EKS

## Helm을 사용하여 애플리케이션 배포



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## Helm: Kubernetes 패키지 관리자



Helm



- 표준화되고 재사용 가능한 템플릿 생성
- 배포 오류 제거
- 애플리케이션 버전 관리
- 현재 위치 업그레이드 수행

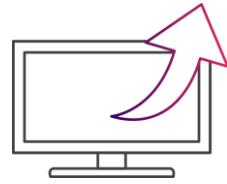
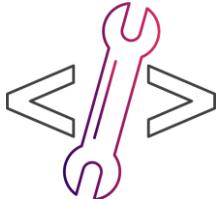
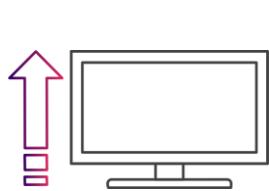
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

Helm은 Kubernetes의 패키지 관리자입니다. Kubernetes 클러스터에 애플리케이션을 설치하고 관리하는 데 도움이 됩니다. Kubernetes 자체와 마찬가지로 Helm은 Cloud Native Computing Foundation(CNCF)의 출입 단계 프로젝트입니다.

Helm에 대한 자세한 내용은 Helm 웹 사이트(<https://docs.helm.sh/>)를 참조하십시오.

## 일반적인 Helm 태스크



애플리케이션 설치/제거

애플리케이션 업그레이드

애플리케이션 게시



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

다음은 각 Helm 태스크에 대한 대략적인 단계입니다.

#### 애플리케이션 설치

1. **localhost**에 **Amazon EKS** 차트 리포지토리를 추가합니다.
2. 설치할 애플리케이션을 검색합니다.
3. 애플리케이션을 설치합니다.

#### 설치된 애플리케이션 업그레이드

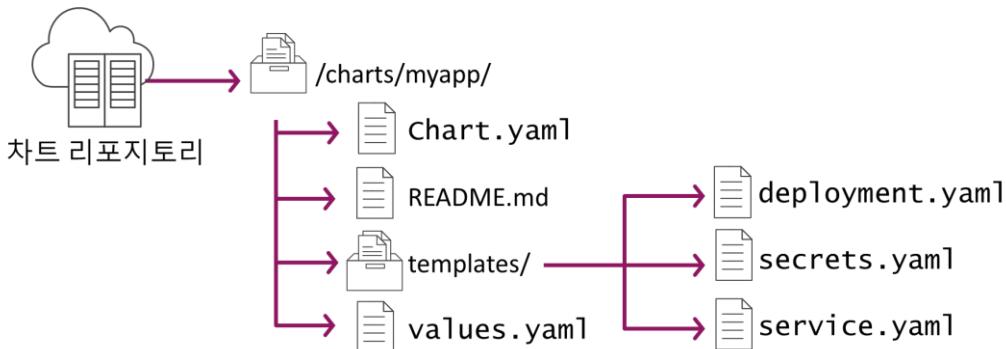
1. 업그레이드를 실행합니다.
2. 필요한 경우 롤백합니다.

#### 애플리케이션 게시

1. 새 차트를 생성합니다.
2. 샘플 파일을 삭제합니다.
3. 새 **Chart.yaml** 파일을 생성합니다.
4. **Deployment** 및 **Service manifest**를 위한 디렉터리를 생성합니다.
5. 매니페스트를 복사합니다.
6. 템플릿을 테스트합니다.
7. 차트를 배포합니다.

## Helm 차트 분석

Helm 차트는 애플리케이션에 필요한 모든 객체를 패키징합니다.



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### ~ALT text

~ 다양한 파일과 디렉터리를 표시하는 **Helm** 차트입니다. 자세한 내용은 노트에 있습니다.

#### | 수강생용 노트

**Helm** 차트는 애플리케이션을 설명하고, 반복 가능한 애플리케이션 설치 지침을 제공하며, 단일 인증 원본 역할을 합니다. 차트를 사용하면 애플리케이션을 더 쉽게 설치, 업그레이드 또는 제거할 수 있습니다. 차트는 파일 자체가 아니라 **Helm**에 애플리케이션 설치 방법을 알려주는 특정 하위 디렉터리와 파일을 포함하는 디렉터리입니다. 차트는 일반적으로 리포지토리에 저장되고 배포를 위해 로컬 호스트에 복사됩니다.

이 예에서는 다음 구성 요소가 포함된 기본 차트를 보여줍니다.

- **Chart.yaml** - 차트에 대한 정보가 들어 있는 **YAML** 파일입니다.
- **LICENSE** - 차트에 대한 라이선스가 포함된 일반 텍스트 파일(선택 사항)입니다.
- **README.md** - 사람이 읽을 수 있는 **README** 파일(선택 사항)입니다.
- **values.yaml** - 차트의 기본 구성 값입니다. 템플릿 디렉터리의 매니페스트는 이 파일에서 값을 가져오므로 매니페스트 템플릿을 지속적으로 업데이트할 필요가 없습니다.
- **values.schema.json** - **values.yaml** 파일에 구조를 부과하기 위한 **JSON** 스키마(선택 사항)입니다.
- **charts/** - 이 차트가 종속된 모든 차트가 포함된 디렉터리입니다.

- **crds/** - 사용자 정의 리소스 정의입니다.
- **templates/** - **values**의 값과 결합되면 유효한 **Kubernetes** 매니페스트 파일을 생성하는 템플릿의 디렉터리입니다.
- **templates/NOTES.txt** - 간단한 사용 노트가 포함된 일반 텍스트 파일(선택 사항)입니다.

## Helm 차트 템플릿 작성(단순화됨)

templates/pod.yaml

```
...  
metadata:  
  name: {{ .Release.Name }}  
  namespace: {{ .Release.Namespace }}  
spec:  
  containers:  
    - name: {{ .values.name }}  
      image:  
        {{ .values.image }}:{{ .values.tag }}  
      ports:  
        - containerPort: {{ .values.port }}
```

values.yaml

```
# container values  
name: my-app  
image: nginx  
tag: 0.2  
  
# port values  
port: 8080
```



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

~Dev notes

~ Reference: [https://helm.sh/docs/chart\\_template\\_guide/builtin\\_objects/](https://helm.sh/docs/chart_template_guide/builtin_objects/)

~

| 강사용 노트

| - 필요한 경우 점 표기법이 YAML 사전 매핑을 따른다는 점을 설명하십시오.

| - Helm의 .Release 및 .Values의 기능에 대해서도 설명하십시오.

([https://helm.sh/docs/chart\\_template\\_guide/builtin\\_objects/](https://helm.sh/docs/chart_template_guide/builtin_objects/))

| - 논의를 DRY 원칙을 따르는 Helm 제어 구조로 확장할 수 있습니다.

| - 예를 들어 {{- with .Values }} 및 {{- end }} 추가가 pod.yaml 파일의 템플릿 지시문에 어떤 영향을 미치는지 논의합니다.

~

| 수강생용 노트

Helm은 YAML 매니페스트에 대한 템플릿 작성 기능을 제공합니다. 왼쪽 코드 블록은 Pod YAML 매니페스트의 각 키-값 쌍에 템플릿 지시문을 사용하는 간단한 예입니다. 오른쪽 코드 블록은 변수에 사용할 값을 보여줍니다({{ .Release.name }} 변수 제외).

'Release' 객체는 릴리스 자체를 설명하며 그 안에 여러 객체가 있습니다. 이 예에서 'Release.Name'은 release라는 이름을 나타내고 'Release.Namespace'는 릴리스될 이름입니다(매니페스트가 재정의되지 않는 경우). 내장 객체에 대한 자세한 내용은 [https://helm.sh/docs/chart\\_template\\_guide/builtin\\_objects/](https://helm.sh/docs/chart_template_guide/builtin_objects/)를 참조하십시오.

기본적으로 Helm은 템플릿 지시문을 values.yaml 파일에 정의된 해당 값으로 바꿉니다. 그러나 사용자 지정 값은 명령줄 파라미터 또는 사용자 제공 values.yaml 파일로 전달할 수 있습니다.

여기에 표시된 예는 **Helm** 템플릿 언어를 간략히 소개하기 위한 것입니다. 차트 템플릿 생성에 대해 자세히 알아보려면 **Helm** 웹 사이트에서 ‘**Chart Template Developer’s Guide**’([https://helm.sh/docs/chart\\_template\\_guide/](https://helm.sh/docs/chart_template_guide/))를 참조하십시오.

## Helm 차트 액세스



Artifact Hub



Amazon ECR



Amazon S3



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 강사용 노트

| 시간이 허락한다면 이 슬라이드 다음에 ‘**Helm**을 사용하여 애플리케이션 배포’ 데모를 진행하는 것이 좋습니다. 원하는 경우 다음 슬라이드를 숨김 해제하여 가이드로 사용할 수 있습니다.

### | 데모 단계:

- | **Helm**을 사용하여 애플리케이션을 설치합니다.
- | 애플리케이션을 업그레이드합니다.
- | 새 차트를 구축합니다.
- | 차트를 배포합니다.

### | 데모 노트:

| 오픈 소스 패키지 관리자로서 **Helm**의 역할을 강조하십시오.

- | **Helm**은 많은 오픈 소스 애플리케이션을 자동으로 배포할 수 있습니다.
- | **Helm**은 ‘마법’이 아닙니다. **Helm** 모의 실행을 통해 **Helm**이 수행하는 프로세스를 확인할 수 있습니다.
- | 이는 **Helm** 내부에서 벌어지는 프로세스에 대한 수강생의 궁금증을 해소하기 위한 것입니다.

### | 수강생용 노트

**Helm** 차트 패키지는 차트 리포지토리에 저장됩니다. **Helm** 클라이언트를 다양한 차트 리포지토리에서 검색 및 배포하도록 구성할 수 있습니다. **Helm** 클라이언트는 **Helm** 차트 리포지토리에 푸시할 수도 있습니다.

**Artifact Hub**는 개발자가 자신의 차트 리포지토리를 다른 개발자에게 공개할 수 있는 퍼블릭 **Helm** 차트 리포지토리의 중앙 목록입니다. **Artifact Hub**에는 차트 패키지가 아니라 **Helm** 차트 리포지토리에 액세스하는 방법에 대한 세부 정보가 있습니다. **Artifact Hub**에 대한 자세한 내용을 알아보려면 공식 웹 사이트 (<https://artifacthub.io>)를 방문하십시오.

**Helm** 차트 사용에 대한 보안 요구 사항으로 인해 퍼블릭 리포지토리를 사용하지 못할 수도 있습니다. 이 때문에 여러 상업용 컨테이너 이미지 레지스트리도 프라이빗 **Helm** 차트 리포지토리를 제공합니다. 예를 들어 **Amazon ECR**을 사용하여 **Helm** 차트 패키지를 프라이빗 리포지토리에 저장할 수 있습니다. 이 옵션에 대해 자세히 알아보려면 **Amazon ECR** 사용 설명서의 ‘**Amazon ECR**과 **Amazon EKS**에서 호스팅되는 **Helm** 차트 설치’ ([https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR\\_on\\_EKS.html#using-helm-charts-eks](https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_EKS.html#using-helm-charts-eks))를 참조하십시오.

**Amazon Simple Storage Service(Amazon S3)**도 차트 리포지토리에서 **Helm** 차트 패키지를 호스팅하는 옵션을 제공할 수 있습니다. 워크플로 일관성을 위해 **Helm** 차트 패키지에 다른 도구를 채택하는 대신 **Amazon S3**를 사용하도록 선택할 수도 있습니다. 그러면 **Helm** 차트 패키지 관리를 컨테이너 이미지 관리에서 분리하는 유연성을 유지하면서도 **Amazon S3**의 모든 이점을 누릴 수 있습니다. **Helm** 차트에 **Amazon S3**를 사용하는 방법에 대해 자세히 알아보려면 **AWS** 권장 가이드 패턴의 ‘**Amazon S3**에서 **Helm v3** 차트 리포지토리 설정’ (<https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/set-up-a-helm-v3-chart-repository-in-amazon-s3.html>)를 참조하십시오.

## 지식 확인 2

Helm 차트에 대한 정보가 포함된 필수 파일은 무엇입니까?

보기	응답
A	Contents.yaml
B	Config.yaml
C	Templates.yaml
D	Chart.yaml



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 지식 확인 2: 정답은 D입니다.

Helm 차트에 대한 정보가 포함된 필수 파일은 무엇입니까?

보기	응답
A	Contents.yaml
B	Config.yaml
C	Templates.yaml
D 정답	Chart.yaml

 ©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | 수강생용 노트

정답은 D입니다. **chart.yaml** 파일에는 차트에 대한 주요 정보가 포함되어 있습니다. 이 정보에는 차트에서 사용 중인 **API** 버전, 차트 유형, 차트와 애플리케이션의 버전 번호가 포함됩니다.

## 모듈 요약



이 모듈에서 학습한 내용:

- 수동 배포 방법과 자동 배포 방법을 구별
- 이미지 레지스트리 워크플로를 설명
- ECR 리포지토리 생성
- 컨테이너 이미지를 ECR로 푸시
- 레지스트리에서 이미지 사용
- Helm을 사용하여 애플리케이션 배포

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| 강사용 노트

| 수강생용 노트



Running Containers on Amazon EKS

## 실습 2: Helm 및 Amazon S3를 사용하여 애플리케이션 배포



©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| 강사용 노트

| 이 실습은 완료하는 데는 약 **30**분 정도 걸립니다.

| 수강생용 노트

실습 2에서는 애플리케이션을 **Amazon EKS** 클러스터에 배포합니다.

## 실습 2



aws

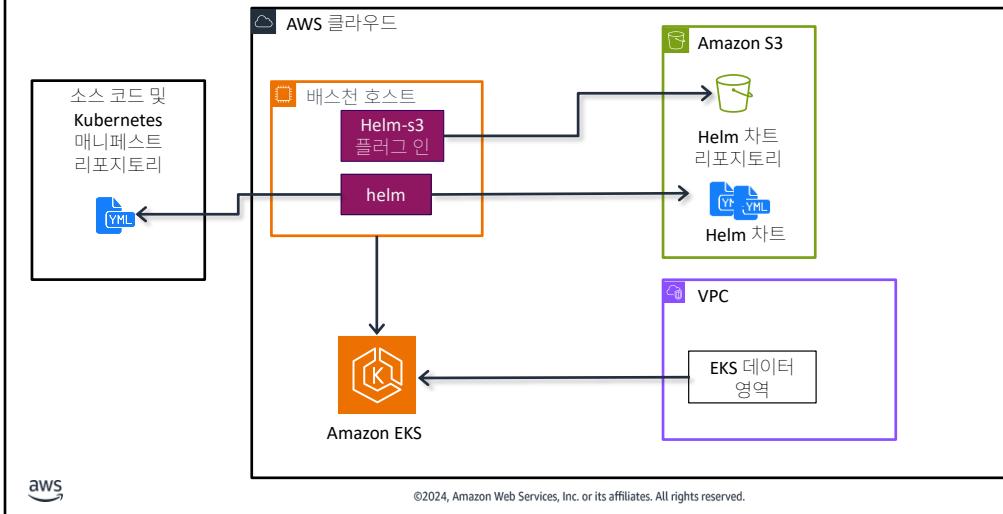
**Helm** 및 **Amazon S3**를 사용하여 애플리케이션 배포

이 실습에서는 다음 과제를 수행합니다.

- Amazon S3을 Helm 리포지토리로 구성
- S3 Helm 리포지토리에 차트를 패키징 및 로드
- Helm을 사용하여 애플리케이션 배포
- 차트 및 values.yaml 파일 검토

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 실습 2 아키텍처 다이어그램





감사합니다.



수정 사항이나 피드백 또는 기타 질문이 있으십니까?

<https://support.aws.amazon.com/#/contacts/aws-training>에서 문의해 주십시오. 모든  
상표는 해당 소유자의 자산입니다.

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| 강사용 노트

| 수강생용 노트