Implementierung mit Trident Operator

Astra Trident

NetApp
May 23, 2023

Inhaltsverzeichnis

Implementierung mit Trident Operator

Sie können Astra Trident über den Trident-Operator implementieren. Es gibt zwei Möglichkeiten zur Implementierung des Trident Operators:

- Verwendung von Trident "Steuerruderdiagramm": Das Helm Chart implementiert den Trident-Operator und installiert Trident in einem Schritt.
- Manuell: Trident bietet ein "Bundle.yaml" Datei, mit der der Operator installiert und zugehörige Objekte erstellt werden können.

Wenn Sie sich nicht bereits mit dem vertraut gemacht haben "Grundkonzepte", Ist jetzt eine tolle Zeit, um das zu tun.

Was Sie benötigen

Bei der Implementierung von Astra Trident sollten die folgenden Voraussetzungen erfüllt sein:

- Sie haben volle Berechtigungen für einen unterstützten Kubernetes-Cluster mit Kubernetes 1.18 - 1.24.
- Sie haben Zugriff auf ein unterstütztes NetApp Storage-System.
- Sie können Volumes von allen Kubernetes-Worker-Nodes einbinden.
- Sie verfügen über einen Linux-Host mit `kubectl` (Oder `oc`, Falls Sie OpenShift nutzen) ist installiert und konfiguriert, um den Kubernetes-Cluster zu managen, den Sie verwenden möchten.
- Sie haben die festgelegt `KUBECONFIG` Umgebungsvariable auf die Kubernetes-Cluster-Konfiguration verweisen.
- Sie haben das aktiviert "Funktionsgates erforderlich von Astra Trident".
- Bei Verwendung von Kubernetes mit Docker Enterprise "Führen Sie die entsprechenden Schritte aus, um den CLI-Zugriff zu aktivieren".

Hast du das alles? Sehr Gut! Fangen wir an.

Setzen Sie den Trident-Operator ein und installieren Sie Astra Trident mit Helm

Führen Sie die aufgeführten Schritte zur Implementierung des Trident-Bedieners mithilfe von Helm durch.

Was Sie benötigen

Zusätzlich zu den oben aufgeführten Voraussetzungen benötigen Sie zur Implementierung des Trident-Operators mithilfe von Helm folgende Voraussetzungen:

- Kubernetes 1.18 - 1.24
- Helm Version 3

Schritte

1. Helm-Repository von Trident hinzufügen:

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

2. Verwenden Sie die `helm install` Führen Sie einen Befehl aus, und geben Sie einen Namen für Ihre Bereitstellung an. Das folgende Beispiel zeigt:

```
helm install <release-name> netapp-trident/trident-operator --version
22.4.0 --namespace <trident-namespace>
```

> Falls Sie noch keinen Namespace für Trident erstellt haben, können Sie den hinzufügen `--create-namespace` Parameter für das `helm install` Befehl. Helm erstellt dann automatisch den Namespace für Sie.

Während der Installation gibt es zwei Möglichkeiten, die Konfigurationsdaten zu übergeben:

- `--values` (Oder `-f`): Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die rechteste Datei hat Vorrang.
- `--set`: Überschreibungen auf der Kommandozeile angeben.

Um beispielsweise den Standardwert von zu ändern `debug`, Ausführen Sie das folgende `--set` Befehl:

```
$ helm install <name> netapp-trident/trident-operator --version 22.4.0
--set tridentDebug=true
```

Der `values.yaml` Die Datei, die Teil des Helm-Diagramms ist, enthält die Liste der Schlüssel und ihre Standardwerte.

`helm list` Zeigt Ihnen Details zur Installation an, z. B. Name, Namespace, Diagramm, Status, App-Version, Revisionsnummer usw.

Setzen Sie den Trident-Operator manuell ein

Führen Sie die aufgeführten Schritte aus, um den Trident-Operator manuell zu implementieren.

Schritt: Qualifizieren Sie Ihren Kubernetes-Cluster

Zunächst müssen Sie sich beim Linux-Host anmelden und überprüfen, ob es ein *Working*, "Unterstützter Kubernetes-Cluster" Dass Sie über die erforderlichen Berechtigungen verfügen.

> Mit OpenShift, verwenden `oc` Statt `kubectl` In allen folgenden Beispielen, und melden Sie sich als **System:admin** zuerst mit dem Ausführen an `oc login -u system:admin` Oder `oc login -u kube-admin`.

Führen Sie den folgenden Befehl aus, um Ihre Kubernetes-Version zu überprüfen:

```
kubectl version
```

So zeigen Sie, ob Sie über die Berechtigungen für Kubernetes Cluster-Administratoren verfügen:

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

Führen Sie den folgenden Befehl aus, um zu überprüfen, ob ein POD mit einem Image aus dem Docker Hub gestartet werden kann und das Storage-System über das POD-Netzwerk erreichen kann:

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

Schritt 2: Laden Sie den Operator herunter und richten Sie ihn ein

Ab 21.01 ist der Trident Operator der Cluster-Umfang. Zur Installation von Trident muss mit dem Trident-Operator der erstellt werden `TridentOrchestrator` Benutzerdefinierte Ressourcendefinition (CRD) und Definition anderer Ressourcen. Führen Sie diese Schritte durch, um den Bediener einzurichten, bevor Sie Astra Trident installieren können.

1. Laden Sie die neueste Version der herunter "Trident Installationspaket" Aus dem Abschnitt *Downloads* extrahieren.

   ```
   wget
   https://github.com/NetApp/trident/releases/download/v22.04.0/trident-
   installer-22.04.0.tar.gz
   tar -xf trident-installer-22.04.0.tar.gz
   cd trident-installer
   ```

2. Verwenden Sie das entsprechende CRD-Manifest, um das zu erstellen `TridentOrchestrator` CRD.- Dann erstellen Sie ein `TridentOrchestrator` Benutzerdefinierte Ressource später, um eine Installation durch den Bediener zu erstellen.

   Führen Sie den folgenden Befehl aus:

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

3. Nach dem `TridentOrchestrator` CRD wird erstellt, so legen Sie die folgenden Ressourcen für die Bedienerbereitstellung an:

   - Ein Servicekonto für den Betreiber
   - ClusterRPole und ClusterRoleBending an das ServiceAccount
   - Eine dedizierte PodSecurityPolicy
   - Der Bediener selbst

     Trident Installer enthält Manifeste für die Definition dieser Ressourcen. Standardmäßig wird der Operator in bereitgestellt `trident` Namespace. Wenn der `trident` Der Namespace ist nicht vorhanden. Verwenden Sie das folgende Manifest, um einen zu erstellen.

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

4. So stellen Sie den Operator in einem anderen Namespace als dem Standard bereit `trident` Namespace, sollten Sie den aktualisieren `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Manifeste und Generate Your `bundle.yaml`.

   Führen Sie den folgenden Befehl aus, um die YAML Manifeste zu aktualisieren und das zu generieren `bundle.yaml` Verwenden der `kustomization.yaml`:

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

   Führen Sie den folgenden Befehl aus, um die Ressourcen zu erstellen und den Operator bereitzustellen:

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

5. Gehen Sie wie folgt vor, um den Status des Bedieners nach der Bereitstellung zu überprüfen:

   ```
   $ 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
   ```

Durch die Implementierung eines Mitarbeiters wird erfolgreich ein Pod erstellt, der auf einem der Worker-Nodes im Cluster ausgeführt wird.

> Es sollte nur eine Instanz* des Operators in einem Kubernetes-Cluster geben. Erstellen Sie nicht mehrere Implementierungen des Trident-Operators.

Schritt 3: Erstellen `TridentOrchestrator` Und Trident installieren

Sie können Astra Trident nun mit dem Operator installieren! Hierfür muss erstellt werden `TridentOrchestrator`. Das Trident Installationsprogramm enthält Beispieldefinitionen für die Erstellung `TridentOrchestrator`. Dies startet eine Installation im `trident` Namespace.

```
$ 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:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    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
```

Der Trident-Operator ermöglicht es Ihnen, die Art und Weise, wie Astra Trident installiert wird, mithilfe der Attribute im anzupassen `TridentOrchestrator` Spez. Siehe "Anpassung der Trident Implementierung".

Der Status von `TridentOrchestrator` Gibt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an.

Status	Beschreibung
Installation	Der Betreiber installiert damit den Astra Trident `TridentOrchestrator` CR.
Installiert	Astra Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert den Astra Trident, denn `spec.uninstall=true`.
Deinstalliert	Astra Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Astra Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator versucht automatisch, aus diesem Zustand wiederherzustellen. Wenn dieser Status weiterhin besteht, müssen Sie eine Fehlerbehebung durchführen.
Aktualisierung	Der Bediener aktualisiert eine vorhandene Installation.
Fehler	Der `TridentOrchestrator` Wird nicht verwendet. Eine weitere ist bereits vorhanden.

Während der Installation den Status von `TridentOrchestrator` Änderungen von `Installing` Bis `Installed`. Wenn Sie die beobachten `Failed` Der Status und der Operator kann sich nicht selbst wiederherstellen. Sie sollten die Protokolle des Operators überprüfen. Siehe "Fehlerbehebung" Abschnitt.

Sie können überprüfen, ob die Astra Trident Installation abgeschlossen wurde, indem Sie sich die erstellten Pods ansehen:

```
$ 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
```

Sie können auch verwenden `tridentctl` Um die installierte Version von Astra Trident zu überprüfen.

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Jetzt können Sie mit diesem Schritt ein Backend erstellen. Siehe "Aufgaben nach der Implementierung".

Informationen zur Fehlerbehebung bei Problemen während der Bereitstellung finden Sie im "Fehlerbehebung" Abschnitt.

Anpassung der Trident Operator-Implementierung

Der Trident-Operator ermöglicht es Ihnen, die Art und Weise, wie Astra Trident installiert wird, mithilfe der Attribute im anzupassen `TridentOrchestrator` Spez.

Die Liste der Attribute finden Sie in der folgenden Tabelle:

Parameter	Beschreibung	Standard
`namespace`	Namespace für die Installation von Astra Trident in	„Standard“
`debug`	Aktivieren Sie das Debugging für Astra Trident	Falsch
`IPv6`	Installieren Sie Astra Trident über IPv6	Falsch
`k8sTimeout`	Zeitüberschreitung für Kubernetes-Betrieb	30 Sek.
`silenceAutosupport`	Schicken Sie AutoSupport Bundles nicht automatisch an NetApp	Falsch
`enableNodePrep`	Automatische Verwaltung der Abhängigkeiten von Workers Node (**BETA**)	Falsch
`autosupportImage`	Das Container-Image für AutoSupport Telemetrie	„netapp/Trident-Autosupport:21.04.0“
`autosupportProxy`	Die Adresse/der Port eines Proxys zum Senden von AutoSupport Telemetrie	"http://proxy.example.com:8888"
`uninstall`	Eine Flagge, die zum Deinstallieren von Astra Trident verwendet wird	Falsch
`logFormat`	Astra Trident Protokollformat zur Verwendung [Text, json]	„Text“
`tridentImage`	Astra Trident-Image zu installieren	„netapp/Trident:21.04“
`imageRegistry`	Pfad zur internen Registrierung des Formats `<registry FQDN>[:port][/subpath]`	„K8s.gcr.io/sig-Speicherung (k8s 1.18+) oder quay.io/k8scsi“
`kubeletDir`	Pfad zum kubelet-Verzeichnis auf dem Host	„/var/lib/kubelet“

Parameter	Beschreibung	Standard
`wipeout`	Eine Liste mit zu löschenden Ressourcen, um Astra Trident vollständig zu entfernen	
`imagePullSecrets`	Secrets, um Bilder aus einer internen Registrierung zu ziehen	
`controllerPluginNodeSelector`	Zusätzliche Node-Selektoren für Pods mit dem Trident Controller CSI Plugin Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`controllerPluginTolerations`	Überschreibungen von Verträgungen für Pods mit dem Trident Controller CSI-Plug-in Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional
`nodePluginNodeSelector`	Zusätzliche Node-Selektoren für Pods, auf denen das Trident Node CSI Plugin ausgeführt wird. Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`nodePluginTolerations`	Überschreibungen von Verträgungen für Pods mit dem Trident Node CSI Plugin Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional

`spec.namespace` Ist in angegeben `TridentOrchestrator` Um zu kennzeichnen, in welchem Namespace Astra Trident installiert ist. Dieser Parameter **kann nicht aktualisiert werden, nachdem Astra Trident installiert wurde**. Der Versuch, dies zu tun, verursacht den Status von `TridentOrchestrator` Zu ändern `Failed`. Astra Trident ist nicht für die Migration auf Namespaces vorgesehen.

Die automatische Workers Node Prep ist eine **Beta-Funktion**, die nur in nicht-Produktionsumgebungen verwendet werden soll.

Weitere Informationen zum Formatieren von Pod-Parametern finden Sie unter "Pods werden Nodes zugewiesen".

Sie können die oben genannten Attribute beim Definieren verwenden `TridentOrchestrator` Um die Installation anzupassen. Hier ein Beispiel:

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

Das folgende Beispiel zeigt, wie Trident mit Node-Selektoren implementiert werden kann:

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

Wenn Sie die Installation über das hinaus anpassen möchten `TridentOrchestrator` Argumente erlauben, sollten Sie erwägen, zu verwenden `tridentctl` So generieren Sie benutzerdefinierte YAML-Manifeste, die Sie nach Bedarf ändern können.