Trident 연산자를 사용하여 구축

Astra Trident

NetApp
April 16, 2024

목차

Trident 연산자를 사용하여 구축

Trident 연산자를 사용하여 Astra Trident를 배포할 수 있습니다.

Astra Trident 22.10에 대한 중요 정보입니다

- Astra Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다. *

strong>Astra Trident 22.10에 대한 중요 정보****

- 이제 Trident에서 Kubernetes 1.25가 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Astra Trident 22.10으로 업그레이드해야 합니다.
- Astra Trident는 이제 SAN 환경에서 다중 경로 구성을 사용하도록 엄격히 적용되며 권장값은 입니다 `find_multipaths: no` 다중 경로 .conf 파일

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 운영자 구축 옵션

Trident 연산자는 다음 두 가지 방법 중 하나로 배포할 수 있습니다.

- Trident 사용 "Helm 차트": 제어 도표는 Trident 연산자를 배포하고 Trident를 한 번에 설치합니다.
- 수동: Trident 연산자를 설치하고 관련 개체를 만드는 데 사용할 수 있는 파일을 제공합니다.
  - Kubernetes 1.24 이하 를 실행하는 클러스터의 경우, 를 사용합니다 "Bundle_PRE_1_25.YAML".
  - Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 "Bundle_post_1_25.YAML".

에 아직 익숙하지 않은 경우 "기본 개념"이제 아주 좋은 시간입니다.

필수 구성 요소를 확인합니다

Astra Trident를 구축하려면 다음과 같은 사전 요구 사항을 충족해야 합니다.

- 지원되는 Kubernetes 버전을 실행 중인 지원되는 Kubernetes 클러스터에 대한 모든 권한이 있습니다. 를 검토합니다 "요구 사항".
- 지원되는 NetApp 스토리지 시스템에 액세스할 수 있습니다.
- 모든 Kubernetes 작업자 노드에서 볼륨을 마운트할 수 있습니다.
- 에 Linux 호스트가 있습니다 `kubectl` (또는 `oc`, OpenShift를 사용하는 경우) 사용하려는 Kubernetes 클러스터를 관리하도록 설치 및 구성한 것입니다.
- 을(를) 설정했습니다 `KUBECONFIG` Kubernetes 클러스터 구성을 가리키는 환경 변수
- 을(를) 활성화했습니다 "Astra Trident에서 요구하는 기능 게이트".

- Docker Enterprise와 함께 Kubernetes를 사용하는 경우, "다음 단계에 따라 CLI 액세스를 설정합니다".

다 잡았나요? 좋습니다! 그럼 이제 시작하겠습니다.

Trident 연산자를 구축하고 Hrom을 사용하여 Astra Trident를 설치합니다

나열된 단계를 수행하여 Helm을 사용하여 Trident 연산자를 배포합니다.

필요한 것

위에 나열된 필수 구성 요소 외에도 Hrom을 사용하여 Trident 연산자를 구축하려면 다음이 필요합니다.

- A "지원되는 Kubernetes 버전"
- Helm 버전 3

단계

1. Trident의 제어 리포지토리 추가:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 를 사용합니다 `helm install` 명령을 입력하고 배포 이름을 지정합니다. 다음 예를 참조하십시오.

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace <trident-namespace>
```

> Trident에 대한 네임스페이스를 이미 만든 경우 를 참조하십시오 `--create-namespace` 매개 변수는 추가 네임스페이스를 만들지 않습니다.

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

- `--values` (또는 `-f`): 재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.
- `--set`: 명령줄에 재정의를 지정합니다.

예를 들어, 의 기본값을 변경합니다 `debug`에서 다음을 실행합니다 `--set` 명령:

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace --set tridentDebug=true
```

를 클릭합니다 `values.yaml` 제어 차트의 일부인 파일 은 키 목록과 해당 기본값을 제공합니다.

`helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등

Trident 연산자를 수동으로 구축하고 Trident를 설치합니다

나열된 단계를 수행하여 Trident 연산자를 수동으로 배포합니다.

1단계: Kubernetes 클러스터 검증

가장 먼저 해야 할 일은 Linux 호스트에 로그인하여 해당 호스트가 관리 중인 _ 작업 _ 을(를) 관리하고 있는지 확인하는 것입니다. "지원되는 Kubernetes 클러스터" 에 필요한 권한이 있어야 합니다.

OpenShift에서는 을 사용합니다 `oc` 대신 `kubectl` 다음 모든 예에서 를 실행하여 먼저 * system:admin * 으로 로그인합니다 `oc login -u system:admin` 또는 `oc login -u kube-admin`.

Kubernetes 버전을 확인하려면 다음 명령을 실행합니다.

```
kubectl version
```

Kubernetes 클러스터 관리자 권한이 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl auth can-i '*' '*' --all-namespaces
```

Docker Hub에서 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

2단계: 운영자를 다운로드하고 설정합니다

21.01부터 Trident 연산자는 클러스터 범위입니다. Trident 연산자를 사용하여 Trident를 설치하려면 을 만들어야 합니다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD) 및 기타 리소스 정의 Astra Trident를 설치하기 전에 다음 단계를 수행하여 운영자를 설정해야 합니다.

1. 에서 최신 버전의 Trident 설치 프로그램 번들을 다운로드하여 압축을 풉니다 "GitHub의 _Assets_섹션".

```
wget
https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

2. 적절한 CRD 매니페스트를 사용하여 를 만듭니다 `TridentOrchestrator` CRD 그런 다음 을 만듭니다 `TridentOrchestrator` 나중에 사용자 지정 리소스를 사용하여 운영자가 설치를 인스턴스화합니다.

다음 명령을 실행합니다.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 를 누릅니다 `TridentOrchestrator` CRD가 생성되면 운용자 배치에 필요한 다음과 같은 리소스를 생성한다.
   - 연산자를 위한 ServiceAccount입니다
   - ServiceAccount에 대한 ClusterRole 및 ClusterRoleBinding
   - 전용 PodSecurityPolicy
   - 작업자 자체

     Trident 설치 프로그램에는 이러한 리소스를 정의하는 매니페스트가 포함되어 있습니다. 기본적으로 연산자는 에 배포됩니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음 매니페스트를 사용하여 네임스페이스를 만듭니다.

```
kubectl apply -f deploy/namespace.yaml
```

4. 기본 네임스페이스 이외의 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스에서 을 업데이트해야 합니다 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을(를) 확인하고 생성합니다 `bundle.yaml`.

   다음 명령을 실행하여 YAML 매니페스트를 업데이트하고 을 생성합니다 `bundle.yaml` 를 사용합니다 `kustomization.yaml`:

```
kubectl kustomize deploy/ > deploy/bundle.yaml
```

   다음 명령을 실행하여 리소스를 생성하고 연산자를 배포합니다.

```
kubectl create -f deploy/bundle.yaml
```

5. 배치한 후 작업자의 상태를 확인하려면 다음을 수행합니다.

```
kubectl get deployment -n <operator-namespace>

NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           3m
```

```
kubectl get pods -n <operator-namespace>

NAME                                READY   STATUS              RESTARTS
AGE
trident-operator-54cb664d-lnjxh   1/1     Running             0
3m
```

운영자 배포는 클러스터의 작업자 노드 중 하나에서 실행되고 있는 포드를 성공적으로 생성합니다.

> Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Trident 연산자의 여러 배포를 생성하지 마십시오.

3단계: 작성 `TridentOrchestrator` Trident를 설치합니다

이제 연산자를 사용하여 Astra Trident를 설치할 준비가 되었습니다! 이 작업을 수행하려면 을 만들어야 합니다 `TridentOrchestrator`. Trident 설치 프로그램에는 작성을 위한 예제 정의가 포함되어 있습니다 `TridentOrchestrator`. 그러면 에서 설치가 시작됩니다 `trident` 네임스페이스.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident
Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:22.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Trident 연산자를 사용하면 의 특성을 사용하여 Astra Trident가 설치되는 방식을 사용자 지정할 수 있습니다 `TridentOrchestrator` 사양 을 참조하십시오 "Trident 구축을 사용자 지정합니다".

의 상태 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다.

상태	설명
설치 중	이 옵션을 사용하여 Astra Trident를 설치합니다 `TridentOrchestrator` 있습니다.
설치되어 있습니다	Astra Trident가 성공적으로 설치되었습니다.
제거 중	그 이유는 운영자가 Astra Trident를 제거하는 중입니다 `spec.uninstall=true`.
제거되었습니다	Astra Trident가 제거되었습니다.
실패했습니다	운영자가 Astra Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 이 상태에서 자동으로 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	를 클릭합니다 `TridentOrchestrator` 사용되지 않습니다. 다른 파일이 이미 있습니다.

설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운용자가 자체적으로 복구할 수 없는 경우 운용자의 로그를 확인해야 한다. 를 참조하십시오 "문제 해결" 섹션을 참조하십시오.

생성된 포드를 살펴보고 Astra Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pod -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0          1m
trident-csi-mr6zc                   2/2     Running   0          1m
trident-csi-xrp7w                   2/2     Running   0          1m
trident-csi-zh2jt                   2/2     Running   0          1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0          3m
```

을 사용할 수도 있습니다 `tridentctl` 설치된 Astra Trident의 버전을 확인합니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

이제 백엔드를 생성할 수 있습니다. 을 참조하십시오 "구축 후 작업".

배포 중 문제 해결에 대한 자세한 내용은 을 참조하십시오 "문제 해결" 섹션을 참조하십시오.

Trident 운영자 배포를 사용자 지정합니다

Trident 운영자는 의 특성을 사용하여 Astra Trident 설치를 사용자 지정할 수 있습니다 `TridentOrchestrator` 사양

설치를 사용자 지정하려면 다음을 선택합니다 `TridentOrchestrator` 인수를 사용하면 사용을 고려해야 합니다 `tridentctl` 필요에 따라 수정할 수 있는 사용자 지정 YAML 매니페스트를 생성합니다.

`spec.namespace` 에 지정됩니다 `TridentOrchestrator` Astra Trident가 설치된 네임스페이스를 나타냅니다. Astra Trident가 설치된 후에는 이 매개 변수 * 를 업데이트할 수 없습니다. 이렇게 하려고 하면 가 발생합니다 `TridentOrchestrator` 변경할 상태입니다 `Failed`. Astra Trident는 네임스페이스 간에 마이그레이션되지 않습니다.

구성 옵션

이 표에 자세히 나와 있습니다 `TridentOrchestrator` 특성:

매개 변수	설명	기본값
`namespace`	Astra Trident를 설치할 네임스페이스입니다	"기본값"
`debug`	Astra Trident에 대한 디버깅을 활성화합니다	거짓
`windows`	를 로 설정합니다 `true` Windows 작업자 노드에 설치할 수 있습니다.	거짓
`IPv6`	IPv6를 통해 Astra Trident를 설치합니다	거짓
`k8sTimeout`	Kubernetes 작업 시간이 초과되었습니다	30초
`silenceAutosupport`	AutoSupport 번들을 NetApp에 자동으로 보내지 않습니다	거짓
`enableNodePrep`	작업자 노드 종속성 자동 관리(* beta*)	거짓
`autosupportImage`	AutoSupport 텔레메트리 컨테이너 이미지입니다	"NetApp/트리덴트 - AutoSupport: 22.10.0"
`autosupportProxy`	AutoSupport 텔레메트리 전송을 위한 프록시의 주소/포트입니다	"http://proxy.example.com:8888"
`uninstall`	Astra Trident를 제거하는 데 사용되는 플래그입니다	거짓
`logFormat`	사용할 Astra Trident 로깅 형식[text,json]	"텍스트"
`tridentImage`	설치할 Astra Trident 이미지	"NetApp/트리덴트: 21.04"

매개 변수	설명	기본값
`imageRegistry`	형식의 내부 레지스트리 경로입니다 `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage(k8s 1.19 이상) 또는 quay.io/k8scsi"
`kubeletDir`	호스트의 kubelet 디렉토리에 대한 경로입니다	“/var/lib/kubelet”
`wipeout`	Astra Trident를 완전히 제거하기 위해 삭제할 리소스 목록입니다	
`imagePullSecrets`	내부 레지스트리에서 이미지를 가져올 수 있는 비밀	
`controllerPluginNodeSelector`	Trident 컨트롤러 CSI 플러그인을 실행하는 Pod용 추가 노드 선택기. pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`controllerPluginTolerations`	Trident 컨트롤러 CSI 플러그인을 실행하는 Pod의 허용 설정을 재정의합니다. pod.spec.Tolerations와 같은 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginNodeSelector`	Trident Node CSI 플러그인을 실행하는 Pod용 추가 노드 선택기 pod.spec.nodeSelector 과 동일한 형식을 따릅니다.	기본값 없음, 선택 사항
`nodePluginTolerations`	Trident Node CSI 플러그인을 실행하는 Pod의 허용 설정을 재정의합니다. pod.spec.Tolerations와 같은 형식을 따릅니다.	기본값 없음, 선택 사항

포드 매개 변수 포맷에 대한 자세한 내용은 을 참조하십시오 "노드에 Pod 할당".

샘플 구성

정의할 때 위에서 언급한 속성을 사용할 수 있습니다 `TridentOrchestrator` 를 눌러 설치를 사용자 정의합니다.

예 **1:** 기본 사용자 정의 구성

다음은 기본 사용자 지정 구성의 예입니다.

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

예 **2:** 노드 선택기를 사용하여 배포

이 예제에서는 노드 선택기를 사용하여 Trident를 배포하는 방법을 보여 줍니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

예 **3: Windows** 작업자 노드에 배포

이 예제에서는 Windows 작업자 노드에 대한 배포를 보여 줍니다.

```
$ cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```