Gerencie e monitore Trident

Trident

NetApp
July 01, 2026

Índice

Gerencie e monitore Trident

Atualizar Trident

Atualizar Trident

A partir da versão 24.02, Trident segue um ciclo de lançamentos de quatro meses, disponibilizando três versões principais a cada ano civil. Cada nova versão se baseia nas versões anteriores e oferece novos recursos, melhorias de desempenho, correções de bugs e aprimoramentos. Recomendamos que você atualize pelo menos uma vez por ano para aproveitar os novos recursos em Trident.

Considerações antes de atualizar

Ao atualizar para a versão mais recente do Trident, considere o seguinte:

- Deve haver apenas uma instância do Trident instalada em todos os namespaces de um determinado cluster Kubernetes.
- Trident 23.07 e versões posteriores exigem snapshots de volume v1 e não oferecem mais suporte para snapshots alpha ou beta.
- Ao atualizar, é importante que você forneça `parameter.fsType` em `StorageClasses` usado pelo Trident. Você pode excluir e recriar `StorageClasses` sem interromper volumes pré-existentes.
  - Este é um **requisito** para impor "contextos de segurança" para volumes SAN.
  - O diretório sample input contém exemplos, como `storage-class-basic.yaml.templ` e `storage-class-bronze-default.yaml`.
  - Para obter mais informações, consulte "Problemas conhecidos".

Passo 1: selecione uma versão

As versões do Trident seguem uma convenção de nomenclatura baseada em data `YY.MM`, onde "YY" são os dois últimos dígitos do ano e "MM" é o mês. As versões com ponto seguem uma `YY.MM.X` convenção, onde "X" é o nível de patch. Você selecionará a versão para atualizar com base na versão da qual está atualizando.

- Você pode realizar uma atualização direta para qualquer versão de destino que esteja dentro de um intervalo de quatro versões da sua versão instalada. Por exemplo, você pode atualizar diretamente de 24.06 (ou qualquer versão secundária de 24.06) para 25.06.
- Se você estiver atualizando de uma versão fora do período de quatro versões, execute uma atualização em várias etapas. Use as instruções de atualização para a "versão anterior" da qual você está atualizando para atualizar para a versão mais recente que se encaixe no período de quatro versões. Por exemplo, se você estiver executando a versão 23.07 e quiser atualizar para a versão 25.06:
  a. Primeira atualização da 23.07 para a 24.06.
  b. Em seguida, atualize de 24.06 para 25.06.

> Ao atualizar usando o operador Trident na OpenShift Container Platform, você deve atualizar para o Trident 21.01.1 ou posterior. O operador Trident lançado com 21.01.0 contém um problema conhecido que foi corrigido em 21.01.1. Para mais detalhes, consulte o "detalhes do problema em GitHub".

Etapa 2: determine o método de instalação original

Para determinar qual versão você usou para instalar originalmente o Trident:

1. Utilize `kubectl get pods -n trident` para examinar os pods.
   - Se não houver um pod de operador, Trident foi instalado usando `tridentctl`.
   - Caso exista um pod de operador, Trident foi instalado usando o Trident operator, seja manualmente ou usando Helm.
2. Se houver um pod de operador, use `kubectl describe torc` para determinar se Trident foi instalado usando Helm.
   - Se houver uma etiqueta Helm, Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, Trident foi instalado manualmente usando o operador Trident.

Etapa 3: selecione um método de atualização

Geralmente, você deve atualizar usando o mesmo método que usou para a instalação inicial, porém você pode "alternar entre métodos de instalação". Existem duas opções para atualizar Trident.

- "Faça o upgrade usando o operador Trident"

> Sugerimos que você revise "Entenda o fluxo de trabalho de atualização do operador" antes de fazer o upgrade com o operador.

*

Atualize com o operador

Entenda o fluxo de trabalho de atualização do operador

Antes de usar o operador Trident para atualizar o Trident, você deve entender os processos em segundo plano que ocorrem durante a atualização. Isso inclui alterações no controlador Trident, no Pod do controlador e nos Pods dos nós, e no DaemonSet dos nós que permitem atualizações contínuas.

Gerenciamento de atualização do operador Trident

Uma das muitas "benefícios de usar o operador Trident" maneiras de instalar e atualizar Trident é o gerenciamento automático de objetos do Trident e do Kubernetes sem interromper os volumes montados existentes. Dessa forma, Trident pode suportar atualizações sem inatividade, ou "*atualizações contínuas*". Em particular, o operador do Trident se comunica com o cluster Kubernetes para:

- Exclua e recrie a implantação do Trident Controller e o DaemonSet do nó.
- Substitua o Trident Controller Pod e os Trident Node Pods por novas versões.
  - Se um nó não for atualizado, isso não impede que os nós restantes sejam atualizados.
  - Somente nós com um Trident Node Pod em execução podem montar volumes.

> Para obter mais informações sobre a arquitetura do Trident no cluster Kubernetes, consulte "Arquitetura do Trident".

Fluxo de trabalho de atualização do operador

Ao iniciar uma atualização usando o Trident operator:

1. O **operador Trident**:
   a. Detecta a versão atualmente instalada do Trident (versão *n*).
   b. Atualiza todos os objetos do Kubernetes, incluindo CRDs, RBAC e Trident SVC.
   c. Exclui a implantação do Trident Controller para a versão *n*.
   d. Cria a implantação do Trident Controller para a versão *n+1*.
2. **Kubernetes** cria o Pod do Controlador Trident para *n+1*.
3. O **operador Trident**:
   a. Exclui o Trident Node DaemonSet para *n*. O operador não espera pelo término do Pod do nó.
   b. Cria o Daemonset Trident Node para *n+1*.
4. **Kubernetes** cria Trident Node Pods em nós que não estão executando Trident Node Pod *n*. Isso garante que nunca haja mais de um Trident Node Pod, de qualquer versão, em um nó.

Atualize uma instalação do Trident usando o Trident operator ou o Helm

Você pode atualizar o Trident usando o operador Trident, seja manualmente ou usando o Helm. Você pode atualizar de uma instalação do operador Trident para outra instalação do operador Trident ou atualizar de uma `tridentctl` instalação para uma versão do operador Trident. Revise "Selecione um método de atualização" antes de atualizar uma instalação do operador Trident.

Atualize uma instalação manual

Você pode atualizar de uma instalação do operador Trident com escopo de cluster para outra instalação do operador Trident com escopo de cluster. Todas as versões do Trident usam um operador com escopo de cluster.

> Para atualizar a partir do Trident instalado usando o operador com escopo de namespace (versões 20.07 a 20.10), use as instruções de atualização para "sua versão instalada" do Trident.

Sobre esta tarefa

Trident fornece um arquivo de pacote que você pode usar para instalar o operador e criar objetos associados para sua versão do Kubernetes.

- Para clusters executando Kubernetes 1.25 ou posterior, use "bundle_post_1_25.yaml".

Antes de começar

Certifique-se de estar usando um cluster Kubernetes em execução "uma versão do Kubernetes compatível".

Passos

1. Verifique sua versão do Trident:

```
./tridentctl -n trident version
```

2. Atualize o `operator.yaml`, `tridentorchestrator_cr.yaml`e `post_1_25_bundle.yaml`com o registro e os caminhos de imagem para a versão para a qual você está atualizando (por exemplo, 25.06) e o segredo correto.
3. Exclua o operador Trident que foi usado para instalar a instância Trident atual. Por exemplo, se você estiver atualizando da 25.02, execute o seguinte comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Se você personalizou sua instalação inicial usando `TridentOrchestrator` atributos, pode editar o objeto `TridentOrchestrator` para modificar os parâmetros de instalação. Isso pode incluir alterações feitas para especificar registros de imagens espelhados do Trident e do CSI para o modo offline, ativar logs de depuração ou especificar segredos de extração de imagens.
5. Instale Trident usando o arquivo YAML de bundle correto para o seu ambiente, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes. Por exemplo, se você estiver instalando Trident 25.06.0, execute o seguinte comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Edite o trident torc para incluir a imagem 25.06.0.

Atualize uma instalação Helm

Você pode atualizar uma instalação do Trident.

> Ao atualizar um cluster Kubernetes da versão 1.24 para a 1.25 ou posterior que tenha Trident instalado, você deve atualizar o arquivo values.yaml para definir `excludePodSecurityPolicy` para `true` ou adicionar `--set excludePodSecurityPolicy=true` ao comando `helm upgrade` antes de poder atualizar o cluster.

Se você já atualizou seu cluster Kubernetes da versão 1.24 para a 1.25 sem atualizar o Trident helm, a atualização do helm falhará. Para que a atualização do helm seja concluída, execute estas etapas como pré-requisitos:

1. Instale o plugin helm-mapkubeapis de https://github.com/helm/helm-mapkubeapis.
2. Execute um teste de execução para o release do Trident no namespace onde Trident está instalado. Isso lista os recursos que serão limpos.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Execute uma execução completa com o helm para fazer a limpeza.

```
helm mapkubeapis trident  --namespace trident
```

Passos

1. Se você "instalou o Trident usando Helm", pode usar `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0` para atualizar em uma única etapa. Se você não adicionou o repositório Helm ou não consegue usá-lo para atualizar:
   a. Baixe a versão mais recente do Trident em "a seção *Ativos* em GitHub".
   b. Utilize o `helm upgrade` comando onde `trident-operator-26.02.0.tgz` reflete a versão para a qual você deseja atualizar.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

> Se você definir opções personalizadas durante a instalação inicial (como especificar registros privados e espelhados para as imagens do Trident e CSI), acrescente o `helm upgrade` comando usando `--set` para garantir que essas opções sejam incluídas no comando de upgrade, caso contrário, os valores serão redefinidos para o padrão.

2. Execute `helm list` para verificar se o gráfico e a versão do aplicativo foram atualizados. Execute `tridentctl logs` para revisar quaisquer mensagens de depuração.

Atualize de uma `tridentctl` instalação para Trident operator

Você pode atualizar para a versão mais recente do operador Trident a partir de uma `tridentctl` instalação. Os backends e PVCs existentes estarão automaticamente disponíveis.

> Antes de alternar entre os métodos de instalação, revise "Alternando entre métodos de instalação".

Passos

1. Baixe a versão mais recente do Trident.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. Crie o `tridentorchestrator`CRD a partir do manifesto.

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

3. Implante o operador com escopo de cluster no mesmo namespace.

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Crie um `TridentOrchestrator` CR para instalação do Trident.

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirme se Trident foi atualizado para a versão pretendida.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

Atualizar com tridentctl

Você pode facilmente atualizar uma instalação existente do Trident usando `tridentctl`.

Sobre esta tarefa

Desinstalar e reinstalar Trident funciona como uma atualização. Ao desinstalar Trident, o Persistent Volume Claim (PVC) e o Persistent Volume (PV) usados pela implantação do Trident não são excluídos. Os PVs que já foram provisionados permanecerão disponíveis enquanto Trident estiver offline, e Trident provisionará volumes para quaisquer PVCs que forem criados nesse meio tempo, após voltar a ficar online.

Antes de começar

Analise "Selecione um método de atualização" antes de atualizar usando `tridentctl`.

Passos

1. Execute o comando de desinstalação em `tridentctl` para remover todos os recursos associados ao Trident, exceto os CRDs e objetos relacionados.

```
./tridentctl uninstall -n <namespace>
```

2. Reinstale Trident. Consulte "Instale Trident usando tridentctl".

> Não interrompa o processo de atualização. Certifique-se de que o instalador seja executado até o fim.

Gerencie Trident usando tridentctl

O "Pacote de instalação Trident" inclui o `tridentctl` utilitário de linha de comando para fornecer acesso simples ao Trident. Usuários do Kubernetes com privilégios suficientes podem usá-lo para instalar Trident ou gerenciar o namespace que contém o pod do Trident.

Comandos e flags globais

Você pode executar `tridentctl help` para obter uma lista de comandos disponíveis para `tridentctl` ou adicionar a flag `--help` a qualquer comando para obter uma lista de opções e flags para esse comando específico.

```
tridentctl [command] [--optional-flag]
```

O utilitário Trident `tridentctl` suporta os seguintes comandos e flags globais.

Comandos

`create`
Adicionar um recurso ao Trident.

`delete`
Remover um ou mais recursos do Trident.

`get`
Obtenha um ou mais recursos do Trident.

`help`
Ajuda sobre qualquer comando.

`images`
Imprima uma tabela das imagens de contêiner que o Trident precisa.

`import`
Importe um recurso existente para Trident.

`install`
Instale Trident.

`logs`
Imprima os logs do Trident.

`send`
Envie um recurso do Trident.

`uninstall`
Desinstale o Trident.

`update`
Modificar um recurso no Trident.

`update backend state`
Suspender temporariamente as operações de backend.

`upgrade`
Atualize um recurso no Trident.

`version`
Imprima a versão do Trident.

Bandeiras globais

`-d, --debug`
Saída de depuração.

`-h, --help`
Ajuda para `tridentctl`.

`-k, --kubeconfig string`
Especifique o `KUBECONFIG` caminho para executar comandos localmente ou de um cluster Kubernetes para outro.

Alternativamente, você pode exportar a `KUBECONFIG` variável para apontar para um cluster Kubernetes específico e emitir `tridentctl` comandos para esse cluster.

`-n, --namespace string`
Namespace da implantação do Trident.

`-o, --output string`
Formato de saída. Um dos seguintes: json|yaml|name|wide|ps (padrão).

`-s, --server string`
Endereço/porta da interface API REST do Trident.

A interface REST do Trident pode ser configurada para escutar e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

Opções e parâmetros de comando

criar

Use o `create` comando para adicionar um recurso ao Trident.

```
tridentctl create [option]
```

Opções

`backend`: Adicione um backend ao Trident.

excluir

Use o comando `delete` para remover um ou mais recursos do Trident.

```
tridentctl delete [option]
```

Opções

`backend`: excluir um ou mais backends de armazenamento do Trident.
`snapshot`: excluir um ou mais snapshots de volume do Trident.
`storageclass`: excluir uma ou mais classes de armazenamento do Trident.

`volume`: excluir um ou mais volumes de armazenamento do Trident.

get

Use o comando `get` para obter um ou mais recursos do Trident.

```
tridentctl get [option]
```

Opções

`backend`: obtenha um ou mais backends de armazenamento do Trident.
`snapshot`: obtenha um ou mais snapshots do Trident.
`storageclass`: obtenha uma ou mais storage classes do Trident.
`volume`: obtenha um ou mais volumes do Trident.

Bandeiras

`-h`, `--help`: ajuda para volumes.
`--parentOfSubordinate string`: limitar a consulta ao volume de origem subordinado.
`--subordinateOf string`: limitar a consulta aos subordinados do volume.

imagens

Use `images` flags para imprimir uma tabela com as imagens dos contêineres que o Trident precisa.

```
tridentctl images [flags]
```

Bandeiras

`-h`, `--help`: ajuda para imagens.
`-v`, `--k8s-version string`: versão semântica do cluster Kubernetes.

importar volume

Use o `import volume` comando para importar um volume existente para Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Aliases

`volume`, `v`

Bandeiras

`-f`, `--filename string`: Caminho para o arquivo PVC em YAML ou JSON.
`-h`, `--help`: Ajuda para volume.
`--no-manage`: Criar somente PV/PVC. Não assumir gerenciamento de ciclo de vida do volume.

instalar

Use os `install` flags para instalar Trident.

```
tridentctl install [flags]
```

Bandeiras

`--autosupport-image string`: A imagem do contêiner para Autosupport Telemetry (padrão "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: O endereço/porta de um proxy para envio de Autosupport Telemetry.
`--enable-node-prep`: Tentar instalar os pacotes necessários nos nós.
`--generate-custom-yaml`: Gerar arquivos YAML sem instalar nada.
`-h, --help`: Ajuda para instalação.
`--http-request-timeout`: Substituir o tempo limite de solicitação HTTP para a API REST do Trident (padrão 1m30s).
`--image-registry string`: O endereço/porta de um registro de imagens interno.
`--k8s-timeout duration`: O tempo limite para todas as operações do Kubernetes (padrão 3m0s).
`--kubelet-dir string`: O local do host do estado interno do kubelet (padrão "/var/lib/kubelet").
`--log-format string`: O formato de log do Trident (text, json) (padrão "text").
`--node-prep`: Permite que o Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado. A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.**
`--pv string`: O nome do PV legado usado pelo Trident; certifique-se de que ele não exista (padrão "trident").
`--pvc string`: O nome do PVC legado usado pelo Trident; certifique-se de que ele não exista (padrão "trident").
`--silence-autosupport`: Não enviar pacotes autosupport para NetApp automaticamente (padrão true).
`--silent`: Desativar a maior parte da saída durante a instalação.
`--trident-image string`: A imagem do Trident a ser instalada.
`--k8s-api-qps`: O limite de consultas por segundo (QPS) para solicitações da API do Kubernetes (padrão 100; opcional).
`--use-custom-yaml`: Usar quaisquer arquivos YAML existentes no diretório de setup.
`--use-ipv6`: Usar IPv6 para a comunicação do Trident.

registros

Use `logs`flags para imprimir os logs do Trident.

```
tridentctl logs [flags]
```

Bandeiras

`-a, --archive`: Crie um arquivo de suporte com todos os logs, a menos que especificado de outra forma.
`-h, --help`: Ajuda para logs.
`-l, --log string`: Log do Trident a ser exibido. Um de trident|auto|trident-operator|all (padrão "auto").
`--node string`: O nome do nó do Kubernetes do qual coletar os logs do pod do nó.
`-p, --previous`: Obtenha os logs da instância anterior do contêiner, se existir.
`--sidecars`: Obtenha os logs dos contêineres sidecar.

enviar

Use o comando `send` para enviar um recurso do Trident.

```
tridentctl send [option]
```

Opções

`autosupport`: Envie um arquivo de Autosupport para NetApp.

desinstalar

Use `uninstall`parâmetros para desinstalar Trident.

```
tridentctl uninstall [flags]
```

Bandeiras

`-h, --help`: Ajuda para desinstalar.
`--silent`: Desativar a maior parte da saída durante a desinstalação.

atualizar

Use o `update` comando para modificar um recurso no Trident.

```
tridentctl update [option]
```

Opções

`backend`: Atualizar um backend no Trident.

atualizar estado do backend

Use o comando `update backend state` para suspender ou retomar as operações de backend.

```
tridentctl update backend state <backend-name> [flag]
```

Pontos a considerar

- Se um backend for criado usando um TridentBackendConfig (tbc), o backend não poderá ser atualizado usando um arquivo `backend.json`.
- Se o `userState` tiver sido definido em um tbc, ele não poderá ser modificado usando o comando `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Para recuperar a capacidade de definir o `userState` via tridentctl após ter sido definido via tbc, o campo `userState` deve ser removido do tbc. Isso pode ser feito usando o comando `kubectl edit tbc`. Após o campo `userState` ser removido, você pode usar o comando `tridentctl update backend state` para alterar o `userState` de um backend.
- Use o `tridentctl update backend state` para alterar o `userState`. Você também pode atualizar o `userState` usando `TridentBackendConfig` ou `backend.json` arquivo; isso aciona uma reinicialização completa do backend e pode ser demorado.

  #### Bandeiras

  `-h, --help`: Ajuda para o estado do backend.
  `--user-state`: Defina como `suspended` para pausar as operações do backend. Defina como `normal` para retomar as operações do backend. Quando definido como `suspended`:

- `AddVolume` e `Import Volume` estão em pausa.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` permanecem disponíveis.

Você também pode atualizar o estado do backend usando `userState` campo no arquivo de configuração do backend `TridentBackendConfig` ou `backend.json`. Para mais informações, consulte "Opções para

gerenciamento de backends" e "Realize o gerenciamento de backend com kubectl".

Exemplo:

JSON

Siga estes passos para atualizar o `userState` usando o arquivo `backend.json`:

1. Edite o `backend.json` arquivo para incluir o `userState` campo com o valor definido como 'suspended'.
2. Atualize o backend usando o `tridentctl update backend` comando e o caminho para o arquivo `backend.json` atualizado.

   Exemplo: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Você pode editar o tbc depois que ele for aplicado usando o `kubectl edit <tbc-name> -n <namespace>` comando. O exemplo a seguir atualiza o estado do backend para suspender usando a `userState: suspended` opção:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

versão

Use `version` flags para imprimir a versão de `tridentctl` e o serviço Trident em execução.

```
tridentctl version [flags]
```

Bandeiras

`--client`: somente versão cliente (não requer servidor).
`-h, --help`: ajuda para versão.

Suporte a plugin

O Tridentctl suporta plugins de forma semelhante ao kubectl. O Tridentctl detecta um plugin se o nome do arquivo binário do plugin seguir o esquema "tridentctl-<plugin>" e o binário estiver localizado em uma pasta listada na variável de ambiente PATH. Todos os plugins detectados são listados na seção de plugins da ajuda do tridentctl. Opcionalmente, você também pode limitar a busca especificando uma pasta de plugin na variável de ambiente TRIDENTCTL_PLUGIN_PATH (Exemplo: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Se a variável for usada, o tridentctl buscará apenas na pasta especificada.

Monitorar Trident

Trident fornece um conjunto de endpoints de métricas do Prometheus que você pode usar para monitorar o desempenho do Trident.

Visão geral

As métricas fornecidas pela Trident permitem que você faça o seguinte:

- Monitore a saúde e a configuração do Trident. Você pode examinar o sucesso das operações e se ele consegue se comunicar com os backends conforme o esperado.
- Analise as informações de utilização do backend e entenda quantos volumes estão provisionados em um backend e a quantidade de espaço consumido, e assim por diante.
- Mantenha um mapeamento da quantidade de volumes provisionados nos backends disponíveis.
- Monitore o desempenho. Você pode verificar quanto tempo Trident leva para se comunicar com os backends e executar operações.

> Por padrão, as métricas do Trident são expostas na porta de destino `8001` no endpoint `/metrics`. Essas métricas são **ativadas por padrão** quando Trident é instalado. Você pode configurar para consumir as métricas do Trident via HTTPS na porta `8444` também.

O que você vai precisar

- Um cluster do Kubernetes com Trident instalado.
- Uma instância do Prometheus. Isso pode ser um "implantação do Prometheus em contêiner" ou você pode optar por executar o Prometheus como um "aplicativo nativo".

Etapa 1: defina um alvo do Prometheus

Você deve definir um alvo Prometheus para coletar as métricas e obter informações sobre os backends que o Trident gerencia, os volumes que ele cria e assim por diante. Veja "Documentação do Prometheus Operator".

Passo 2: criar um Prometheus ServiceMonitor

Para consumir as métricas do Trident, você deve criar um Prometheus ServiceMonitor que monitore o `trident-csi` serviço e escute na `metrics` porta. Um exemplo de ServiceMonitor é o seguinte:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Esta definição de ServiceMonitor recupera as métricas retornadas pelo `trident-csi` serviço e procura especificamente pelo endpoint `metrics` do serviço. Como resultado, o Prometheus agora está configurado para entender as métricas do Trident.

Além das métricas disponíveis diretamente no Trident, kubelet expõe muitas `kubelet_volume_*` métricas por meio de seu próprio endpoint de métricas. Kubelet pode fornecer informações sobre os volumes que estão anexados, e pods e outras operações internas que gerencia. Consulte "aqui".

Consuma métricas do Trident por HTTPS

Para consumir métricas do Trident via HTTPS (porta 8444), você deve modificar a definição do ServiceMonitor para incluir a configuração TLS. Você também precisa copiar o `trident-csi` segredo do `trident` namespace para o namespace onde o Prometheus está sendo executado. Você pode fazer isso usando o seguinte comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Um exemplo de ServiceMonitor para métricas HTTPS é o seguinte:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident oferece suporte a métricas HTTPS em todos os métodos de instalação: tridentctl, Helm chart e Operator:

- Se você estiver usando o `tridentctl install` comando, pode passar a `--https-metrics` flag para habilitar métricas HTTPS.
- Se você estiver usando o gráfico Helm, você pode definir o parâmetro `httpsMetrics` para habilitar métricas HTTPS.
- Se você estiver usando arquivos YAML, pode adicionar a `--https_metrics` flag ao `trident-main` contêiner no `trident-deployment.yaml` arquivo.

Etapa 3: Consultar métricas do Trident com PromQL

PromQL é bom para criar expressões que retornam séries temporais ou dados tabulares.

Aqui estão algumas consultas PromQL que você pode usar:

Obtenha informações de integridade do Trident

- **Porcentagem de respostas HTTP 2XX do Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Percentual de respostas REST do Trident via código de status**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duração média em ms das operações realizadas pelo Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenha informações de uso do Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espaço total de volume provisionado por cada backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenha o uso de volume individual

Isso só é habilitado se as métricas do kubelet também forem coletadas.

- **Percentual de espaço utilizado para cada volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Saiba mais sobre a telemetria Trident AutoSupport

Por padrão, Trident envia métricas do Prometheus e informações básicas de backend para a NetApp em uma cadência diária.

- Para impedir que o Trident envie métricas do Prometheus e informações básicas de backend para NetApp, passe a `--silence-autosupport` flag durante a instalação do Trident.
- Trident também pode enviar logs de contêiner para o suporte da NetApp sob demanda via `tridentctl send autosupport`. Você precisará acionar o Trident para fazer upload dos logs. Antes de enviar os logs, você deve aceitar a NetApphttps://www.netapp.com/company/legal/privacy-policy/["política de privacidade"^].
- A menos que especificado, Trident busca os logs das últimas 24 horas.
- Você pode especificar o período de retenção de logs com a `--since` flag. Por exemplo: `tridentctl send autosupport --since=1h`. Essas informações são coletadas e enviadas por meio de um `trident-autosupport` contêiner que é instalado junto com Trident. Você pode obter a imagem do contêiner em "Trident AutoSupport".
- Trident AutoSupport não coleta nem transmite Informações de Identificação Pessoal (PII) ou Informações Pessoais. Vem com uma "Contrato de licença do usuário final" que não se aplica à própria imagem do contêiner Trident. Você pode saber mais sobre o compromisso da NetApp com a segurança e a confiança dos dados "aqui".

Um exemplo de carga enviada pelo Trident é semelhante a esta:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- As mensagens do AutoSupport são enviadas para o endpoint do AutoSupport da NetApp. Se você estiver usando um registro privado para armazenar imagens de contêiner, poderá usar a flag `--image-registry`.

- Você também pode configurar URLs de proxy gerando os arquivos YAML de instalação. Isso pode ser feito usando `tridentctl install --generate-custom-yaml` para criar os arquivos YAML e adicionando o `--proxy-url` argumento para o contêiner `trident-autosupport` em `trident-deployment.yaml`.

Desativar métricas do Trident

Para **desativar** a geração de relatórios de métricas, você deve gerar arquivos YAML personalizados (usando a `--generate-custom-yaml` flag) e editá-los para remover a `--metrics` flag da invocação do contêiner `trident-main`.

Desinstalar Trident

Você deve usar o mesmo método para desinstalar Trident que usou para instalar Trident.

Sobre esta tarefa

- Se você precisar corrigir erros observados após uma atualização, problemas de dependência ou uma atualização incompleta ou malsucedida, você deve desinstalar Trident e reinstalar a versão anterior usando as instruções específicas para essa "versão". Esta é a única maneira recomendada de *fazer o downgrade* para uma versão anterior.
- Para facilitar a atualização e reinstalação, desinstalar Trident não remove os CRDs ou objetos relacionados criados por Trident. Se precisar remover completamente Trident e todos os seus dados, consulte "Remova completamente Trident e os CRDs".

Antes de começar

Se você estiver desativando clusters Kubernetes, deverá excluir todos os aplicativos que utilizam volumes criados pelo Trident antes da desinstalação. Isso garante que os PVCs sejam despublicados nos nós do Kubernetes antes de serem excluídos.

Determine o método de instalação original

Você deve usar o mesmo método para desinstalar Trident que usou para instalá-lo. Antes de desinstalar, verifique qual versão você usou para instalar originalmente o Trident.

1. Utilize `kubectl get pods -n trident` para examinar os pods.
   - Se não houver um pod de operador, Trident foi instalado usando `tridentctl`.
   - Caso exista um pod de operador, Trident foi instalado usando o Trident operator, seja manualmente ou usando Helm.
2. Se houver um pod de operador, use `kubectl describe tproc trident` para determinar se Trident foi instalado usando Helm.
   - Se houver uma etiqueta Helm, Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, Trident foi instalado manualmente usando o operador Trident.

Desinstalar uma instalação do Trident

Você pode desinstalar uma instalação do operador Trident manualmente ou usando Helm.

Desinstalar instalação manual

Se você instalou Trident usando o operador, pode desinstalá-lo fazendo uma das seguintes ações:

1. **Edite `TridentOrchestrator` o CR e defina o sinalizador de desinstalação**:

   ```
   kubectl patch torc <trident-orchestrator-name> --type=merge -p
   '{"spec":{"uninstall":true}}'
   ```

   Quando a `uninstall` flag está definida como `true`, o operador Trident desinstala Trident, mas não remove o TridentOrchestrator em si. Você deve limpar o TridentOrchestrator e criar um novo se quiser instalar Trident novamente.

2. **Excluir `TridentOrchestrator`**: Ao remover o `TridentOrchestrator` CR que foi usado para implantar Trident, você instrui o operador a desinstalar Trident. O operador processa a remoção de `TridentOrchestrator` e procede à remoção da implantação e do daemonset do Trident, excluindo os pods do Trident que havia criado como parte da instalação.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Desinstalar instalação Helm

Se você instalou Trident usando Helm, pode desinstalá-lo usando `helm uninstall`.

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstalar uma `tridentctl` instalação

Use o `uninstall` comando em `tridentctl` para remover todos os recursos associados ao Trident, exceto os CRDs e objetos relacionados:

```
./tridentctl uninstall -n <namespace>
```

Informações sobre direitos autorais

Informações sobre marcas comerciais