요구 사항

Kubernetes clusters

NetApp
January 04, 2024

목차

요구 사항

AWS의 Kubernetes 클러스터 요구사항

AWS에서 관리되는 EKS(Amazon Elastic Kubernetes Service) 클러스터 또는 자체 관리되는 Kubernetes 클러스터를 BlueXP에 추가할 수 있습니다. 클러스터를 BlueXP에 추가하려면 다음 요구 사항을 충족해야 합니다.

이 항목에서는 EKS 및 자체 관리 Kubernetes 클러스터에 대한 구성이 동일한 _Kubernetes 클러스터를 사용합니다. 클러스터 유형은 구성이 다른 곳에서 지정됩니다.

요구 사항

아스트라 트리덴트

Astra Trident의 최신 버전 4개 중 하나가 필요합니다. BlueXP에서 Astra Trident를 직접 설치하거나 업그레이드할 수 있습니다. 당신은 해야 한다 "사전 요구 사항을 검토합니다" Astra Trident를 설치하기 전

Cloud Volumes ONTAP

Cloud Volumes ONTAP for AWS는 클러스터를 위한 백엔드 스토리지로 설정해야 합니다. "구성 단계를 보려면 Astra Trident 문서로 이동합니다".

BlueXP 커넥터

필요한 권한이 있는 Connector가 AWS에서 실행되고 있어야 합니다. 아래에서 자세히 알아보십시오.

네트워크 연결

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 Cloud Volumes ONTAP 사이에 네트워크 연결이 필요합니다. 아래에서 자세히 알아보십시오.

RBAC 인증

BlueXP Connector 역할은 각 Kubernetes 클러스터에서 인증되어야 합니다. 아래에서 자세히 알아보십시오.

커넥터를 준비합니다

Kubernetes 클러스터를 검색하고 관리하려면 AWS에 BlueXP Connector가 필요합니다. 새 Connector를 만들거나 필요한 권한이 있는 기존 Connector를 사용해야 합니다.

새 커넥터를 작성합니다

아래 링크 중 하나에 있는 단계를 따르십시오.

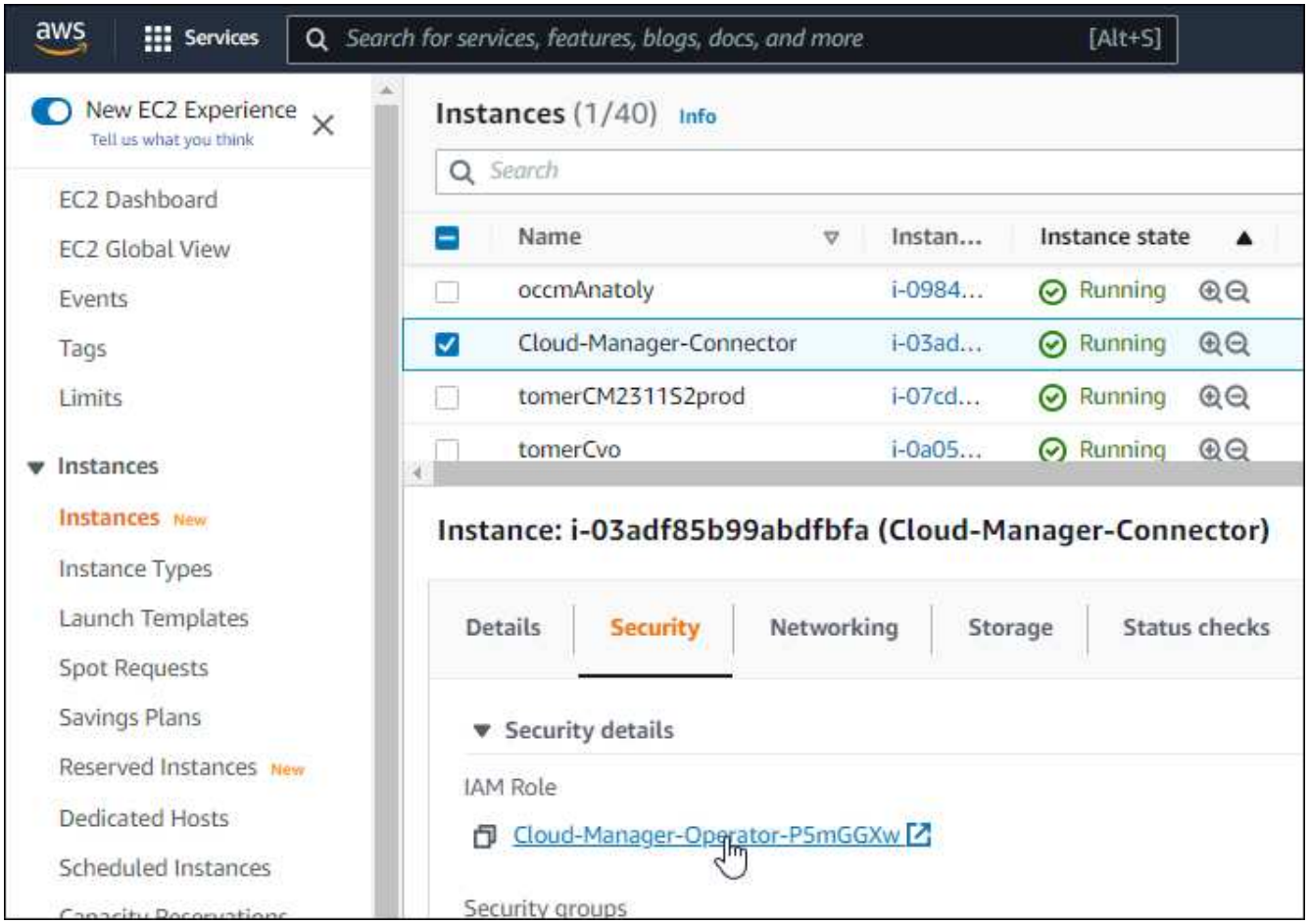
- "BlueXP에서 커넥터를 만듭니다" (권장)
- "AWS Marketplace에서 Connector를 생성합니다"
- "AWS의 기존 Linux 호스트에 커넥터를 설치합니다"

기존 **Connector**에 필요한 권한을 추가합니다

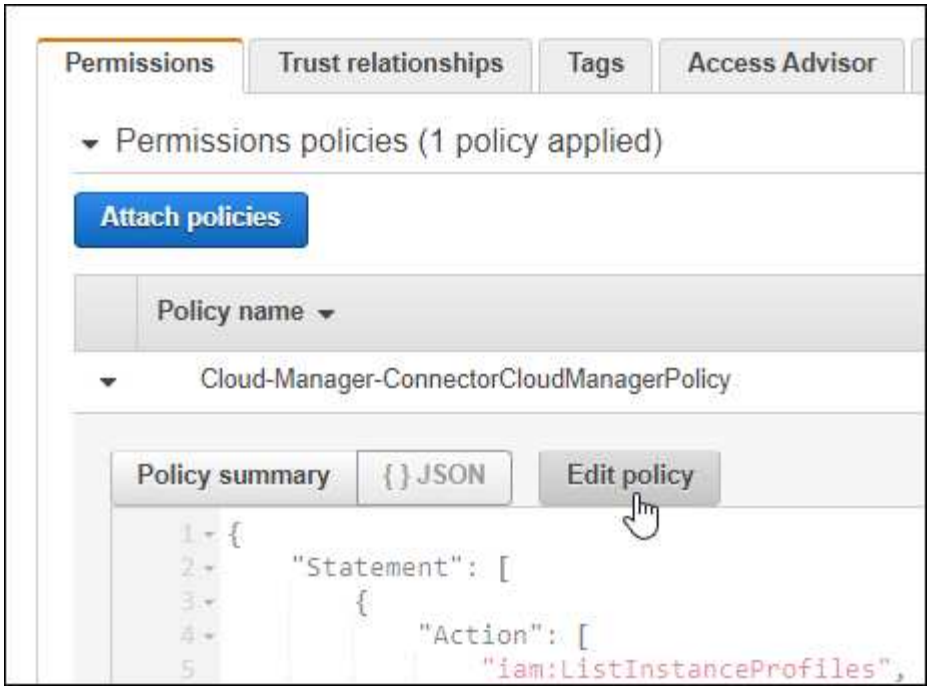
3.9.13 릴리스부터 _NEWED_DEPLOY 커넥터에는 Kubernetes 클러스터의 검색 및 관리를 지원하는 세 가지 새로운 AWS 권한이 포함되어 있습니다. 이 릴리스 전에 Connector를 생성한 경우, Connector의 IAM 역할에 대한 기존 정책을 수정하여 권한을 제공해야 합니다.

단계

1. AWS 콘솔로 이동하여 EC2 서비스를 엽니다.
2. Connector 인스턴스를 선택하고 * Security * 를 클릭한 다음 IAM 역할의 이름을 클릭하여 IAM 서비스의 역할을 확인합니다.



3. 사용 권한 * 탭에서 정책을 확장하고 * 정책 편집 * 을 클릭합니다.



4. JSON * 을 클릭하고 첫 번째 작업 세트에서 다음 권한을 추가합니다.

   - EC2: 설명
   - EKS: ListClusters
   - EKS: DescribeCluster
   - IAM:GetInstanceProfile 을 참조하십시오

   "정책의 전체 JSON 형식을 봅니다"

5. 정책 검토 * 를 클릭한 다음 * 변경 사항 저장 * 을 클릭합니다.

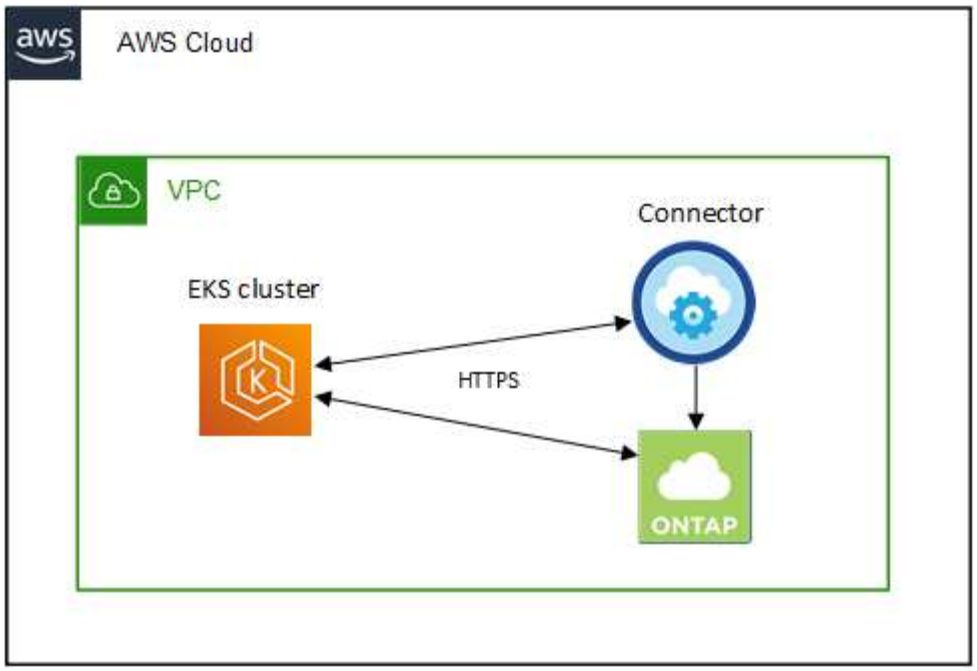
네트워킹 요구 사항을 검토합니다

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 클러스터에 백엔드 스토리지를 제공하는 Cloud Volumes ONTAP 시스템 간에 네트워크 연결을 제공해야 합니다.

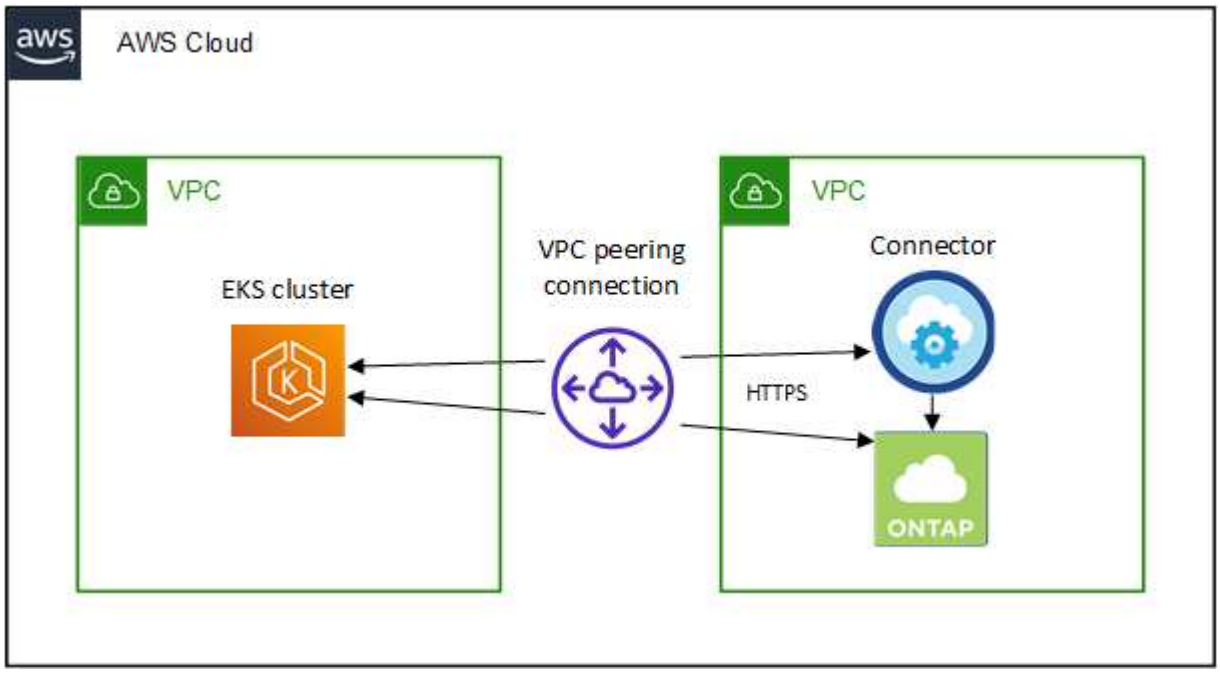
- 각 Kubernetes 클러스터에는 Connector로부터 인바운드 연결이 있어야 합니다
- Connector는 포트 443을 통해 각 Kubernetes 클러스터에 대한 아웃바운드 연결을 가지고 있어야 합니다

이 연결을 제공하는 가장 간단한 방법은 Kubernetes 클러스터와 같은 VPC에 Connector와 Cloud Volumes ONTAP를 구축하는 것입니다. 그렇지 않으면 다른 VPC 간에 VPC 피어링 연결을 설정해야 합니다.

다음은 동일한 VPC의 각 구성 요소를 보여 주는 예입니다.



이 또 다른 예는 다른 VPC에서 실행되는 EKS 클러스터를 보여 줍니다. 이 예에서 VPC 피어링은 EKS 클러스터용 VPC와 커넥터 및 Cloud Volumes ONTAP용 VPC 간에 연결을 제공합니다.



RBAC 승인을 설정합니다

Connector가 클러스터를 검색 및 관리할 수 있도록 각 Kubernetes 클러스터에서 Connector 역할을 승인해야 합니다.

다른 기능을 사용하려면 다른 권한이 필요합니다.

백업 및 복원

백업 및 복원에는 기본 인증만 필요합니다.

스토리지 클래스를 추가합니다

BlueXP를 사용하여 스토리지 클래스를 추가하고 클러스터에서 백엔드의 변경 사항을 모니터링하려면 확장된 인증이 필요합니다.

Astra 트리덴트 설치

Astra Trident를 설치하려면 BlueXP에 대한 전체 인증을 제공해야 합니다.

Astra Trident를 설치할 때 BlueXP는 Astra Trident가 스토리지 클러스터와 통신해야 하는 자격 증명이 포함된 Astra Trident 백엔드 및 Kubernetes 암호를 설치합니다.

단계

1. 클러스터 역할 및 역할 바인딩을 생성합니다.
   a. 요구 사항에 따라 권한 부여를 사용자 지정할 수 있습니다.

백업/복원

Kubernetes 클러스터의 백업 및 복원을 위한 기본 인증을 추가하십시오.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - namespaces
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumes
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - pods
        - pods/exec
      verbs:
        - get
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumeclaims
      verbs:
        - list
        - create
        - watch
    - apiGroups:
        - storage.k8s.io
      resources:
        - storageclasses
      verbs:
        - list
```

```
    - apiGroups:
        - trident.netapp.io
      resources:
        - tridentbackends
      verbs:
        - list
        - watch
    - apiGroups:
        - trident.netapp.io
      resources:
        - tridentorchestrators
      verbs:
        - get
        - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: Group
      name: cloudmanager-access-group
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

스토리지 클래스

BlueXP를 사용하여 저장소 클래스를 추가하려면 확장된 권한을 추가합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - secrets
        - namespaces
        - persistentvolumeclaims
        - persistentvolumes
        - pods
        - pods/exec
```

```
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: Group
      name: cloudmanager-access-group
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Trident 설치

명령줄을 사용하여 전체 인증을 제공하고 BlueXP에서 Astra Trident를 설치할 수 있도록 합니다.

```
eksctl create iamidentitymapping --cluster < > --region < > --arn
< > --group "system:masters" --username
system:node:{{EC2PrivateDNSName}}
```

b. 클러스터에 구성을 적용합니다.

```
kubectl apply -f <file-name>
```

2. 권한 그룹에 대한 ID 매핑을 만듭니다.

eksctl을 사용합니다

eksctl을 사용하여 클러스터와 BlueXP Connector의 IAM 역할 사이에 IAM ID 매핑을 생성합니다.

"전체 지침은 eksctl 설명서를 참조하십시오".

아래에 예가 나와 있습니다.

```
eksctl create iamidentitymapping --cluster <eksCluster> --region
<us-east-2> --arn <ARN of the Connector IAM role> --group
cloudmanager-access-group --username
system:node:{{EC2PrivateDNSName}}
```

AWS-auth를 편집합니다

AWS-auth ConfigMap을 직접 편집하여 BlueXP Connector의 IAM 역할에 RBAC 액세스를 추가합니다.

"전체 지침은 AWS EKS 설명서를 참조하십시오".

아래에 예가 나와 있습니다.

```
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - cloudmanager-access-group
      rolearn: <ARN of the Connector IAM role>
     username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2021-09-30T21:09:18Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1021"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: dcc31de5-3838-11e8-af26-02e00430057c
```

Azure의 Kubernetes 클러스터 요구사항

BlueXP를 사용하여 Azure에서 관리되는 Azure Kubernetes 클러스터(AKS) 및 자체 관리 Kubernetes 클러스터를 추가하고 관리할 수 있습니다. 클러스터를 BlueXP에 추가하려면 다음 요구 사항을 충족해야 합니다.

이 항목에서는 AKS 및 자체 관리 Kubernetes 클러스터의 구성이 동일한 _Kubernetes 클러스터를 사용합니다. 클러스터 유형은 구성이 다른 곳에서 지정됩니다.

요구 사항

아스트라 트리덴트

Astra Trident의 최신 버전 4개 중 하나가 필요합니다. BlueXP에서 Astra Trident를 직접 설치하거나 업그레이드할 수 있습니다. 당신은 해야 한다 "사전 요구 사항을 검토합니다" Astra Trident를 설치하기 전

Cloud Volumes ONTAP

Cloud Volumes ONTAP를 클러스터에 대한 백엔드 스토리지로 설정해야 합니다. "구성 단계를 보려면 Astra Trident 문서로 이동합니다".

BlueXP 커넥터

Connector는 필요한 권한을 사용하여 Azure에서 실행 중이어야 합니다. 아래에서 자세히 알아보십시오.

네트워크 연결

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 Cloud Volumes ONTAP 사이에 네트워크 연결이 필요합니다. 아래에서 자세히 알아보십시오.

RBAC 인증

BlueXP는 Active Directory를 사용하는 RBAC 지원 클러스터를 지원합니다. 각 Azure 클러스터마다 BlueXP Connector 역할이 허가되어야 합니다. 아래에서 자세히 알아보십시오.

커넥터를 준비합니다

Kubernetes 클러스터를 검색하고 관리하려면 Azure의 BlueXP Connector가 필요합니다. 새 Connector를 만들거나 필요한 권한이 있는 기존 Connector를 사용해야 합니다.

새 커넥터를 작성합니다

아래 링크 중 하나에 있는 단계를 따르십시오.

- "BlueXP에서 커넥터를 만듭니다" (권장)
- "Azure Marketplace에서 Connector를 생성합니다"
- "기존 Linux 호스트에 커넥터를 설치합니다"

기존 **Connector**에 필요한 사용 권한 추가(관리되는 **AKS** 클러스터 검색)

관리되는 AKS 클러스터를 검색하려면 Connector의 사용자 지정 역할을 수정하여 사용 권한을 제공해야 할 수 있습니다.

단계

1. Connector 가상 머신에 할당된 역할을 확인합니다.
   a. Azure 포털에서 가상 머신 서비스를 엽니다.
   b. Connector 가상 머신을 선택합니다.
   c. 설정에서 * ID * 를 선택합니다.

d. Azure 역할 할당 * 을 클릭합니다.

e. Connector 가상 머신에 할당된 사용자 지정 역할을 기록해 둡니다.

2. 사용자 지정 역할 업데이트:

a. Azure 포털에서 Azure 구독을 엽니다.

b. IAM(액세스 제어) > 역할 * 을 클릭합니다.

c. 사용자 지정 역할에 대한 줄임표(...)를 클릭한 다음 * 편집 * 을 클릭합니다.

d. JSON을 클릭하고 다음 권한을 추가합니다.

```
"Microsoft.ContainerService/managedClusters/listClusterUserCredential
/action"
"Microsoft.ContainerService/managedClusters/read"
```

e. 검토 + 업데이트 * 를 클릭한 다음 * 업데이트 * 를 클릭합니다.

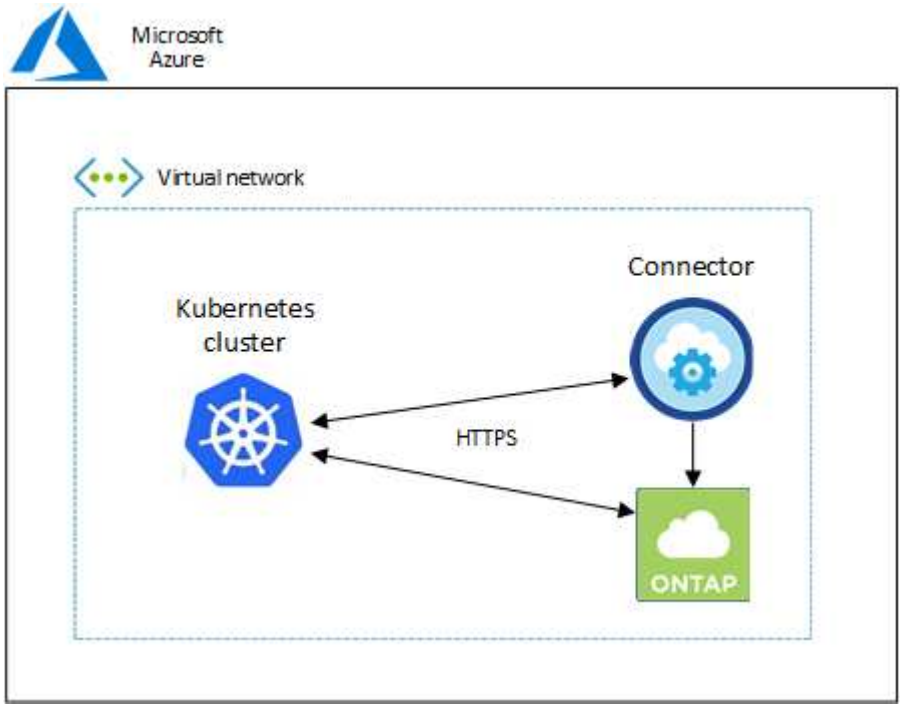
네트워킹 요구 사항을 검토합니다

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 클러스터에 백엔드 스토리지를 제공하는 Cloud Volumes ONTAP 시스템 간에 네트워크 연결을 제공해야 합니다.

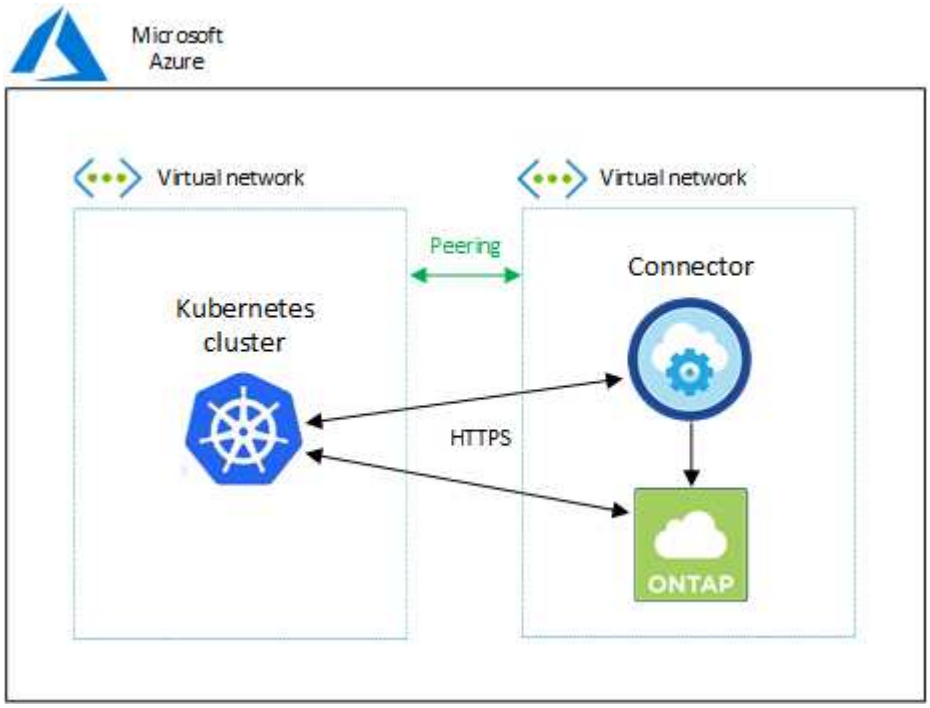
- 각 Kubernetes 클러스터에는 Connector로부터 인바운드 연결이 있어야 합니다
- Connector는 포트 443을 통해 각 Kubernetes 클러스터에 대한 아웃바운드 연결을 가지고 있어야 합니다

이 연결을 제공하는 가장 간단한 방법은 Kubernetes 클러스터와 같은 VNET에 Connector와 Cloud Volumes ONTAP를 구축하는 것입니다. 그렇지 않으면 다른 VNets 간의 피어링 연결을 설정해야 합니다.

다음은 동일한 VNET의 각 구성 요소를 보여 주는 예입니다.



그리고 다른 VNET에서 실행되는 Kubernetes 클러스터를 보여 주는 또 다른 예가 있습니다. 이 예에서 피어링은 Kubernetes 클러스터의 VNET와 커넥터 및 Cloud Volumes ONTAP용 VNET 간의 연결을 제공합니다.



RBAC 승인을 설정합니다

RBAC 검증은 AD(Active Directory)가 활성화된 Kubernetes 클러스터에서만 실행됩니다. AD를 사용하지 않는 Kubernetes 클러스터는 검증을 자동으로 통과합니다.

Connector가 클러스터를 검색 및 관리할 수 있도록 각 Kubernetes 클러스터에서 커넥터 역할을 승인해야 합니다.

백업 및 복원

백업 및 복원에는 기본 인증만 필요합니다.

스토리지 클래스를 추가합니다

BlueXP를 사용하여 스토리지 클래스를 추가하고 클러스터에서 백엔드의 변경 사항을 모니터링하려면 확장된 인증이 필요합니다.

Astra 트리덴트 설치

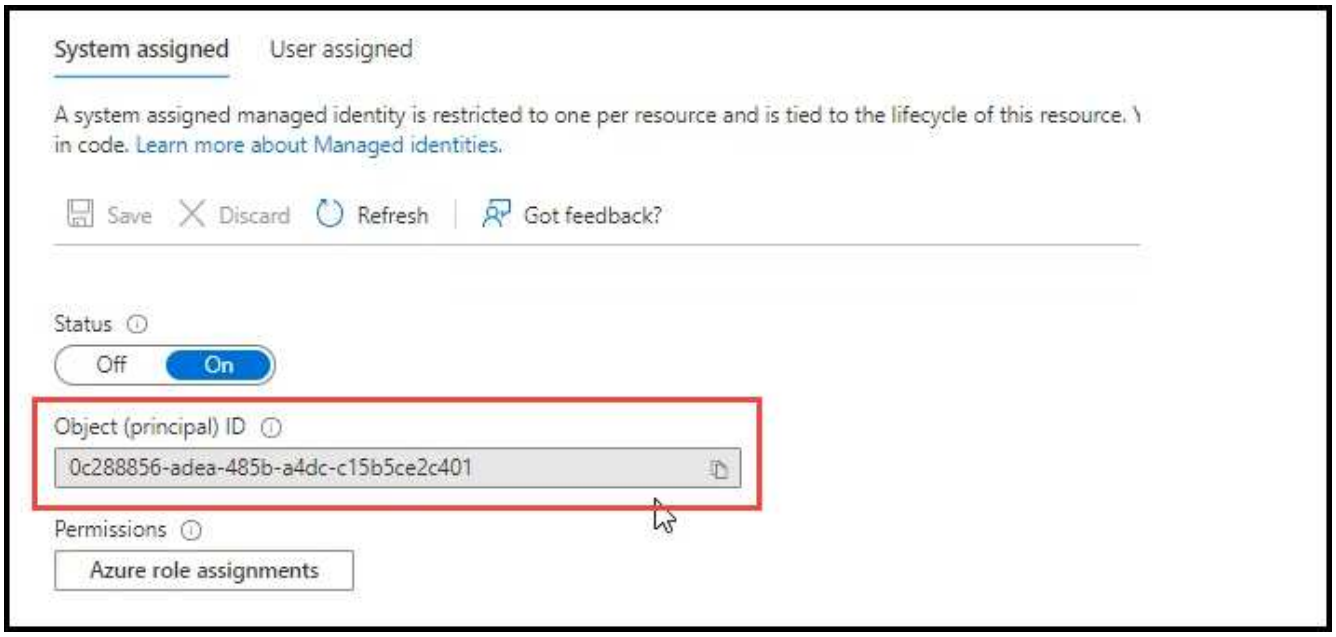
Astra Trident를 설치하려면 BlueXP에 대한 전체 인증을 제공해야 합니다.

Astra Trident를 설치할 때 BlueXP는 Astra Trident가 스토리지 클러스터와 통신해야 하는 자격 증명이 포함된 Astra Trident 백엔드 및 Kubernetes 암호를 설치합니다.

시작하기 전에

RBAC "프로젝트:이름:" 구성은 Kubernetes 클러스터 유형에 따라 약간 다릅니다.

- 관리되는 AKS 클러스터 * 를 배포하는 경우, Connector에 대해 시스템에서 할당한 관리 ID의 객체 ID가 필요합니다. 이 ID는 Azure 관리 포털에서 사용할 수 있습니다.



- 자체 관리되는 Kubernetes 클러스터 * 를 구축하는 경우 권한이 있는 사용자의 사용자 이름이 필요합니다.

단계

클러스터 역할 및 역할 바인딩을 생성합니다.

1. 요구 사항에 따라 권한 부여를 사용자 지정할 수 있습니다.

백업/복원

Kubernetes 클러스터의 백업 및 복원을 위한 기본 인증을 추가하십시오.

를 교체합니다 `subjects: kind:` 사용자 이름 및 을(를) 사용하여 변수를 지정합니다 `subjects: name:` 위에서 설명한 대로 시스템에서 할당한 관리 ID의 개체 ID 또는 권한이 부여된 사용자의 사용자 이름을 사용합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - namespaces
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumes
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - pods
        - pods/exec
      verbs:
        - get
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumeclaims
      verbs:
        - list
        - create
        - watch
    - apiGroups:
        - storage.k8s.io
```

```
      resources:
          - storageclasses
      verbs:
          - list
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
      verbs:
          - list
          - watch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentorchestrators
      verbs:
          - get
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

스토리지 클래스

BlueXP를 사용하여 저장소 클래스를 추가하려면 확장된 권한을 추가합니다.

를 교체합니다 `subjects: kind:` 사용자 이름 및 을(를) 사용하여 변수를 지정합니다 `subjects: user:` 위에서 설명한 대로 시스템에서 할당한 관리 ID의 개체 ID 또는 권한이 부여된 사용자의 사용자 이름을 사용합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
```

```
      resources:
          - secrets
          - namespaces
          - persistentvolumeclaims
          - persistentvolumes
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
```

```
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Trident 설치

명령줄을 사용하여 전체 인증을 제공하고 BlueXP에서 Astra Trident를 설치할 수 있도록 합니다.

```
eksctl create iamidentitymapping --cluster < > --region < > --arn <
> --group "system:masters" --username
system:node:{{EC2PrivateDNSName}}
```

2. 클러스터에 구성을 적용합니다.

```
kubectl apply -f <file-name>
```

Google Cloud의 Kubernetes 클러스터 요구사항

BlueXP를 사용하여 Google에서 관리되는 GKE(Google Kubernetes Engine) 클러스터와 자체 관리 Kubernetes 클러스터를 추가하고 관리할 수 있습니다. 클러스터를 BlueXP에 추가하려면 다음 요구 사항을 충족해야 합니다.

이 항목에서는 _Kubernetes cluster_를 사용합니다. 여기서 구성은 GKE 및 자체 관리되는 Kubernetes 클러스터의 경우 동일합니다. 클러스터 유형은 구성이 다른 곳에서 지정됩니다.

요구 사항

아스트라 트리덴트

Astra Trident의 최신 버전 4개 중 하나가 필요합니다. BlueXP에서 Astra Trident를 직접 설치하거나 업그레이드할 수 있습니다. 당신은 해야 한다 "사전 요구 사항을 검토합니다" Astra Trident를 설치하기 전

Cloud Volumes ONTAP

Cloud Volumes ONTAP는 Kubernetes 클러스터와 동일한 테넌시 계정, 작업 공간 및 커넥터 아래의 BlueXP에 있어야 합니다. "구성 단계를 보려면 Astra Trident 문서로 이동합니다".

BlueXP 커넥터

Connector는 필요한 권한으로 Google에서 실행 중이어야 합니다. 아래에서 자세히 알아보십시오.

네트워크 연결

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 Cloud Volumes ONTAP 사이에 네트워크 연결이 필요합니다. 아래에서 자세히 알아보십시오.

RBAC 인증

BlueXP는 Active Directory를 사용하는 RBAC 지원 클러스터를 지원합니다. 각 GKE 클러스터마다 BlueXP Connector 역할이 허가되어야 합니다. 아래에서 자세히 알아보십시오.

커넥터를 준비합니다

Kubernetes 클러스터를 검색하고 관리하려면 Google의 BlueXP Connector가 필요합니다. 새 Connector를 만들거나 필요한 권한이 있는 기존 Connector를 사용해야 합니다.

새 커넥터를 작성합니다

아래 링크 중 하나에 있는 단계를 따르십시오.

- "BlueXP에서 커넥터를 만듭니다" (권장)
- "기존 Linux 호스트에 커넥터를 설치합니다"

기존 **Connector**에 필요한 사용 권한 추가**(**관리되는 **GKE** 클러스터 검색**)**

관리되는 GKE 클러스터를 검색하려면 Connector의 사용자 지정 역할을 수정하여 권한을 제공해야 할 수 있습니다.

단계

1. 인치 "클라우드 콘솔"에서 * 역할 * 페이지로 이동합니다.
2. 페이지 맨 위에 있는 드롭다운 목록을 사용하여 편집할 역할이 포함된 프로젝트나 조직을 선택합니다.
3. 사용자 지정 역할을 클릭합니다.
4. 역할 편집 * 을 클릭하여 역할의 권한을 업데이트합니다.
5. 역할에 다음과 같은 새 권한을 추가하려면 * 권한 추가 * 를 클릭합니다.

```
container.clusters.get
container.clusters.list
```

6. Update * 를 클릭하여 편집된 역할을 저장합니다.

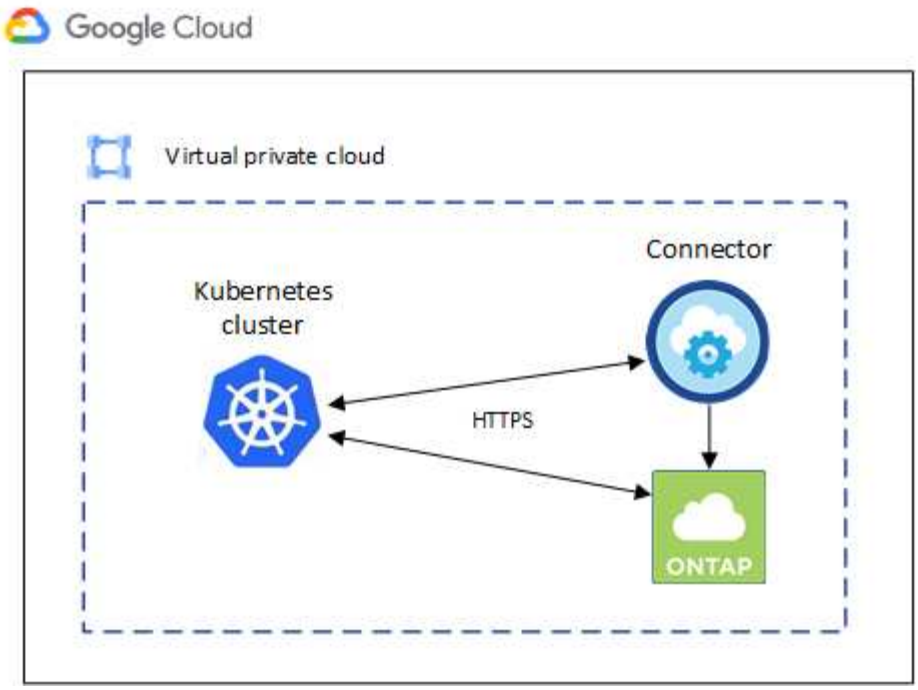
네트워킹 요구 사항을 검토합니다

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 클러스터에 백엔드 스토리지를 제공하는 Cloud Volumes ONTAP 시스템 간에 네트워크 연결을 제공해야 합니다.

- 각 Kubernetes 클러스터에는 Connector로부터 인바운드 연결이 있어야 합니다
- Connector는 포트 443을 통해 각 Kubernetes 클러스터에 대한 아웃바운드 연결을 가지고 있어야 합니다

이 연결을 제공하는 가장 간단한 방법은 Kubernetes 클러스터와 같은 VPC에 Connector와 Cloud Volumes ONTAP를 구축하는 것입니다. 그렇지 않으면 다른 VPC 간에 피어링 연결을 설정해야 합니다.

다음은 동일한 VPC의 각 구성 요소를 보여 주는 예입니다.



RBAC 승인을 설정합니다

RBAC 검증은 AD(Active Directory)가 활성화된 Kubernetes 클러스터에서만 실행됩니다. AD를 사용하지 않는 Kubernetes 클러스터는 검증을 자동으로 통과합니다.

Connector가 클러스터를 검색 및 관리할 수 있도록 각 Kubernetes 클러스터에서 커넥터 역할을 승인해야 합니다.

백업 및 복원

백업 및 복원에는 기본 인증만 필요합니다.

스토리지 클래스를 추가합니다

BlueXP를 사용하여 스토리지 클래스를 추가하고 클러스터에서 백엔드의 변경 사항을 모니터링하려면 확장된 인증이 필요합니다.

Astra 트리덴트 설치

Astra Trident를 설치하려면 BlueXP에 대한 전체 인증을 제공해야 합니다.

Astra Trident를 설치할 때 BlueXP는 Astra Trident가 스토리지 클러스터와 통신해야 하는 자격 증명이 포함된 Astra Trident 백엔드 및 Kubernetes 암호를 설치합니다.

시작하기 전에

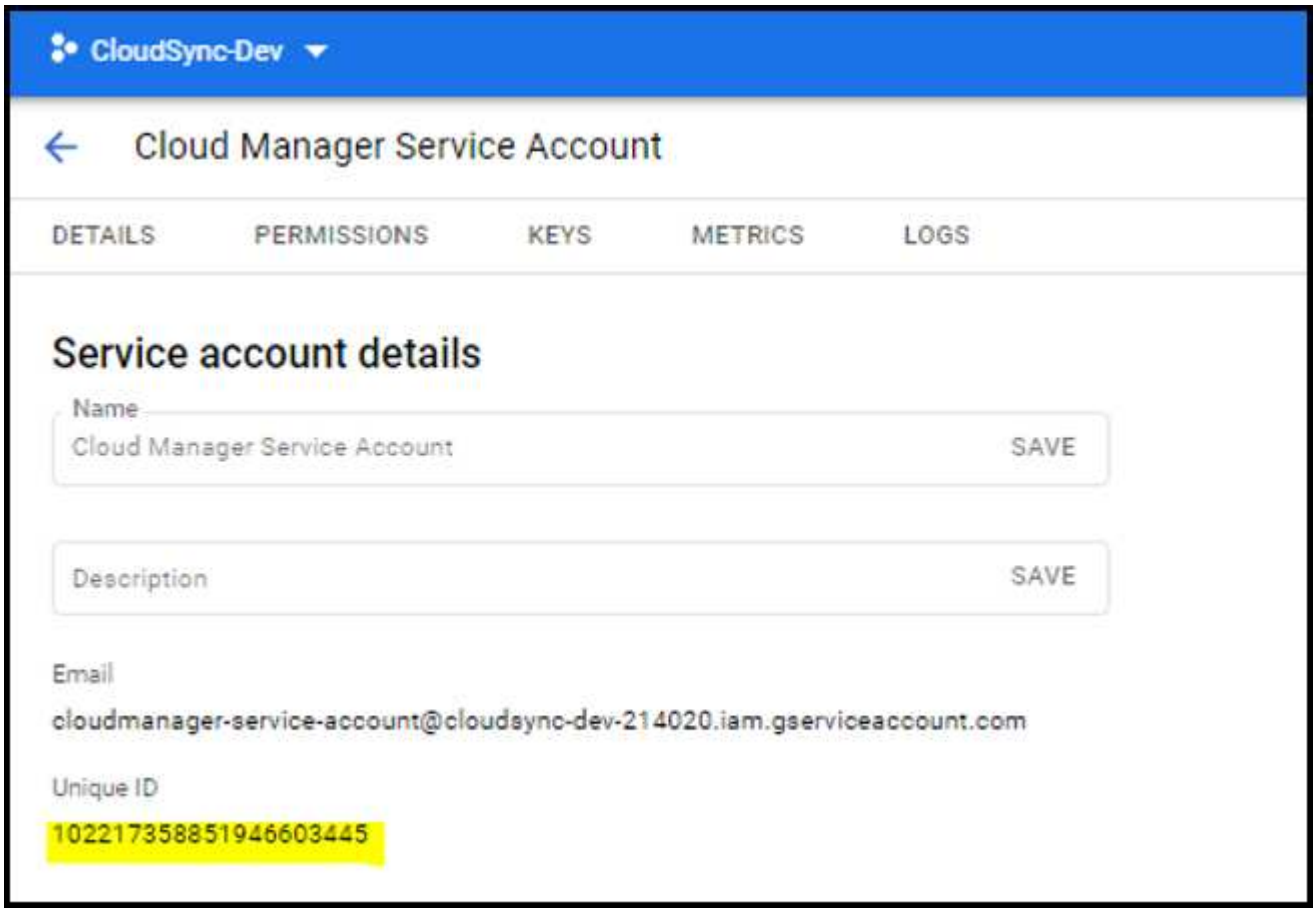
를 눌러 구성합니다 `subjects: name:` YAML 파일에서 BlueXP 고유 ID를 알아야 합니다.

고유 ID는 다음 두 가지 방법 중 하나로 찾을 수 있습니다.

- 명령 사용:

```
gcloud iam service-accounts list
gcloud iam service-accounts describe <service-account-email>
```

- 의 서비스 계정 세부 정보 를 클릭합니다 "클라우드 콘솔".



단계

클러스터 역할 및 역할 바인딩을 생성합니다.

1. 요구 사항에 따라 권한 부여를 사용자 지정할 수 있습니다.

백업/복원

Kubernetes 클러스터의 백업 및 복원을 위한 기본 인증을 추가하십시오.

를 교체합니다 `subjects: kind:` 사용자 이름 및 을(를) 사용하여 변수를 지정합니다 `subjects: name:` 인증된 서비스 계정의 고유 ID를 사용합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - namespaces
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumes
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - pods
        - pods/exec
      verbs:
        - get
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumeclaims
      verbs:
        - list
        - create
        - watch
    - apiGroups:
        - storage.k8s.io
      resources:
```

```
          - storageclasses
      verbs:
          - list
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
      verbs:
          - list
          - watch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentorchestrators
      verbs:
          - get
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

스토리지 클래스

BlueXP를 사용하여 저장소 클래스를 추가하려면 확장된 권한을 추가합니다.

를 교체합니다 `subjects: kind:` 사용자 이름 및 을(를) 사용하여 변수를 지정합니다 `subjects: user:` 인증된 서비스 계정의 고유 ID를 사용합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
```

```
            - secrets
            - namespaces
            - persistentvolumeclaims
            - persistentvolumes
            - pods
            - pods/exec
        verbs:
            - get
            - list
            - watch
            - create
            - delete
            - watch
    - apiGroups:
            - storage.k8s.io
        resources:
            - storageclasses
        verbs:
            - get
            - create
            - list
            - watch
            - delete
            - patch
    - apiGroups:
            - trident.netapp.io
        resources:
            - tridentbackends
            - tridentorchestrators
            - tridentbackendconfigs
        verbs:
            - get
            - list
            - watch
            - create
            - delete
            - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Trident 설치

명령줄을 사용하여 전체 인증을 제공하고 BlueXP에서 Astra Trident를 설치할 수 있도록 합니다.

```
kubectl create clusterrolebinding test --clusterrole cluster-admin
--user <Unique ID>
```

2. 클러스터에 구성을 적용합니다.

```
kubectl apply -f <file-name>
```

OpenShift의 Kubernetes 클러스터 요구사항

BlueXP를 사용하여 자체 관리되는 OpenShift Kubernetes 클러스터를 추가하고 관리할 수 있습니다. 클러스터를 BlueXP에 추가하려면 다음 요구 사항을 충족해야 합니다.

요구 사항

아스트라 트리덴트

Astra Trident의 최신 버전 4개 중 하나가 필요합니다. BlueXP에서 Astra Trident를 직접 설치하거나 업그레이드할 수 있습니다. 당신은 해야 한다 "사전 요구 사항을 검토합니다" Astra Trident를 설치하기 전

Cloud Volumes ONTAP

Cloud Volumes ONTAP를 클러스터에 대한 백엔드 스토리지로 설정해야 합니다. "구성 단계를 보려면 Astra Trident 문서로 이동합니다".

BlueXP 커넥터

Kubernetes 클러스터를 가져오고 관리하려면 BlueXP Connector가 필요합니다. 새 Connector를 만들거나 클라우드 공급자에 대한 필수 권한이 있는 기존 Connector를 사용해야 합니다.

- "AWS 커넥터"
- "Azure 커넥터"
- "Google Cloud Connector"

네트워크 연결

Kubernetes 클러스터와 Connector 간, Kubernetes 클러스터와 Cloud Volumes ONTAP 사이에 네트워크 연결이 필요합니다.

RBAC 권한이 있는 Kubernetes 구성 파일(kubecononfig)

OpenShift 클러스터를 가져오려면 다른 기능을 활성화하는 데 필요한 RBAC 권한이 있는 kubecononfig 파일이 필요합니다. kubecononfig 파일을 생성합니다.

- 백업 및 복원: 백업 및 복원에는 기본 인증만 필요합니다.
- 스토리지 클래스 추가: BlueXP를 사용하여 스토리지 클래스를 추가하고 클러스터에서 백엔드의 변경 사항을 모니터링하려면 확장된 인증이 필요합니다.
- Astra Trident 설치: BlueXP에 Astra Trident를 설치하려면 전체 인증을 제공해야 합니다.

Astra Trident를 설치할 때 BlueXP는 Astra Trident가 스토리지 클러스터와 통신해야 하는 자격 증명이 포함된 Astra Trident 백엔드 및 Kubernetes 암호를 설치합니다.

kubecononfig 파일을 생성합니다

OpenShift CLI를 사용하여 kubecon무화과 파일을 만들어 BlueXP로 가져옵니다.

단계

1. 관리 사용자가 있는 공용 URL에서 OC 로그인을 사용하여 OpenShift CLI에 로그인합니다.
2. 다음과 같이 서비스 계정을 생성합니다.

   a. OC-SERVICE-ACCOUNT.YAML이라는 서비스 계정 파일을 생성합니다.

      필요에 따라 이름 및 네임스페이스를 조정합니다. 여기에서 변경한 경우 다음 단계에서 동일한 변경 사항을 적용해야 합니다.

```
oc-service-account.yaml
```

+

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: oc-service-account
  namespace: default
```

   a. 서비스 계정 적용:

```
kubectl apply -f oc-service-account.yaml
```

3. 인증 요구 사항에 따라 사용자 지정 역할 바인딩을 만듭니다.

   a. OC-clusterrobinding.YAML이라는 ClusterRoleBinding 파일을 만듭니다.

```
oc-clusterrolebinding.yaml
```

b. 클러스터에 필요한 RBAC 승인을 구성합니다.

백업/복원

Kubernetes 클러스터의 백업 및 복원을 위한 기본 인증을 추가하십시오.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - namespaces
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumes
      verbs:
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - pods
        - pods/exec
      verbs:
        - get
        - list
        - watch
    - apiGroups:
        - ''
      resources:
        - persistentvolumeclaims
      verbs:
        - list
        - create
        - watch
    - apiGroups:
        - storage.k8s.io
      resources:
        - storageclasses
      verbs:
        - list
```

```
    - apiGroups:
        - trident.netapp.io
      resources:
        - tridentbackends
      verbs:
        - list
        - watch
    - apiGroups:
        - trident.netapp.io
      resources:
        - tridentorchestrators
      verbs:
        - get
        - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cloudmanager-access-clusterrole
subjects:
    - kind: ServiceAccount
      name: oc-service-account
      namespace: default
```

스토리지 클래스

BlueXP를 사용하여 저장소 클래스를 추가하려면 확장된 권한을 추가합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
        - ''
      resources:
        - secrets
        - namespaces
        - persistentvolumeclaims
        - persistentvolumes
        - pods
        - pods/exec
```

```
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cloudmanager-access-clusterrole
subjects:
    - kind: ServiceAccount
      name: oc-service-account
      namespace: default
```

Trident 설치

전체 관리자 권한을 부여하고 BlueXP에서 Astra Trident를 설치할 수 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cloudmanager-access-clusterrole
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: oc-service-account
  namespace: default
```

c. 클러스터 역할 바인딩을 적용합니다.

```
kubectl apply -f oc-clusterrolebinding.yaml
```

4. '<context>'을(를) 설치에 적합한 컨텍스트로 대체하여 서비스 계정 암호를 나열합니다.

```
kubectl get serviceaccount oc-service-account --context <context>
--namespace default -o json
```

출력의 끝은 다음과 유사합니다.

```
"secrets": [
{ "name": "oc-service-account-dockercfg-vhz87"},
{ "name": "oc-service-account-token-r59kr"}
]
```

제탑 배열의 각 요소에 대한 지수는 0으로 시작합니다. 위의 예에서 OC-SERVICE-ACCOUNT-dockercfg-vhz87 인덱스는 0이고 OC-SERVICE-ACCOUNT-TOKEN-r59kr 인덱스는 1입니다. 출력에서 "token"이라는 단어가 포함된 서비스 계정 이름의 인덱스를 기록해 둡니다.

5. 다음과 같이 kubecononfig를 생성합니다.

a. create-kubecononfig.sh 파일을 만듭니다. 다음 스크립트 시작 부분의 token_index를 올바른 값으로 바꿉니다.

```
create-kubeconfig.sh
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=oc-service-account
NAMESPACE=default
NEW_CONTEXT=oc
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
```

```
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

```
source create-kubeconfig.sh
```

결과

결과를 사용합니다 `kubeconfig-sa` 파일 - BlueXP에 OpenShift 클러스터를 추가할 수 있습니다.

저작권 정보

상표 정보