



Running Containers on Amazon EKS

모듈 8:

Amazon EKS에서 스토리지 관리



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



~Dev Notes

~This module was pulled from Module 8 of the Amazon EKS for Developer course with some updates made.
~

| 강사용 노트(본 모듈: 65분)

| 실습 5는 이 모듈 Amazon EKS의 영구 스토리지의 끝부분에 있습니다.



Running Containers on Amazon EKS

모듈 8 개요

- 스토리지 설계 패턴
- **Kubernetes**의 영구 스토리지
- AWS 스토리지 서비스를 사용하는 영구 스토리지
- **Secret** 관리
- 실습 5 준비



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

| 수강생용 노트

이 모듈에서는 **Amazon EKS**에서 실행하는 애플리케이션에 사용 가능한 **AWS** 스토리지 옵션에 대해 자세히 알아봅니다.



Running Containers on Amazon EKS

스토리지 설계 패턴



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

| 수강생용 노트

이 섹션에서는 스테이트풀 워크로드와 스테이트리스 워크로드를 비교하고 임시 Pod에 영구 스토리지를 사용하는 패턴을 검토합니다.

애플리케이션 아키텍처 유형

스테이트리스

- 복제본이 동시에 실행되고 빠른 교체가 가능
- 애플리케이션 데이터가 일시적
- 임시 스토리지가 적합

스테이트풀

- 각 인스턴스의 고유한 자격 증명
- 애플리케이션에 데이터 내구성 필요
- 영구 스토리지가 적합



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

| 수강생용 노트

스테이트리스 애플리케이션은 이전 상호 작용에 대한 지식이 필요 없고 세션 정보를 저장하지 않는 애플리케이션입니다. 과거 트랜잭션에 대한 저장된 지식 또는 참조가 없습니다. 각 트랜잭션은 처음인 것처럼 수행됩니다. 필요한 경우 애플리케이션의 기능이 저장된 상태에 의존하지 않으므로 임시 스토리지가 트랜잭션을 수행하는데 적합합니다. 데이터 지속성이 요구 사항인 경우 스테이트리스 애플리케이션이 데이터 지속성을 위해 설계된 스테이트풀 서비스(예: **Amazon Aurora**와 같은 데이터 스토어)로 데이터를 전송합니다.

스테이트풀 애플리케이션은 다음 세션에서 사용하기 위해 한 세션의 활동 데이터를 저장합니다. 이 데이터를 애플리케이션의 상태라고 합니다. 이는 의도된 기능이므로 애플리케이션에 데이터 내구성이 필요합니다. 스테이트풀 애플리케이션의 규모를 조정하려면 각 인스턴스가 해당 세션 상태에 대한 고유한 **ID**를 유지해야 합니다. 예를 들어, 인스턴스 **A**가 인스턴스 **B**의 세션 상태를 처리하지 않습니다. 스테이트풀 애플리케이션의 컨테이너화된 배포는 이러한 애플리케이션 아키텍처 요구 사항을 준수해야 합니다.

스토리지 측면에서 스테이트풀 애플리케이션에는 영구 스토리지가 가장 적절합니다. 이 모듈의 나머지 부분에서는 영구 스토리지에 대한 스토리지 옵션을 다룹니다.

StatefulSet

- 배포와 비슷하지만 생성된 Pod는 상호 교환할 수 없습니다.
- 각 Pod에 영구 식별자가 있습니다.
- StatefulSet는 다음이 필요한 애플리케이션에 유용합니다.
 - 안정적이고 고유한 네트워크 식별자
 - 안정적인 영구 스토리지
 - 순서가 지정되고 점진적인 배포 및 크기 조정
 - 순서가 지정되고 자동화된 롤링 업데이트



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

| 수강생용 노트

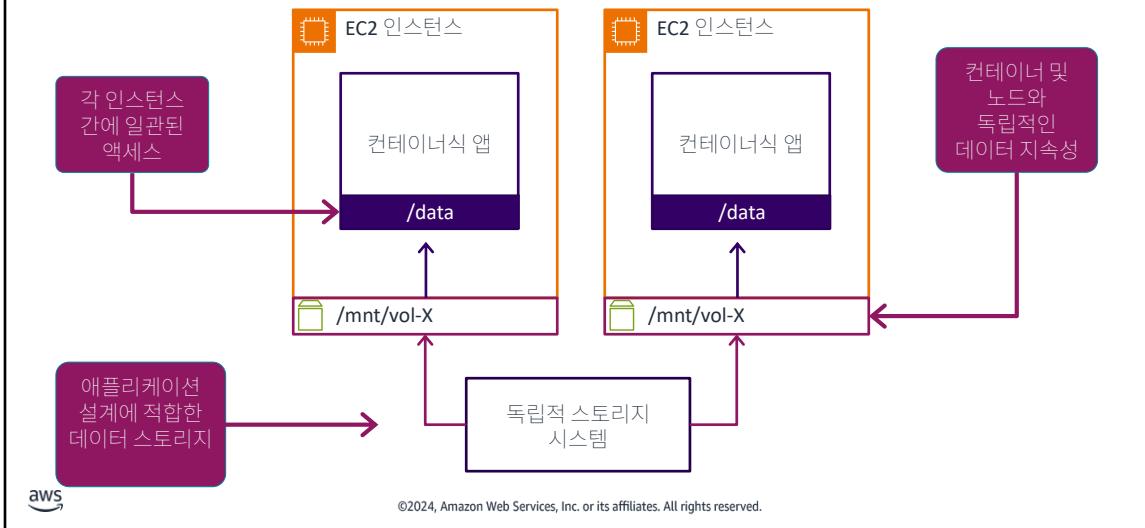
Kubernetes가 제공하는 대부분의 기능은 스테이트리스 애플리케이션으로 작업한다고 가정합니다. 하지만 영구 스토리지가 필요한 경우에는 어떻게 해야 할까요?
워크로드에 지속성을 제공하기 위해 스토리지 볼륨을 사용하려는 경우(다음 슬라이드에서 자세히 설명) StatefulSet를 솔루션의 일부로 사용할 수 있습니다.

StatefulSet의 개별 Pod는 오류에 취약하지만 영구 Pod 식별자를 사용하여 기존 볼륨을 오류가 발생한 항목을 대체하는 새 Pod와 더 쉽게 일치시킬 수 있습니다.

자세한 내용은

<https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>를
참조하십시오.

스테이트풀 애플리케이션 스토리지 요구 사항



~Dev notes

~statefull-app-storage: Architecture diagram of containerized stateful application using independent storage.

|수강생용 노트

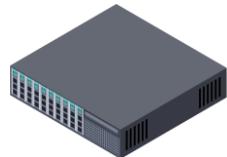
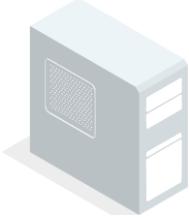
원래 컨테이너는 스테이트리스 애플리케이션을 실행하도록 구축되었습니다. 컨테이너 설계가 이동성 및 유연성에 적합했기 때문입니다. 그러나 컨테이너 사용은 스테이트풀 애플리케이션을 포함하도록 확장되었습니다. 스테이트풀 애플리케이션을 실행하는 데 따른 어려움은 임시 시스템에서 애플리케이션 상태를 보존하는 문제입니다. 따라서 어떻게 하면 이동성 및 유연성이라는 임시 시스템의 이점을 희생하지 않고 스테이트풀 컨테이너에 데이터 지속성을 제공할 수 있는지에 대한 질문을 해결하는 것이 핵심 과제입니다.

스테이트풀 애플리케이션 스토리지 요구 사항은 다음과 같이 요약될 수 있습니다.

- 각 애플리케이션 인스턴스 간에 일관된 액세스
컨테이너식 애플리케이션 설계는 기본 스토리지 구현과 분리되어야 합니다.
애플리케이션 설계와 관련하여 애플리케이션 개발자는 애플리케이션에서 사용할 데이터가 상주하는 위치를 자유롭게 참조하고 스토리지 구현에 의존하여 영구 스토리지를 제공할 수 있어야 합니다. 이러한 방식으로 컨테이너식 애플리케이션은 컨테이너화된 환경에서 실행되는 이점을 유지합니다.

- 컨테이너 및 노드와 독립적인 데이터 지속성
데이터 지속성은 컨테이너 또는 노드 수준에서 구현할 수 없습니다. 컨테이너는 일시적입니다. 이는 컨테이너 파일 시스템에 저장된 데이터가 일시적이라는 의미이기도 합니다. 컨테이너를 실행하는 노드에 로컬로 데이터를 저장하면 컨테이너 이상의 지속성을 제공하지만 이는 프로덕션 수준 컨테이너 오케스트레이션 환경을 위한 최적의 스토리지 솔루션은 아닙니다. 컨테이너 오케스트레이션 시스템은 적절히 또는 지정된 대로 컨테이너를 자유롭게 배치할 수 있어야 합니다. 더 나은 방법은 애플리케이션 스토리지를 컨테이너 오케스트레이션 시스템에서 완전히 분리하는 것입니다.
- 애플리케이션 설계에 적합한 데이터 스토리지
스테이트풀 애플리케이션은 데이터 스토어의 클러스터링된 인스턴스로 실행하거나 대규모 데이터 세트에 대한 배치 작업을 실행하는 등 다양한 비즈니스 요구 사항을 충족합니다. 따라서 스테이트풀 애플리케이션 요구 사항을 충족하는 독립적인 스토리지 옵션이 제공되어야 합니다.

영구 스토리지 옵션



호스트 기반(개발/테스트)

전용 스토리지

클라우드 스토리지



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

| 수강생용 노트

영구 스토리지 옵션에는 세 가지 범주가 있습니다.

- **호스트 기반 옵션(개발 및 테스트 환경에 가장 적합)** - 애플리케이션을 실행하는 서버의 로컬 스토리지입니다.
- **전용 스토리지** - 네트워크 연결 스토리지, 직접 연결 스토리지 및 스토리지 영역 네트워크 디바이스는 모두 전용 스토리지 옵션의 예입니다.
- **클라우드 기반 스토리지** - Amazon Elastic Block Store(Amazon EBS), Amazon Elastic File System(Amazon EFS), Amazon FSx for Lustre 및 Amazon FSx for ONTAP는 Amazon EKS에 사용할 수 있는 클라우드 스토리지 옵션입니다.

AWS 스토리지 옵션



Amazon Elastic Block Store(Amazon EBS)

영구 블록 스토리지를 사용하여 Pod 간에 볼륨을 공유합니다.



Amazon Elastic File System(Amazon EFS)

Amazon EKS에 배포되고 파일 시스템에 모두 동시에 액세스할 수 있는 스테이트풀 마이크로서비스 세트에서 파일 시스템을 공유합니다.



Amazon FSx

Amazon EKS 클러스터 전반에서 파일 시스템을 공유합니다. Amazon EKS는 Amazon FSx에서 지원하는 4가지 파일 시스템을 모두 지원합니다.



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

|수강생용 노트

AWS는 Pod가 사용할 수 있는 다음의 여러 스토리지 옵션을 제공합니다.

- **Amazon EBS:** 처리량 집약적 워크로드와 트랜잭션 집약적 워크로드 모두를 위한 고성능 영구 블록 스토리지입니다.
Amazon EBS에 대한 자세한 내용은 AWS 제품 페이지 (<https://aws.amazon.com/ebs>)를 참조하십시오.
- **Amazon EFS:** 스토리지를 프로비저닝하거나 관리하지 않고도 파일 데이터를 공유할 수 있는 탄력적 서버리스 파일 시스템입니다.
Amazon EFS에 대한 자세한 내용은 AWS 제품 페이지 (<https://aws.amazon.com/efs/>)를 참조하십시오.
- **Amazon FSx:** Amazon EKS 클러스터 전반에서 파일 시스템을 공유합니다. Amazon EKS는 Amazon FSx에서 지원하는 4가지 파일 시스템을 모두 지원합니다. 자세한 내용은 <https://aws.amazon.com/fsx/>를 참조하십시오.

이 모듈에서는 클라우드 스토리지 옵션인 Amazon EBS 및 Amazon EFS를 다룹니다.

Amazon FSx for Lustre에 대해 자세히 알아보려면 FSx for Lustre 사용 설명서 (<https://docs.aws.amazon.com/fsx/latest/LustreGuide/what-is.html>)를 참조하십시오.

Amazon FSx for NetApp ONTAP에 대해 자세히 알아보려면 FSx for ONTAP 사용 설명서 (<https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/what-is-fsx-ontap.html>)를 참조하십시오.



Running Containers on Amazon EKS

Kubernetes의 영구 스토리지

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



| 수강생용 노트

이 섹션에서는 **Kubernetes** 스토리지 모델의 구성 요소에 대해 설명합니다.

임시 볼륨

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: my-app
    image: corp.repo/my-app:1.0
    volumeMounts:
    - name: myapp-volume
      mountPath: /cache
  volumes:
  - name: myapp-volume
    emptyDir: {}
```



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

~Dev notes

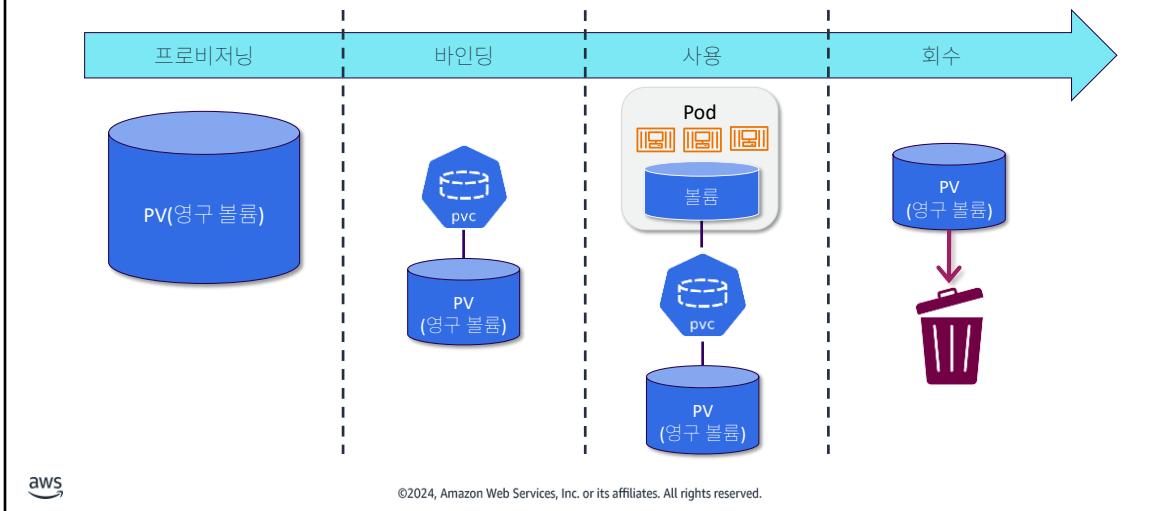
~Pod with Volume 3: Pod with an ephemeral volume.

| 수강생용 노트

애플리케이션은 공유 볼륨에 액세스하여 **Pod**에서 데이터를 공유하고 컨테이너 재시작 시에도 데이터를 지속할 수 있습니다. 이 **PodSpec** 예제 표시된 것처럼 **Pod**가 삭제되거나 다시 생성되면 기존 데이터가 모두 손실됩니다. 따라서 이 볼륨은 임시 볼륨으로 간주되며 장기적인 지속성 또는 내구성이 필요하지 않은 애플리케이션 데이터에 적합합니다.

Kubernetes 볼륨에 대해 자세히 알아보려면 **Kubernetes** 설명서의 **Volumes** 섹션 (<https://kubernetes.io/docs/concepts/storage/volumes/>)을 참조하십시오.

영구 볼륨(PV) 수명 주기



~Dev notes

~persistent-volume-lifecycle: Diagram of the persistent volume lifecycle. Details are in the notes.

~

| 수강생용 노트

영구 볼륨(PV)은 Kubernetes 클러스터의 네임스페이스가 지정되지 않은 리소스입니다.

영구 볼륨 클레임(PVC)은 Kubernetes 클러스터의 네임스페이스가 지정된

리소스입니다. PVC는 PV에 대한 요청이며, 리소스에 대한 클레임 예약 역할도 합니다.

PV와 PVC 간의 상호 작용은 다음 수명 주기를 따릅니다.

• 프로비저닝(영구 볼륨 생성)

영구 볼륨을 생성하는 방법에는 정적 및 동적 두 가지가 있습니다. 정적 방법은 영구 볼륨을 수동으로 생성하는 것입니다. 일반적으로 클러스터 관리자는 클러스터 사용자를 대신하여 PV를 생성합니다. PV는 네임스페이스가 지정되지 않은 객체이므로 Kubernetes에서 PV를 생성하려면 높은 클러스터 권한이 필요합니다.

따라서 정적 방법에는 높은 수준의 관리 오버헤드가 요구됩니다. 동적 방법에는 PVC 요청 또는 초기 PVC 소비를 통해 자동으로 PV를 생성하는 방법이 포함됩니다. 동적 방법은 상대적으로 소비 속도가 빠르고 관리 오버헤드가 적기 때문에 선호됩니다.

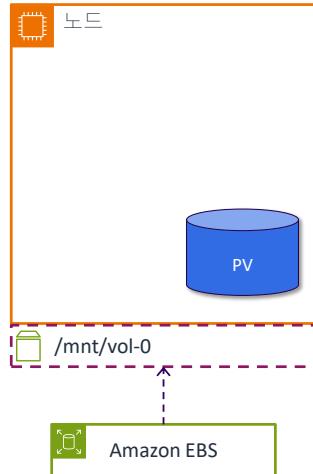
- **바인딩**
영구 볼륨 바인더 컨트롤러는 명시된 **PVC** 기준과 일치하는 **PV**를 찾은 다음 이들을 바인딩합니다. **PVC**는 항상 요청된 스토리지와 일치하거나 초과하는 **PV**를 가져옵니다. 일치하는 **PV**가 없는 경우 **PVC**는 바인딩되지 않은 상태로 유지됩니다. **PVC**는 항상 동적으로 프로비저닝된 **PV**와 일치하므로 동적 방법에는 이러한 제한이 적용되지 않습니다.
- **사용**
Pod는 **PVC**를 볼륨으로 사용합니다. **PV**는 **PVC**가 존재하는 한 항상 **PVC**에 바인딩됩니다. 즉, **PV**를 사용하는 **Pod**가 다시 생성되더라도 **PV**가 유지되고 사용 가능한 상태를 유지합니다.
- **회수**
PV는 더 이상 필요하지 않은 경우 회수 정책에 따라 회수할 수 있습니다. 현재 회수 정책은 보존 및 삭제 두 가지가 지원됩니다. 회수를 통해 리소스를 수동으로 회수할 수 있습니다. 관리자가 수동으로 **PV**를 제거하고 스토리지 객체에서 기존 데이터를 제거한 다음 스토리지 객체를 삭제해야 합니다. 삭제는 효과 측면에서 보존과 동일하지만 수동 개입이 필요하지 않습니다. 세 번째 정책인 재활용도 있지만 더 이상 사용되지 않습니다.

영구 볼륨(PV)

```
apiVersion: v1
kind: PersistentVolume
...
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  csi:
    driver: ebs.csi.aws.com
    volumeHandle: vol-09568627e337e1f35
  ...
...
```



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



~Dev notes

~PV-diagram: Diagram showing an Amazon EBS volume attached to a Kubernetes node.

|수강생용 노트

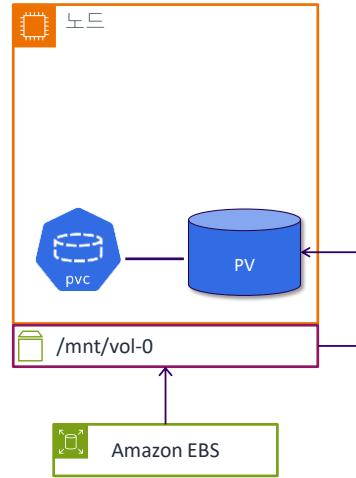
이 YAML 매니페스트는 PV 객체 기준을 지정합니다. 객체의 기준에는 5기비바이트 용량과 ReadWriteOnce 액세스 모드가 포함됩니다. 이는 볼륨을 단일 클러스터 노드에 의한 읽기-쓰기로 탑재할 수 있음을 의미합니다. 또한 YAML에는 볼륨을 백업할 의도된 스토리지 유형을 지정하는 **CSI(Container Storage Interface)** 필드도 포함되어 있습니다. 이 다이어그램은 PV가 생성되어 클레임에 바인딩된 최종 결과를 보여줍니다. 설계상 클레임되지 않은 PV는 볼륨 생성 및 클러스터 노드에 탑재 프로세스를 시작하지 않습니다. Amazon EBS 프로비저닝에 대한 자세한 내용은 이후 섹션에서 다룰 것입니다.

영구 볼륨 클레임(PVC)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```



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



~Dev notes

~pvc-diagram: Diagram showing an Amazon EBS volume associated with a PVC.

| 수강생용 노트

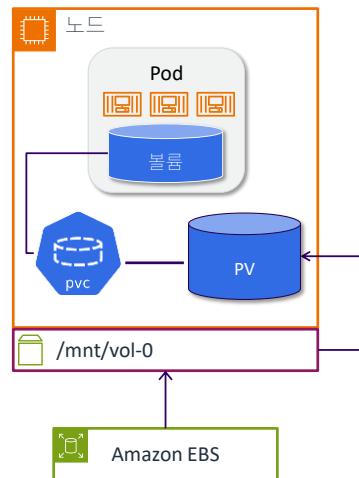
이 YAML 매니페스트는 영구 볼륨 클레임(PVC)의 파라미터를 선언합니다. 객체의 파라미터는 이전에 표시된 PV의 스토리지 기준과 일치합니다. 따라서 PVC는 PV에 바인딩됩니다. 이 다이어그램은 Amazon EBS 볼륨이 지원하는 PV에 바인딩된 PVC를 보여줍니다. 이 예에서 PVC 생성은 PV에 대한 바인딩 프로세스를 시작합니다.

PVC를 볼륨으로 사용

```
apiVersion: v1
kind: Pod
...
spec:
  containers:
    - name: stateful-app
      image: corp.repo/stateful-app:1.0
      volumeMounts:
        - mountPath: "/data"
          name: data-volume
      volumes:
        - name: data-volume
          persistentVolumeClaim:
            claimName: my-pvc
```



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

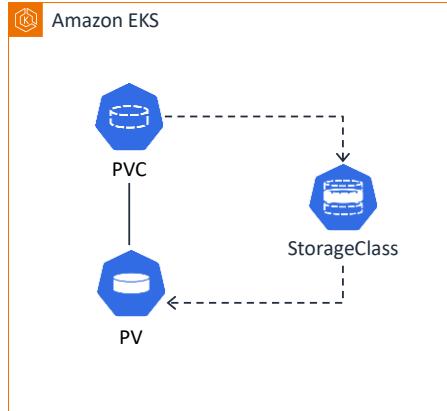


| 수강생용 노트

이 Pod YAML 매니페스트 예는 PodSpec에서 PVC를 사용하는 방법을 보여줍니다. 객체의 volumes 필드는 앞서 표시된 PVC 이름과 일치합니다. 따라서 Pod는 PVC를 통해 프로비저닝된 볼륨을 탑재합니다. 이 다이어그램은 PVC를 사용하는 Pod를 보여줍니다. 이 PVC는 간접적으로 Amazon EBS 볼륨을 사용합니다.

스토리지 클래스

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: simple-ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: Immediate
```



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

~Alt text

~ 스토리지 클래스를 사용하는 PV 생성 워크플로를 보여주는 다이어그램입니다.
자세한 내용은 노트에 있습니다.

~

|수강생용 노트

StorageClass 객체는 관리자가 제공하는 스토리지의 ‘클래스’를 설명하는 방법입니다. 다양한 클래스가 QoS 수준, 백업 정책 또는 클러스터 관리자가 결정한 임의의 정책에 매핑될 수 있습니다. 이 YAML 매니페스트 예는 프로비저너 및 **volumeBindingMode** 필드를 제공합니다. 프로비저너는 **Amazon EBS CSI** 드라이버가 클러스터 사용자를 대신하여 스토리지를 프로비저닝하도록 지정합니다. 볼륨 바인딩 모드를 **Immediate**로 설정하면 PV가 생성되고 PVC 생성 시 바인딩됩니다. 대체 값은 **WaitForFirstConsumer**입니다. 이는 Pod가 생성된 후에만 PV가 PVC에 바인딩된다는 의미입니다. **WaitForFirstConsumer**를 설정하면 볼륨이 Pod의 대상 클러스터 노드에 바인딩됩니다.

애플리케이션 개발자는 스토리지 클래스를 사용하여 애플리케이션에 필요한 스토리지 유형을 지정할 수 있습니다. 스토리지 클래스를 사용하려면 PVC에서 대상 스토리지 클래스의 이름을 지정합니다. 다음 섹션에서는 PVC에서 스토리지 클래스를 지정하는 예를 제공합니다.



Running Containers on Amazon EKS

AWS 스토리지 서비스를 사용하는 영구 스토리지



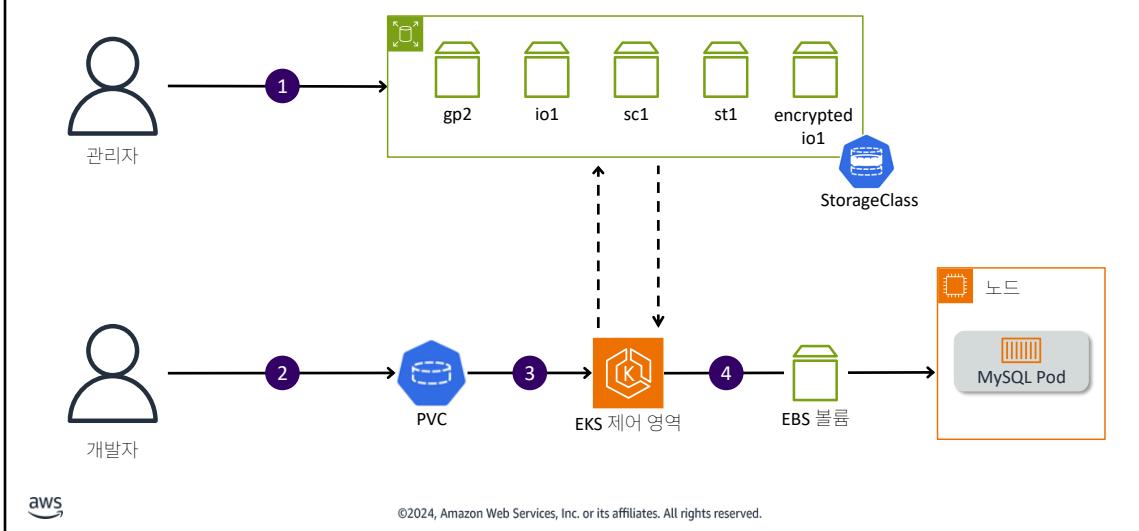
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| 수강생용 노트

이 섹션에서는 다음 작업을 수행하는 방법을 설명합니다.

- PV 및 PVC의 수명 주기 설명
- CSI 드라이버 구성
- 가장 적절한 AWS 스토리지 서비스 선택
- EKS 클러스터에 스토리지 프로비저닝
- Pod에 스토리지 할당

워크플로: 스토리지 볼륨 수명 주기



~Dev notes

~storage-cycle-workflow: This slide contains a diagram outlining a four step process on how a developer might provision a storage volume.

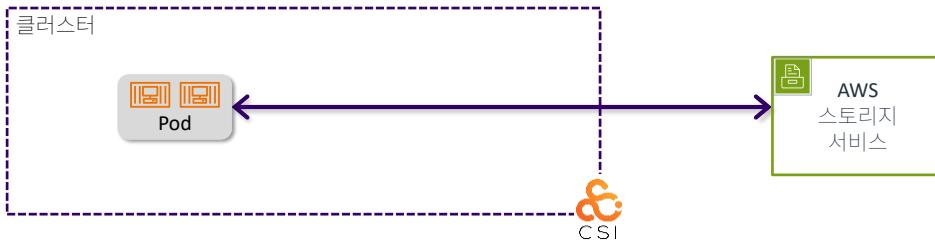
|수강생용 노트

위 다이어그램은 EKS 서비스를 사용하여 스토리지 볼륨을 프로비저닝하는 워크플로를 보여줍니다.

1. 관리자는 CSI 드라이버를 지정하는 **StorageClass**를 사전 프로비저닝할 수 있습니다.
2. 최종 사용자는 특정 볼륨 유형을 요청할 수 있습니다. (예를 들어 암호화된 io1 EBS 볼륨을 원할 수도 있습니다.)
3. 제어 루프는 PVC 요청을 감시하고 PV가 존재하는 경우 볼륨을 할당합니다.
4. 그런 다음 스테이트풀 워크로드가 생성됩니다(이 예에서는 MySQL Pod).

자세한 내용은 <https://docs.aws.amazon.com/eks/latest/userguide/storage.html>를 참조하십시오.

Container Storage Interface(CSI) 드라이버



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

~ Alt text

~ Pod가 한 AWS 스토리지 서비스에 연결되어 있습니다.

~

| 수강생용 노트

Container Storage Interface는 Kubernetes와 같은 컨테이너 오케스트레이션 시스템(COS)의 컨테이너화된 워크로드에 임의의 블록 및 파일 스토리지 시스템을 노출하는 표준입니다.

서드 파티 스토리지 공급자는 CSI를 사용하여 핵심 Kubernetes 코드를 건드리지 않고도 Kubernetes에서 새 스토리지 시스템을 노출하는 플러그 인을 쓰고 배포할 수 있습니다.

AWS는 Kubernetes에 노출할 수 있는 다양한 유형의 스토리지를 위한 CSI 드라이버를 제공합니다. 사용 가능한 CSI 드라이버 옵션에 대해 자세히 알아보려면 **Amazon EKS** 사용 설명서의 **스토리지** 섹션

(<https://docs.aws.amazon.com/eks/latest/userguide/storage.html>)을 참조하십시오.

CSI 드라이버 요구 사항



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

~Alt text

~ CSI 드라이버가 Amazon EKS 클러스터 내에서 작동하는 데 필요한 구성 요소를 보여주는 다이어그램입니다.

~

| 수강생용 노트

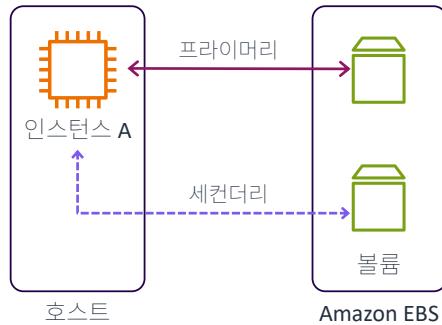
각 AWS 스토리지 옵션에는 클러스터용으로 설치 및 구성하는 CSI 드라이버가 있습니다. CSI 드라이버에는 대상 AWS 스토리지 서비스에 대한 적절한 IAM 권한이 필요합니다. 특히, 명시된 권한이 있는 IAM 정책이 IAM 역할에 할당됩니다. IAM 역할을 수임하도록 CSI 드라이버 Service Account가 구성됩니다. CSI 드라이버 Service Account는 CSI 드라이버 애플리케이션을 실행하는 Pod에 할당됩니다. Service Account에 AWS 권한을 할당하는 방법에 대한 자세한 내용은 다른 모듈에서 설명합니다.

Amazon EKS용 CSI 드라이버 설치 및 구성에 대해 자세히 알아보려면 **Amazon EKS 사용 설명서의 스토리지 섹션**

(<https://docs.aws.amazon.com/eks/latest/userguide/storage.html>)을 참조하십시오.

Amazon Elastic Block Store(Amazon EBS)

- Amazon EC2 인스턴스를 위한 블록 수준 스토리지입니다.
- 하나 이상의 EBS 볼륨을 단일 EC2 인스턴스에 연결합니다.
- EBS 볼륨은 EC2 인스턴스와 독립적인 수명 주기를 갖습니다.



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

| 수강생용 노트

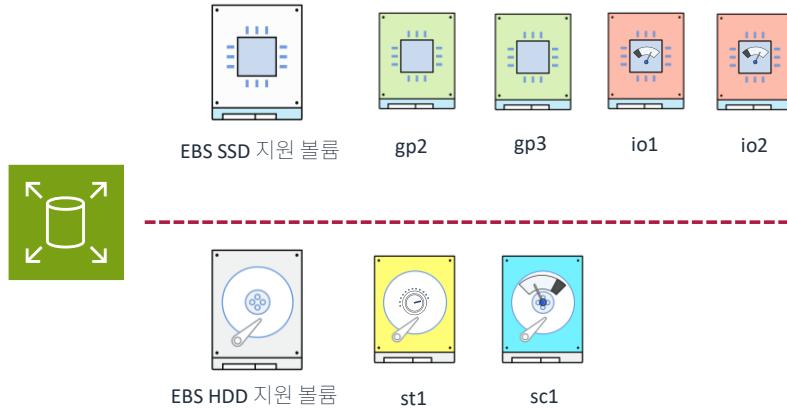
Amazon EBS 볼륨은 Amazon EC2 인스턴스를 위해 안정적이고 분리 가능한 블록 수준 스토리지를 제공합니다. 이 볼륨은 인스턴스에 탑재되므로 데이터가 저장된 위치와 인스턴스에서 사용되는 위치 간에 매우 짧은 지연 시간을 제공할 수 있습니다. 이러한 이유로 Amazon EC2 인스턴스에서 높은 IOPS 요구 사항이 있는 애플리케이션을 실행하는 데 사용할 수 있습니다.

Amazon EBS는 Amazon EC2 인스턴스와 독립적으로 유지되는 스토리지 볼륨으로 노출됩니다. 원하는 인프라 상태에 따라 필요한 경우 EC2 인스턴스에서 Amazon EBS 볼륨을 연결 및 분리할 수 있습니다.

Amazon EBS에 대한 자세한 내용은 **Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 ‘Amazon Elastic Block Store(Amazon EBS)’** (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html>)를 참조하십시오.

Amazon EBS 볼륨 유형

- 솔리드 스테이트 드라이브(SSD)는 고성능 및 범용 워크로드를 위한 것입니다.
- 하드 디스크 드라이브(HDD)는 대용량 또는 자주 액세스하지 않는 데이터용입니다.
- io2에는 Block Express용 옵션이 포함되어 있습니다.



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

| 수강생용 노트

범용 솔리드 스테이트 드라이브(SSD) 볼륨(gp2, gp3)에서는 광범위한 사용 사례에 이상적인 비용 효율적인 스토리지를 제공합니다. 부팅 볼륨, 중소형 데이터베이스, 그리고 개발 및 테스트 환경에는 이상적입니다.

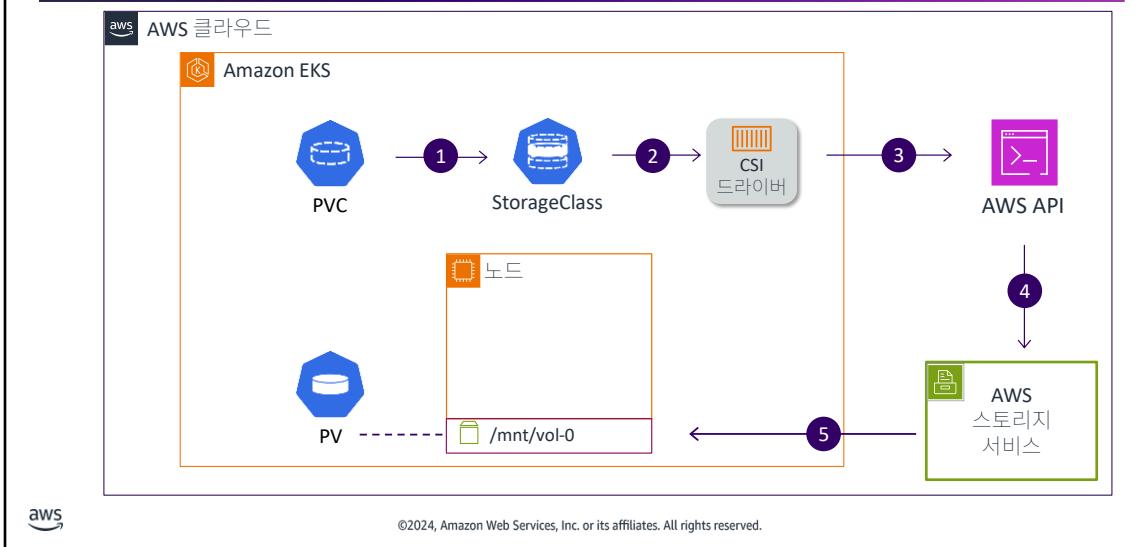
프로비저닝된 IOPS SSD 볼륨(io1, io2)은 스토리지 성능 및 일관성에 민감한 데이터베이스 워크로드와 같은 I/O 집약적 워크로드의 요구사항을 충족하도록 설계되었습니다. 프로비저닝된 IOPS SSD 볼륨은 일관된 IOPS 속도를 사용합니다. 볼륨을 생성할 때 속도를 지정합니다. Amazon EBS에서는 전체 사용 시간 중 99.9% 동안 프로비저닝된 성능을 제공합니다. 참고로 2023년 11월 21일 이후 생성된 io2 볼륨은 모두 io2 Block Express 볼륨입니다.

처리량 최적화 하드 디스크 드라이브(HDD) 볼륨(st1)에서는 IOPS가 아닌 처리량을 기준으로 성능을 정의하는 저비용 마그네틱 스토리지를 제공합니다. 이 볼륨 유형은 Amazon EMR, 추출, 변환, 로드(ETL), 데이터 웨어하우스, 로그 처리 등의 대규모 순차 워크로드에 적합합니다.

콜드 HDD(sc1) 볼륨에서는 IOPS가 아닌 처리량을 기준으로 성능이 정의되는 저비용 마그네틱 스토리지를 제공합니다. sc1은 대규모 순차 콜드 데이터 워크로드에 적합합니다. 데이터에 자주 액세스하지 않는 경우 저렴한 블록 스토리지를 제공합니다.

Amazon EBS 볼륨 유형에 대한 자세한 내용은 ‘Amazon EBS 볼륨 유형’
(<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html>)을 참조하십시오.

Amazon EKS 영구 스토리지 워크플로



~Alt text

~ AWS 스토리지를 Amazon EKS 노드에 제공하는 워크플로를 보여주는
다이어그램입니다. 자세한 내용은 노트에 있습니다.

~

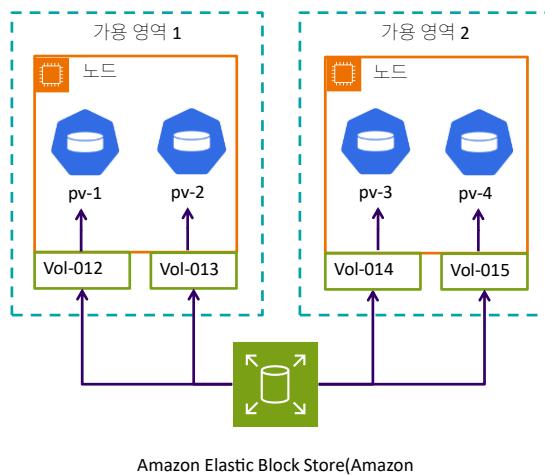
| 수강생용 노트

이 다이어그램은 동적 방법을 사용하여 AWS 스토리지 옵션을 Amazon EKS 클러스터에
제공하는 개략적 워크플로를 나타냅니다.

1. PVC는 스토리지 요청에 대한 세부 정보를 제공하고 요청을 이행하기 위한 스토리지
클래스를 지정합니다.
2. StorageClass 객체는 CSI 드라이버에게 적절한 AWS 스토리지 볼륨을
프로비저닝하도록 지시합니다.
3. CSI 드라이버는 대상 AWS 스토리지 서비스에 대한 AWS API 호출을 수행합니다.
4. 이러한 호출은 대상 AWS 스토리지로 전달됩니다.
5. AWS 스토리지 볼륨은 Amazon EKS 클러스터의 대상 노드에 탑재되고 영구 볼륨
객체와 연결됩니다.

영구 볼륨: Amazon EBS

- 애플리케이션을 위한 일대일 스토리지 옵션
- StatefulSet 객체에 적합
- 동적 규모 조정 옵션



Amazon Elastic Block Store(Amazon EBS)



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

~Alt text

~ Amazon EBS가 지원하는 두 노드에 분산된 영구 볼륨을 보여주는 아키텍처 다이어그램입니다.

~

|수강생용 노트

Amazon EBS가 지원하는 영구 스토리지는 스테이트풀 애플리케이션을 위한 일대일 스토리지 옵션을 제공합니다. 이는 애플리케이션의 각 인스턴스가 저장된 상태에 대한 독립적인 스토리지를 갖는 것을 의미합니다. 각 볼륨은 단일 가용 영역의 한 노드에 바인딩됩니다. Amazon EBS에서는 일대다 스토리지 옵션이 가능하지만 노드 종속성과 복잡한 Pod 예약 때문에 권장하지 않습니다. 권장 방법은 StatefulSet 객체를 사용하는 것입니다. 각 복제본이 모든 가용 영역의 모든 노드에서 실행되고 자체 영구 스토리지에 액세스할 수 있기 때문입니다. StatefulSet 객체에 대해 자세히 알아보려면 Kubernetes 설명서의 StatefulSets 섹션

(<https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>)을 참조하십시오.

Amazon EBS가 지원하는 영구 볼륨은 동적으로 크기를 조정할 수 있습니다.

StorageClass 객체가 이를 허용하도록 구성되어야 합니다. Amazon EBS 스토리지 클래스 예에서 이 설정을 보여줍니다.

예: Amazon EBS 스토리지 클래스

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-standard
  provisioner: ebs.csi.aws.com
  volumeBindingMode: WaitForFirstConsumer
  allowVolumeExpansion: true
  reclaimPolicy: Delete
parameters:
  type: gp3
  encrypted: "true"
```



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

| 수강생용 노트

여기에서 표시된 YAML 매니페스트는 Amazon EBS가 지원하는 **StorageClass** 객체를 선언합니다. 이 예에서는 **StorageClass**의 기능 및 구성을 보여줍니다. 예를 들어 스토리지 클래스를 사용하면 요청 시 동적으로 볼륨을 확장할 수 있습니다. 기존 볼륨 크기를 변경하려면 클러스터 사용자는 기존 **PVC**의 스토리지 요청을 새 크기로 변경합니다. **PVC**를 변경한 후 확장이 적용되도록 하려면 **PVC**를 사용하는 **Pod**를 다시 생성해야 합니다.

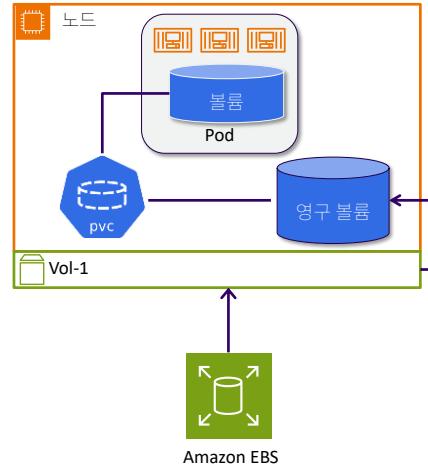
파라미터 섹션에서는 **PV**에 **gp3** 스토리지를 사용되고 볼륨에서 데이터를 암호화하도록 지정합니다. 사용 가능한 **StorageClass** 파라미터에 대해 자세히 알아보려면 [Amazon EBS CSI 드라이버 GitHub 리포지토리\(<https://github.com/kubernetes-sigs/aws-ebs-csi-driver/blob/master/docs/parameters.md>\)](https://github.com/kubernetes-sigs/aws-ebs-csi-driver/blob/master/docs/parameters.md)를 참조하십시오.

예: Amazon EBS 영구 볼륨 클레임

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: app-pvc
  namespace: prod
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  storageClassName: ebs-standard
  resources:
    requests:
      storage: 10Gi
```



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



~Alt text

~ Amazon EBS 볼륨이 지원하는 PVC를 사용하는 Pod입니다.

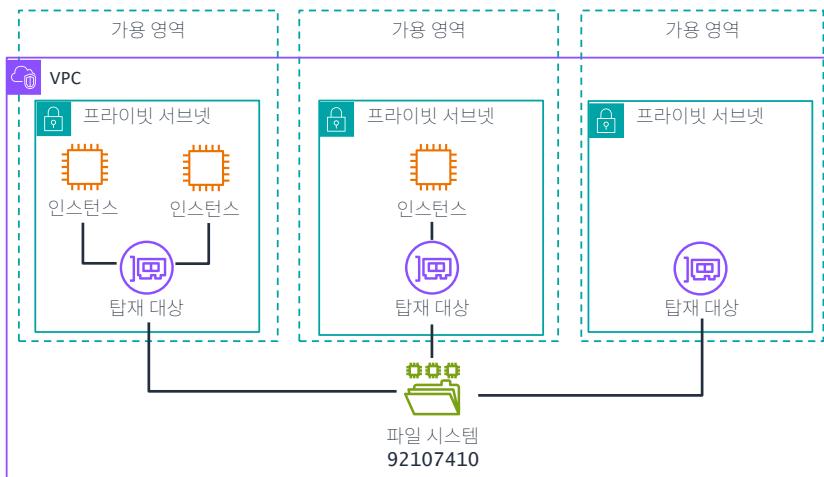
~

|수강생용 노트

다음은 **ebs-standard**라는 스토리지 클래스를 사용하는 Amazon EBS PVC의 YAML 매니페스트 예입니다. **accessMode**인 **ReadWriteOnce**는 하나의 노드만 볼륨을 읽기-쓰기로 탑재할 수 있음을 의미합니다. **volumeMode**는 EBS 볼륨을 파일 시스템으로 포맷하도록 지시합니다. 또는 **volumeMode**를 **Block**으로 설정할 수 있습니다. 이는 Pod가 원시 블록 디바이스로 볼륨에 액세스한다는 의미입니다. 원시 블록 디바이스를 사용하려면 별도의 프로세스를 통해 원하는 파일 시스템 유형으로 포맷한 다음 적절하게 탑재해야 합니다. 여러 Pod는 동일한 노드에서 실행되는 한 볼륨에 액세스할 수 있습니다. 이 PVC 예는 테스트 목적, 정적 Pod 또는 특수 배포에 적합합니다.

Amazon Elastic File System(Amazon EFS)

- 확장 가능하고 탄력적인 파일 시스템을 위해 Amazon EFS를 선택합니다.
- NFSv4 프로토콜을 사용하여 연결합니다.
- 여러 EC2 인스턴스에서 동시에 파일 시스템에 액세스할 수 있습니다.



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

~Alt text

~탑재 대상을 통해 EFS 파일 시스템 하나에 액세스하는 AZ 3개에 EC2 인스턴스가 포함되어 있는 VPC입니다. 해당 내용은 노트에 설명되어 있습니다.

~

|수강생용 노트

Amazon EFS는 Linux 기반의 워크로드를 AWS 클라우드 서비스와 온프레미스 리소스에서 사용할 수 있도록 확장 가능하며 탄력적인 파일 시스템을 제공합니다.

파일 시스템을 생성하여 Amazon EC2 인스턴스에 탑재한 다음, 해당 파일 시스템에서 데이터를 읽고 쓸 수 있습니다. Amazon EFS 파일 시스템을 Network File System(NFS) 버전 4.0 및 4.1(NFSv4) 프로토콜을 통해 VPC에 탑재할 수 있습니다. 스토리지 용량 필요가 증가함에 따라 파일 시스템을 확장하기 위해 조치를 취할 필요가 없습니다.

VPC의 여러 Amazon EC2 인스턴스에서 동시에 Amazon EFS 파일 시스템에 액세스할 수 있으므로 단일 연결을 넘어 확장되는 애플리케이션이 파일 시스템에 액세스할 수 있습니다.

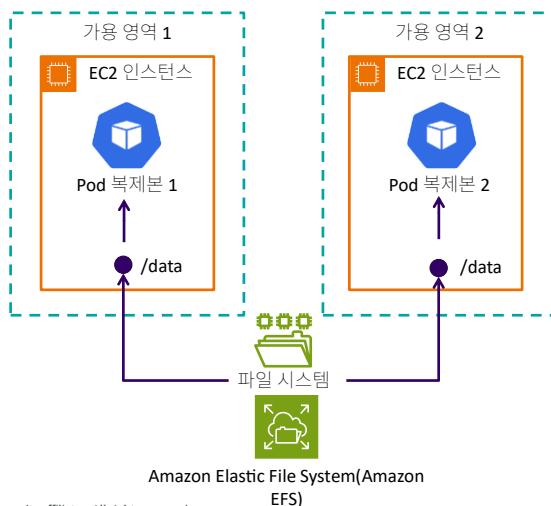
Amazon EFS 파일 시스템은 AWS 리전 내에서 데이터를 중복 저장하므로 우수한 가용성과 내구성이 보장됩니다. 파일 시스템의 가용성 및 내구성을 고려하여 다음과 같은 스토리지 클래스를 선택할 수 있습니다.

- **Standard** 스토리지 클래스를 선택하면 AWS 리전의 모든 가용 영역에 파일 시스템 데이터와 메타데이터를 중복 저장하는 파일 시스템이 생성됩니다. AWS 리전의 각 가용 영역에 탑재 대상을 생성할 수도 있습니다. **Standard** 스토리지 클래스에서는 최고 수준의 가용성과 내구성을 제공합니다.
- **One Zone** 스토리지 클래스를 선택하면 단일 가용 영역 내에 파일 시스템 데이터와 메타데이터를 저장하는 파일 시스템이 생성됩니다. **One Zone** 스토리지 클래스를 사용하는 파일 시스템에는 탑재 대상이 하나만 포함될 수 있습니다. 이 탑재 대상은 파일 시스템이 생성되는 가용 영역에 있습니다.

슬라이드의 예에서 VPC는 Amazon EFS Standard 스토리지 클래스를 사용합니다. VPC에는 프라이빗 서브넷이 3개 있으며 각 프라이빗 서브넷은 서로 다른 AZ에 있습니다. Standard 스토리지 클래스에서는 각 서브넷에 자체 탑재 대상이 포함될 수 있습니다. 각 서브넷의 EC2 인스턴스는 해당 AZ에 있는 탑재 대상을 통해 파일 시스템에 액세스할 수 있습니다.

영구 볼륨: Amazon EFS

- 애플리케이션을 위한 일대다 스토리지 옵션
- 데이터 공유에 적합
- EFS 인프라 사전 요구 사항
- AWS Fargate에서 실행되는 Pod에 사용 가능한 옵션



~Alt text

~ Amazon EFS가 지원하는 두 노드에 분산된 영구 볼륨을 보여주는 아키텍처 다이어그램입니다.

~

|수강생용 노트

Amazon EFS가 지원하는 영구 볼륨은 스테이트풀 애플리케이션을 위한 일대다 스토리지 옵션을 제공합니다. Amazon EFS 영구 스토리지는 동일한 리전의 여러 가용 영역에 있는 노드 간에 단일 네트워크 볼륨을 공유할 수 있는 유연성을 제공합니다.

Amazon EFS 스토리지의 유연성 덕분에 스테이트풀 애플리케이션이 동일한 데이터 스토리지에 액세스하는 여러 복제본을 실행할 수 있습니다. 애플리케이션 개발자는 적절한 액세스 모드를 설정할 경우 스테이트풀 애플리케이션에 배포 객체를 사용할 수 있습니다.

Amazon EFS 영구 스토리지를 사용할 때 주의할 점은 애플리케이션 배포 전에 EFS 파일 시스템과 네트워크 액세스를 생성해야 한다는 것입니다. Amazon EFS는 노드 간 파일 공유 액세스를 위해 NFS 버전 4.1을 사용합니다. 따라서 Amazon EFS에서 들어오는 NFS 네트워크 액세스에 대해 클러스터의 VPC 서브넷 보안 그룹을 구성해야 합니다.

Amazon EFS 파일 시스템을 클러스터 서브넷의 탑재 대상과 함께 생성해야 합니다.

Amazon EFS에 대해 자세히 알아보려면 **Amazon Elastic File System** 사용 설명서 (<https://docs.aws.amazon.com/efs/latest/ug/index.html>)를 참조하십시오.

마지막으로, **AWS Fargate Pod**는 **Amazon EFS**만 지원합니다. 스테이트풀 애플리케이션 워크로드가 **Fargate Pod**를 대상으로 하는 경우 **Amazon EFS**를 영구 볼륨으로 사용하십시오.

예: Amazon EFS 스토리지 클래스

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
  provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap
  fileSystemId: fs-92107410
  directoryPerms: "700"
  basePath: "/dynamic_provisioning" # optional
```



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

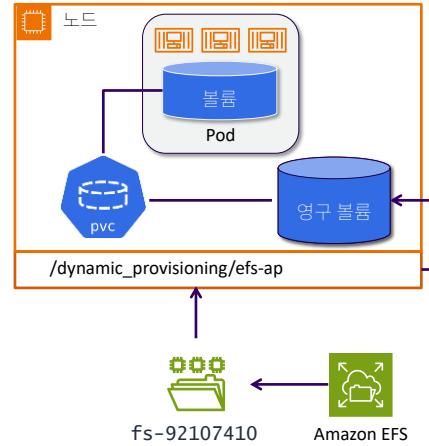
| 수강생용 노트

여기에서 표시된 YAML 매니페스트는 Amazon EFS가 지원하는 **StorageClass** 객체의 예입니다. 파라미터에는 파일 시스템이 대상 호스트 및 사전 생성된 Amazon EFS 파일 시스템에 탑재되는 방법 지정이 포함됩니다. Amazon EFS가 지원하는 스토리지 클래스를 지정하는 **PVC**가 생성되면 CSI 드라이버는 AWS API 호출을 수행하여 기존 EFS 파일 시스템에 액세스 포인트를 생성합니다. 새 액세스 포인트는 해당 영구 볼륨을 탑재하는 데 사용됩니다.

사용 가능한 Amazon EFS StorageClass 파라미터에 대해 자세히 알아보려면 Amazon EFS CSI 드라이버 GitHub 리포지토리(<https://github.com/kubernetes-sigs/aws-efs-csi-driver>)를 참조하십시오.

예: Amazon EFS 영구 볼륨 클레임

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: efs-claim
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: efs-sc
  resources:
    requests:
      storage: 5Gi
```



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

~Alt text

~ Amazon EFS가 지원하는 영구 스토리지를 사용하는 Pod의 아키텍처
다이어그램입니다.

~

|수강생용 노트

YAML 매니페스트는 Amazon EFS에서 영구 스토리지를 요청하는 PVC 객체를 선언하는 예입니다. accessMode는 **ReadWriteMany**를 지정합니다. 이는 볼륨이 여러 노드에 의한 읽기-쓰기로 탑재될 수 있음을 의미합니다. 클러스터 사용자는 여러 노드에 여러 복제본을 배포하려고 할 때 이 **accessMode**를 설정합니다.

Amazon EFS는 파일 시스템 용량을 강제 적용하지 않는 탄력적 파일 시스템입니다. PV 및 PVC의 실제 스토리지 용량 값은 파일 시스템 생성 시 사용되지 않습니다. 그러나 PVC 및 PV의 스토리지 용량은 Kubernetes의 필수 필드이므로 유효한 값을 지정해야 합니다. 이 예에서 5기비바이트는 임의의 값입니다. 대상 Pod는 5기비바이트 이상의 실제 Amazon EFS 파일 시스템 스토리지에 액세스할 수 있습니다.

영구 스토리지 사용 가이드라인

애플리케이션 개발자

- 애플리케이션 YAML 매니페스트에 PV 객체를 포함하지 마십시오.
- 애플리케이션 YAML 매니페스트에 PVC 객체를 포함하십시오.
- 적절한 스토리지 클래스를 지정하십시오.
- 기존 리소스 할당량을 준수하십시오.

클러스터 관리자

- 스토리지 클래스를 사용하여 스토리지 옵션을 제공하십시오.
- 기본 스토리지 클래스를 정의하십시오.
- 바인딩되지 않은 PVC를 모니터링하십시오.
- 리소스 할당량에서 필요한 스토리지 한도를 정의하십시오.



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

~Dev notes

~ <https://kubernetes.io/docs/concepts/storage/persistent-volumes/#writing-portable-configuration>

~

| 강사용 노트

| 지금이 영구 스토리지 프로비저닝을 시연하기 좋은 시점입니다.

| 1. Amazon EBS 스토리지 클래스를 배포합니다.

| 2. 영구 볼륨 클레임을 배포합니다.

| 3. 영구 볼륨 클레임을 사용하여 애플리케이션을 배포합니다.

| 4. 영구 볼륨 및 애플리케이션의 상태를 확인합니다.

| 5. 애플리케이션 및 영구 볼륨 클레임을 제거합니다.

| 6. 영구 볼륨의 상태를 확인합니다.

| 데모를 시작하기 전에 EBS CSI 드라이버가 설치되어 작동하는지 확인하십시오.

| **reclaimPolicy** 설정(보존)을 호출하여 PVC가 삭제된 후 볼륨에 어떤 영향이 있는지 수강생에게 보여줍니다.

| 수강생용 노트

애플리케이션 개발자:

수동 생성 대신 스토리지 클래스를 사용하여 애플리케이션을 위한 영구 볼륨을 프로비저닝합니다. 영구 볼륨은 네임스페이스가 지정되지 않은 객체이므로 영구 볼륨을 생성하려면 더 높은 수준의 클러스터 권한이 필요합니다. 사용자 또는 **CI/CD** 파이프라인에 대상 클러스터 환경 내에서 **PV**를 생성하는 데 필요한 권한이 없을 수 있습니다. **YAML** 매니페스트 번들 또는 **Helm** 차트에 **PVC** 객체를 포함하십시오. **PVC YAML** 매니페스트에서 대상 **Amazon EKS** 클러스터에 대해 스토리지 클래스가 지정되어 있는지 확인하십시오. 스토리지 요청이 대상 네임스페이스의 스토리지 양을 제한하는 스토리지 할당량을 준수하는지 확인하십시오.

클러스터 관리자:

스토리지 클래스를 사용하여 클러스터 사용자에게 스토리지 옵션을 제공하십시오. 또한 스토리지 클래스에 대한 스토리지 체계를 생성하십시오. 포괄적인 스토리지 클래스 체계를 통해 개발자는 애플리케이션에 적합한 스토리지 클래스를 신속하게 참조할 수 있습니다. 네트워크 스토리지와 연결된 기본 스토리지 클래스를 지정하십시오. 기본 스토리지 클래스는 별도의 스토리지 클래스를 지정하지 않은 경우 **PVC** 객체가 일부 스토리지에 바인딩되도록 보장합니다. 마지막으로 바인딩되지 않은 **PVC**를 모니터링하십시오. 애플리케이션 개발자는 바인딩되지 않은 **PVC**에 대한 알림을 받으면 즉시 조치를 취할 수 있습니다. 또한 바인딩되지 않은 **PVC**가 클러스터 자체의 기본 구성 문제를 나타낼 수 있습니다. 스토리지 사용량을 제한해야 하는 경우 리소스 할당량을 사용하십시오.

활동



aws

스토리지 옵션 선택

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

| 수강생용 노트

이 모듈에서 다루는 스토리지 옵션을 사용하여 명시된 요구 사항에 맞는 스토리지 유형을 선택합니다.

스토리지 선택(1/8)

시나리오: 여러 복제본과 함께 배포된 스테이트리스 애플리케이션

스토리지:

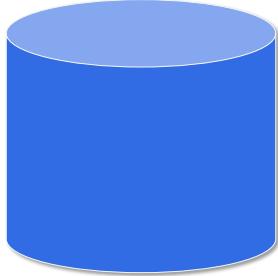


©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

스토리지 선택(2/8)

시나리오: 여러 복제본과 함께 배포된 스테이트리스 애플리케이션

스토리지: 볼륨(임시)



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

스토리지 선택(3/8)

시나리오: 스테이트풀 애플리케이션 복제본이 공유 데이터에
액세스해야 함

스토리지:



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

스토리지 선택(4/8)

시나리오: 스테이트풀 애플리케이션 복제본이 공유 데이터에
액세스해야 함

스토리지: Amazon EFS



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

스토리지 선택(5/8)

시나리오: 각 스테이트풀 애플리케이션 복제본에 필요에 따라 확장 가능한 독립적인 데이터 스토리지가 필요

스토리지:

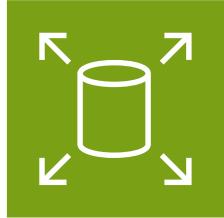


©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

스토리지 선택(6/8)

시나리오: 각 스테이트풀 애플리케이션 복제본에 필요에 따라 확장 가능한 독립적인 데이터 스토리지가 필요

스토리지: Amazon EBS



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

스토리지 선택(7/8)

시나리오: AWS Fargate Pod를 사용하여 스테이트풀 애플리케이션을 배포

스토리지:



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

스토리지 선택(8/8)

시나리오: AWS Fargate Pod를 사용하여 웹 애플리케이션을 배포

스토리지: Amazon EFS



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



Running Containers on Amazon EKS

Secret 관리



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

| 수강생용 노트

이 섹션에서는 **ASCP**를 사용하여 **Kubernetes Secret**을 관리하는 방법을 알아봅니다.

Secret 저장



aws

Secret으로 저장할 수 있는 것은 무엇입니까?

- 사용자 및 애플리케이션 자격 증명
- 암호
- 토큰
- SSH 키

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

Secret은 암호, 토큰, 또는 키와 같은 몇 가지 민감한 데이터를 포함하는 객체입니다. 모든 민감한 정보를 Secret으로 저장하십시오. **Kubernetes Secret** 객체는 이 민감한 정보를 PodSpec 또는 컨테이너 이미지의 외부에 저장하고 관리할 수 있게 해줍니다.

Secret 관리



Secret 관리

- 외부 Secret은 각 컨테이너에서 개별적으로 가져옵니다.
- 컨테이너는 중앙 집중식 서비스를 인증하거나 해당 서비스와 페더레이션됩니다.
- 서비스는 요청된 Secret을 암호화하여 컨테이너에 전달합니다.
- 컨테이너는 Secret을 해독하고 데이터베이스 또는 API를 인증합니다.



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

자격 증명을 Dockerfile이나 컨테이너 이미지로 하드 코딩해서는 안 되지만, 컨테이너는 여전히 데이터베이스와 스토리지에 액세스해야 하고 애플리케이션에서 다른 API에 인증해야 합니다. 이러한 과정은 Secret 관리를 통해 이루어집니다. Secret 관리를 이용할 경우 AWS Secrets Manager 또는 Vault by Hashicorp와 같은 서비스를 통해 모든 Secret을 외부에 저장합니다.

각 컨테이너는 필요할 때 Secret을 하나씩 가져옵니다. 컨테이너는 특정 컨테이너가 어떤 Secret을 검색할 수 있는지 알고 있는 중앙 집중식 서비스에 인증합니다. Secret 관리 서비스는 요청된 Secret을 암호화하여 컨테이너에 전달합니다. 컨테이너는 해당 Secret을 해독하고 데이터베이스, API 또는 필요한 곳에 인증합니다.

이러한 방식으로 Secret을 중앙에서 관리하고 순환할 수 있습니다. 일반적으로 Secret 관리 서비스는 개별 컨테이너에 일회용 또는 시간 구분 키를 제공합니다. 항상 환경 변수가 아닌 중앙 위치에서 Secret을 가져오거나 탑재하십시오. 환경 변수는 로그 파일에 표시되어 민감한 데이터로 처리되지 않을 수 있습니다.

예: Kubernetes Secret

```
$ kubectl create secret generic mysecret \
--from-literal=username=admin \
--from-literal=password=p@ssw0rd

$ kubectl get secret mysecret -o yaml
apiVersion: v1
data:
  password: cEBzc3cwcmQK
  username: YWRtaW4=
kind: Secret
metadata:
  name: mysecret
  namespace: default
  ...
type: Opaque
```



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

이 예에서는 특정 키-값 쌍을 사용하여 **Secret** 객체를 명령형으로 생성합니다. 값은 **etcd** 데이터스토어에 저장되기 전에 **Kubernetes** 제어 영역에 의해 **base64**로 인코딩됩니다. 다음으로, **Secret** 객체(YAML로 표시됨)의 내용을 기록하고 객체와 그 내용이 **Kubernetes**에 있는지 확인합니다.

예: PodSpec의 Secret

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-secret-env
spec:
  containers:
    - name: pod-secret-env
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
        - name: SECRET_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
  restartPolicy: Never
```



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

Secret을 Pod 환경 변수로 사용하는 예입니다. ConfigMaps와 마찬가지로 Secret을 Pod 볼륨으로 사용하도록 선택할 수도 있습니다(일반적으로 더 안전한 것으로 여겨짐).

AWS Secrets Manager and Configuration Provider(ASCP)



- Amazon EKS에서 실행되는 Pod가 Secret에 액세스할 수 있습니다.
 - Pod 파일 시스템에 볼륨으로 탑재됨
 - Kubernetes Secret 리소스로 노출됨
- Secrets Manager 또는 SSM Parameter Store에서 Secret을 안전하게 저장 및 관리합니다.
- 자동 키 교체가 가능합니다.
- IRSA를 사용하는 IAM 정책으로 Secret 액세스를 특정 Pod로 제한합니다.

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

| 강사용 노트

| 이번이 아마도 이 과정에서 IRSA에 대한 첫 번째 언급일 것입니다. IRSA에 대해서는 다른 모듈에서 더 자세히 논의한다고 수강생에게 알려주십시오.

| 수강생용 노트

Secrets Manager의 Secret을 Amazon EKS Pod에 탑재된 파일로 표시하려면 Kubernetes Secrets Store CSI 드라이버용 AWS Secrets and Configuration Provider(ASCP)를 사용할 수 있습니다. ASCP는 Amazon EC2 노드 그룹을 실행하는 Amazon Elastic Kubernetes Service(Amazon EKS) 1.17 이상에서 작동합니다.

ASCP를 사용하면 Secrets Manager에서 Secret을 저장 및 관리하고 Amazon EKS에서 실행되는 워크로드를 통해 검색할 수 있습니다. Secret에 JSON 형식의 키/값 쌍이 여러 개 포함되어 있는 경우 Amazon EKS에 탑재할 쌍을 선택할 수 있습니다. ASCP는 JMESPath 구문을 사용하여 Secret의 키/값 쌍을 쿼리합니다. ASCP에 Parameter Store 파라미터를 사용할 수도 있습니다. JMESPath 구문에 대한 자세한 내용은 해당 웹 사이트(<https://jmespath.org/>)를 참조하십시오.

IAM 역할 및 정책을 사용하여 클러스터의 특정 Amazon EKS Pod에 Secret에 대한 액세스 권한을 부여합니다.

Amazon EKS Pod에서 생성할 파일과 여기에 삽입할 Secret을 설명하려면 SecretProviderClass YAML 파일을 생성합니다. SecretProviderClass는 참조되는 Amazon EKS Pod와 동일한 네임스페이스에 있어야 합니다.

프라이빗 Amazon EKS 클러스터를 사용하는 경우 클러스터가 상주하는 VPC에 Secrets Manager 엔드포인트가 있어야 합니다. Secrets Store CSI 드라이버는 엔드포인트를 사용하여 Secrets Manager를 호출합니다. VPC에서 엔드포인트를 생성하는 방법에 대한 자세한 내용은 VPC 엔드포인트 설명서 (<https://docs.aws.amazon.com/secretsmanager/latest/userguide/vpc-endpoint-overview.html>)를 참조하십시오.

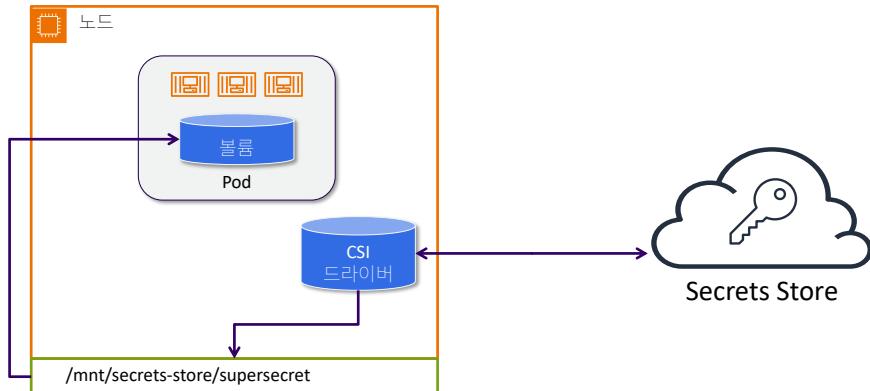
Secret에 Secrets Manager 자동 교체를 사용하는 경우 Secrets Store CSI 드라이버 교체 조정기 기능을 사용하여 Secrets Manager에서 최신 Secret을 검색할 수도 있습니다. 자세한 내용은 Auto rotation of mounted contents and synced Kubernetes Secrets(<https://secrets-store-csi-driver.sigs.k8s.io/topics/secret-auto-rotation.html>)를 참조하십시오.

ASCP를 다운로드하려면 <https://github.com/aws/secrets-store-csi-driver-provider-aws>를 방문하십시오.

ASCP 사용 방법에 대한 자습서는 자습서: Amazon EKS Pod에서 AWS Secrets Manager Secret 생성 및 탑재(https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_csi_driver_tutorial.html)를 참조하십시오.

참고: AWS Fargate 노드 그룹은 지원되지 않습니다.

Kubernetes Secrets Store CSI 드라이버를 사용하는 ASCP



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

| 강사용 노트

| 학습 편의상 사용된 탑재 경로가 완전히 정확하지는 않습니다. CSI 드라이버가 노드 내의 Pod에 볼륨을 탑재하고 있습니다.

| 참고로 생성된 볼륨은 EBS 볼륨이 아니지만 기본 Kubernetes/Linux 기능을 사용하고 있습니다.

| 수강생용 노트

이 슬라이드의 다이어그램은 ASCP 작동 방식의 예를 보여줍니다. CSI 드라이버를 설치하면 드라이버가 노드에 탑재됩니다. CSI 드라이버는 Secrets Store에서 Secret을 검색합니다. 그런 다음 Secret을 볼륨으로 탑재합니다.

자세한 내용은 <https://aws.amazon.com/blogs/security/how-to-use-aws-secrets-configuration-provider-with-kubernetes-secrets-store-csi-driver/>를 참조하십시오.

예: SecretProviderClass

```
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClass
metadata:
  name: nginx-deployment-aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "MySecret"
        objectType: "secretsmanager"
        objectAlias: "supersecret"
```



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

| 수강생용 노트

ASCP가 Amazon EKS에 파일 시스템 파일로 탑재할 Secret을 결정하려면 SecretProviderClass YAML 파일을 생성합니다. SecretProviderClass YAML은 탑재할 Secret과 이를 탑재할 파일 이름을 나열합니다. SecretProviderClass는 참조되는 Amazon EKS Pod와 동일한 네임스페이스에 있어야 합니다.

자세한 내용은

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_csi_driver_SecretProviderClass.html를 참조하십시오.

예: 이름 또는 ARN 기준으로 Secret 탑재

```
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "arn:aws:secretsmanager:us-east-2:111122223333:secret:MySecret2-
d4e5f6"
        - objectName: "MySecret3"
          objectType: "secretsmanager"
        - objectName: "MySecret4"
          objectType: "secretsmanager"
          objectVersionLabel: "AWSCURRENT"
```



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

| 수강생용 노트

다음 예에서는 Amazon EKS에 파일 3개를 탑재하는 SecretProviderClass를 보여줍니다.

1. 전체 ARN으로 지정된 Secret입니다.
2. 이름으로 지정된 Secret입니다.
3. 특정 버전의 Secret입니다.

예: 배포

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
spec:
  serviceAccountName: nginx-deployment-sa
  volumes:
  - name: secrets-store-inline
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "nginx-deployment-aws-secrets"
  containers:
  - name: nginx-deployment
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: secrets-store-inline
      mountPath: "/mnt/secrets-store"
      readOnly: true
```



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

| 수강생용 노트

이 슬라이드는 **Secret**을 사용하는 배포 yaml의 예를 보여줍니다.

지식 확인 1

Amazon EKS에서 Kubernetes Secret에 대한 기본 동작은 무엇입니까?

- | 보기 | 응답 |
|----|--|
| A. | Amazon EKS에서는 Secret이 지원되지 않습니다. |
| B. | Secret은 AES256을 사용하여 암호화됩니다. |
| C. | Secret은 base64로 인코딩되며 암호화된 볼륨으로 etcd에 저장됩니다. |
| D. | Secret은 KMS 키로 암호화됩니다. |

지식 확인 1: 정답은 C입니다.

Amazon EKS에서 Kubernetes Secret에 대한 기본 동작은 무엇입니까?

보기

응답

- A. Amazon EKS에서는 Secret이 지원되지 않습니다.
- B. Secret은 AES256을 사용하여 암호화됩니다.
- C. Secret은 base64로 인코딩되며 암호화된 볼륨으로 etcd에 저장됩니다.
- D. Secret은 KMS 키로 암호화됩니다.

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

정답은 C입니다. 기본적으로 전체 ETCD 볼륨은 DEK(데이터 암호화 키)로 암호화됩니다. 봉투 암호화를 활성화하면 각 Secret이 고유한 DEK로 암호화됩니다. AWS KMS는 KMS 키를 사용하여 DEK를 암호화하는 데 사용됩니다.

모듈 요약



이 모듈에서 학습한 내용:

- 스테이트풀 애플리케이션에는 데이터 지속성 요구 사항을 충족하는 스토리지 옵션이 필요합니다.
- **Kubernetes**에서는 영구 볼륨으로 독립적 스토리지를 요구합니다.
- 애플리케이션의 데이터 요구 사항을 가장 잘 충족하는 스토리지 클래스를 선택합니다.

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

| 강사용 노트

| 수강생용 노트



Running Containers on Amazon EKS

실습 5: Amazon EKS의 영구 스토리지



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

| 수강생용 노트

실습 5에서는 **Amazon EKS**에서 실행되는 애플리케이션을 위한 스토리지를
프로비저닝합니다.

실습 5



aws

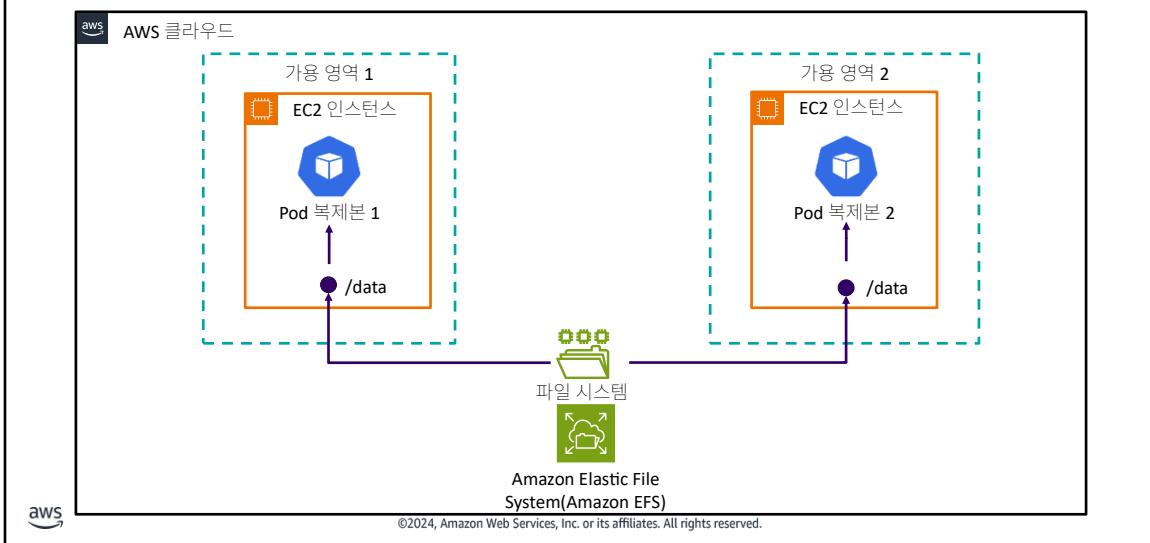
Amazon EKS의 영구 스토리지

이 실습에서는 다음 과제를 수행합니다.

- 사전 생성된 스토리지 클래스를 사용하여 스토리지를 소비하는 **PVC** 생성
- PVC를 사용하여 Pod에 스토리지 할당
- 스토리지 수명 주기 관리

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

실습 5 아키텍처 다이어그램



~Dev notes

~lab-5-results: Diagram of two pods across two AZs consuming persistent storage backed by Amazon EFS.



Running Containers on Amazon EKS

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.