设置 **Astra** 控制中心

Astra Control Center

NetApp
August 11, 2025

This PDF was generated from https://docs.netapp.com/zh-cn/astra-control-center/get-started/add-license.html on August 11, 2025. Always check docs.netapp.com for the latest.

目录

设置 Astra 控制中心

添加 Astra 控制中心的许可证

安装Astra Control Center时、已安装嵌入式评估版许可证。如果您正在评估Astra Control Center、则可以跳过此步骤。

您可以使用Astra Control UI或添加新许可证 "Astra Control API"。

Astra控制中心许可证使用Kubernetes CPU单元测量CPU资源、并计算分配给所有受管Kubernetes集群的工作节点的CPU资源。许可证基于vCPU使用量。有关如何计算许可证的详细信息、请参见 "许可"。

如果您的安装增长到超过许可的 CPU 单元数，则 Astra 控制中心将阻止您管理新应用程序。超过容量时，将显示警报。

要更新现有评估版或完整许可证、请参见 "更新现有许可证"。

开始之前

- 访问新安装的Astra Control Center实例。
- 管理员角色权限。
- 答 "NetApp 许可证文件" (nlf)。

步骤

1. 登录到 Astra 控制中心 UI 。
2. 选择 * 帐户 * > * 许可证 * 。
3. 选择 * 添加许可证 * 。
4. 浏览到您下载的许可证文件（ NLF ）。
5. 选择 * 添加许可证 * 。
   - 帐户 * > * 许可证 * 页面显示许可证信息，到期日期，许可证序列号，帐户 ID 和使用的 CPU 单元。

如果您拥有评估版许可证、并且不向AutoSupport 发送数据、请确保存储您的帐户ID、以避免在Astra控制中心发生故障时丢失数据。

启用Asta Control配置程序

Astra Trident 23.10及更高版本提供了使用Astra Control配置程序的选项、允许获得许可的Astra Control用户访问高级存储配置功能。除了基于标准Asta三端CSI的功能之外、Asta Control配置程序还提供了此扩展功能。

在即将推出的Astra Control更新中、Astra Control配置程序将取代Astra Trandent作为存储配置程序和流程编排程序、并且Astra Control必须使用它。因此、强烈建议Asta Control用户启用Asta Control配置程序。Asta三元数据将继续保持开源状态、并使用NetApp的新CSI和其他功能进行发布、维护、支持和更新。

关于此任务

如果您是Astra控制中心的许可用户、并且希望使用Astra控制配置程序功能、则应遵循此操作步骤。如果您是Astra三端数据库的用户、并且希望使用Astra Control配置程序提供的其他功能、而不同时使用Astra Control、则还应遵循此操作步骤。

对于每种情况、默认情况下在Astra Trident 24.02中不会启用配置程序功能、必须启用此功能。

开始之前

如果要启用Asta Control配置程序、请先执行以下操作：

Asta Control为用户提供Asta Control Center

- 获取**Astra**控制中心许可证：您需要 "Asta Control Center许可证" 启用Astra Control配置程序并访问它提供的功能。
- 安装或升级到**Astra Control Center 23.10**或更高版本：如果您计划在Astra Control中使用最新的Astra Control配置程序功能(24.02)，则需要最新的Astra Control Center版本(24.02)。
- *确认集群具有一个AMD64*系统架构：Astra Control配置程序映像在amd64和ARM64 CPU架构中都提供，但Astra Control Center仅支持amd64。
- 获取用于注册表访问的**Asta Control**服务帐户：如果要使用Asta Control注册表而不是NetApp 支持站点 来下载Asta Control配置程序映像、请完成的注册 "Asta Control Service帐户"。填写并提交表单并创建BlueXP帐户后、您将收到Astra Control Service欢迎电子邮件。
- 如果您安装了**Astra**三端安装程序，请确认其版本在四个版本的窗口内：如果您的Astra三端安装程序在版本24.02的四个版本窗口内，则可以使用Astra Control置备程序直接升级到Astra三端安装程序24.02。例如、您可以直接从Asta三端23.04升级到24.02。

仅适用于**Asta Control**配置程序用户

- 获取**Astra**控制中心许可证：您需要 "Asta Control Center许可证" 启用Astra Control配置程序并访问它提供的功能。
- 如果您安装了**Astra**三端安装程序，请确认其版本在四个版本的窗口内：如果您的Astra三端安装程序在版本24.02的四个版本窗口内，则可以使用Astra Control置备程序直接升级到Astra三端安装程序24.02。例如、您可以直接从Asta三端23.04升级到24.02。
- 获取**Astra Control Service**帐户以访问注册表：您需要访问注册表才能下载Astra Control配置程序映像。要开始使用、请完成的注册 "Asta Control Service帐户"。填写并提交表单并创建BlueXP帐户后、您将收到Astra Control Service欢迎电子邮件。

(步骤1)获取Asta Control配置程序映像

Asta控制中心用户可以使用Asta控制注册表或NetApp 支持站点 方法获取Asta控制配置程序映像。要在不使用Astra Control的情况下使用Astra Control配置程序的Astra Trent用户应使用注册表方法。

Astra Control图像注册表

在此操作步骤中、您可以使用Podman而不是Docker执行命令。如果您使用的是Windows环境、建议使用PowerShell。

1. 访问NetApp Astra控件映像注册表：
   a. 登录到Astra Control Service Web UI、然后选择页面右上角的图图标。
   b. 选择* API访问*。
   c. 记下您的帐户ID。
   d. 在同一页面中，选择*Generate API t令牌*并将API令牌字符串复制到剪贴板，然后将其保存在编辑器中。
   e. 使用您的首选方法登录Astra Control注册表：

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (仅限自定义注册表)按照以下步骤将图像移动到自定义注册表。如果您不使用注册表、请按照中的三端修复操作符步骤进行操作 "下一节"。
   a. 从注册表中提取Asta Control配置程序映像：

提取的映像不支持多个平台、只支持与提取映像的主机相同的平台、例如Linux amd64。

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform <cluster platform>
```

示例

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform linux/amd64
```

   a. 标记图像：

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0
```

b. 将映像推送到自定义注册表：

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

> 您可以使用"删除副本"来替代运行以下Docker命令：
> `crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0`

NetApp 支持站点

1. 下载Asta Control配置程序包 (`trident-acp-[version].tar`) "Astra Control Center下载页面"。
2. (建议但可选)下载Astra Control Center的证书和签名捆绑包(astra-control-crier-certs -[version].tar.gz)、以验证trident - acp-[version] tar捆绑包的签名。

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenterDockerImages-public.pub -signature certs/trident-acp-[version].tar.sig trident-acp-[version].tar
```

3. 加载Asta Control配置程序映像：

```
docker load < trident-acp-24.02.0.tar
```

响应：

```
Loaded image: trident-acp:24.02.0-linux-amd64
```

4. 标记图像：

```
docker tag trident-acp:24.02.0-linux-amd64 <my_custom_registry>/trident-acp:24.02.0
```

5. 将映像推送到自定义注册表：

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

(第2步)在Asta Trident中启用Asta Control配置程序

确定原始安装方法是否使用 "运算符(手动或使用Helm)或trdentcd" 并根据原始方法完成相应的步骤。

Asta三端操作员

1. "下载Asta三端安装程序并解压缩"。
2. 如果您尚未安装Astra三端安装程序、或者您从初始Astra三端安装程序中删除了操作员、请完成以下步骤：

   a. 创建客户需求日：

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

   b. 创建三项命名空间 (`kubectl create namespace trident`)或确认三项命名空间仍然存在 (`kubectl get all -n trident`）。如果已删除此命名空间、请重新创建它。

3. 将Astra Trdent更新到24.02.0：

   对于运行Kubornetes 1.24或更早版本的集群、请使用 `bundle_pre_1_25.yaml`。对于运行Kubernetes 1.25或更高版本的集群、请使用 `bundle_post_1_25.yaml`。

   ```
   kubectl -n trident apply -f trident-installer/deploy/<bundle-
   name.yaml>
   ```

4. 验证Astra trident是否正在运行：

   ```
   kubectl get torc -n trident
   ```

   响应：

   ```
   NAME      AGE
   trident   21m
   ```

5. 如果您有一个使用机密的注册表，请创建一个用于提取Astra Control置备程序映像的密钥：

   ```
   kubectl create secret docker-registry <secret_name> -n trident
   --docker-server=<my_custom_registry> --docker-username=<username>
   --docker-password=<token>
   ```

6. 编辑TridentOrchestrator CR并进行以下编辑：

   ```
   kubectl edit torc trident -n trident
   ```

a. 为Astra三端映像设置自定义注册表位置或从Astra Control注册表中提取该映像 (`tridentImage: <my_custom_registry>/trident:24.02.0` 或 `tridentImage: netapp/trident:24.02.0`) 。

b. 启用Asta Control配置程序 (`enableACP: true`) 。

c. 设置Asta Control配置程序映像的自定义注册表位置或将其从Asta Control注册表中提取 (`acpImage: <my_custom_registry>/trident-acp:24.02.0` 或 `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`) 。

d. 如果您已建立 图像拉取密钥 在本操作步骤的前面部分、您可以在此处设置它们 (`imagePullSecrets: - <secret_name>`) 。使用您在前面步骤中创建的相同名称机密名称。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

7. 保存并退出文件。部署过程将自动开始。

8. 验证是否已创建操作员、部署和副本集。

```
kubectl get all -n trident
```

> 在 Kubernetes 集群中只能有 * 一个操作符实例 * 。请勿创建多个部署的Asta三端操作员。

9. 验证 `trident-acp` 容器正在运行 `acpVersion` 为 `24.02.0` 状态为 `Installed`：

```
kubectl get torc -o yaml
```

响应：

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

Tridentctl

1. "下载Asta三端安装程序并解压缩"。
2. "如果您已有Asta Trident、请从托管它的集群中将其卸载"。
3. 在启用Asta Control配置程序的情况下安装Asta Trent (`--enable-acp=true`）：

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

4. 确认已启用Asta Control配置程序：

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

响应：

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. | +----------------
+----------------+-------------+
```

掌舵

1. 如果您安装了Astra Trident 23.07.1或更早版本、 "卸载" 操作员和其他组件。
2. 如果您的Kubornetes集群运行的是1.24或更早版本、请删除PSP：

```
kubectl delete psp tridentoperatorpod
```

3. 添加Astra Trident Helm存储库：

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

4. 更新Helm图表：

```
helm repo update netapp-trident
```

响应：

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

5. 列出图像：

```
./tridentctl images -n trident
```

响应：

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-driver-
registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

6. 确保提供了三项运算符24.02.0：

```
helm search repo netapp-trident/trident-operator --versions
```

响应：

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

7. 使用 ... `helm install` 并运行以下选项之一、其中包括这些设置：
   - 部署位置的名称
   - Astra三端版本
   - Asta Control配置程序映像的名称
   - 用于启用配置程序的标志
   - (可选)本地注册表路径。如果您使用的是本地注册表、则为 " {\f270 {\f151 {\f270} " 可以位于一个注册表或不同的注册表中、但所有CSI映像都必须位于同一注册表中。
   - 三端名称空间

选项

- 没有注册表的映像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-acp:24.02.0
--set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- 一个或多个注册表中的图像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

您可以使用 `helm list` 查看安装详细信息、例如名称、命名空间、图表、状态、应用程序版本、和修订版号。

如果您在使用Helm部署TRIdent时遇到任何问题、请运行此命令以完全卸载Asta TRIdent：

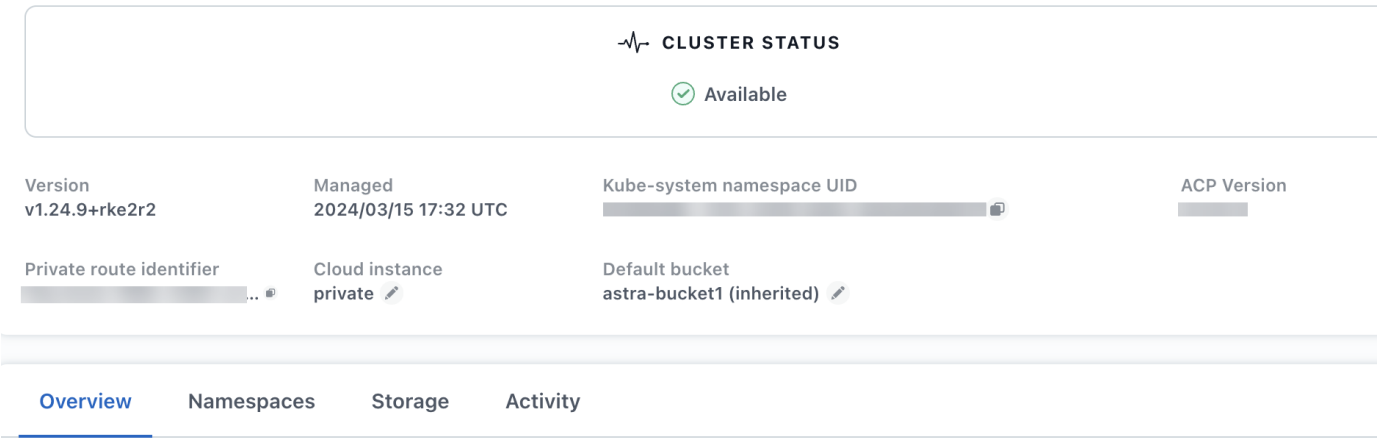
```
./tridentctl uninstall -n trident
```

请勿 "完全删除Asta Trdent CRD" 在尝试重新启用Astra Control配置程序之前、作为卸载的一部分。

结果

Asta Control配置程序功能已启用、您可以使用当前运行的版本可用的任何功能。

(仅适用于Asta Control Center用户)安装Asta Control配置程序后、在Asta Control Center UI中托管此配置程序的集群将显示 `ACP version` 而不是 `Trident version` 字段和当前安装的版本号。



有关详细信息 ...

- "Asta Trdent升级文档"

使用Astra Control准备用于集群管理的环境

在添加集群之前、应确保满足以下前提条件。此外、您还应运行资格检查、以确保您的集群已准备好添加到Astra Control Center、并根据需要创建kubeconfig"集群角色。

Astra Control允许您根据环境和首选项添加由自定义资源(Custom Resource、CR)或kubeconfig"管理的集群。

开始之前

- 满足环境前提条件：您的环境满足 "操作环境要求" A作用 控制中心。
- 配置工作节点：确保您 "配置工作节点" 在集群中使用适当的存储驱动程序、以便Pod可以与后端存储进行交互。
- 启用**PSA**限制：如果集群启用了POD安全准入强制(这是Kubernetes 1.25及更高版本集群的标准配置)、则需要对以下名称空间启用PSA限制：
  - `netapp-acc-operator` 命名空间：

```
kubectl label --overwrite ns netapp-acc-operator pod-
security.kubernetes.io/enforce=privileged
```

  - `netapp monitoring` 命名空间：

```
kubectl label --overwrite ns netapp-monitoring pod-
security.kubernetes.io/enforce=privileged
```

- * ONTAP 凭据*：您需要在备用ONTAP 系统上设置ONTAP 凭据以及超级用户和用户ID、以便使用Astra控制中心备份和还原应用程序。

  在ONTAP 命令行中运行以下命令：

```
export-policy rule modify -vserver <storage virtual machine name>
-policyname <policy name> -ruleindex 1 -superuser sys
export-policy rule modify -vserver <storage virtual machine name>
-policyname <policy name> -ruleindex 1 -anon 65534
```

- *kubeconfig-managed cluster requirement *：这些要求特定于由kubeconfig-managed的应用程序集群。
  - 使**kubeconfig***可访问：您可以访问 "默认集群kubeconfig" 那 "您在安装期间配置的"。
  - 证书颁发机构注意事项：如果使用引用私有证书颁发机构(CA)的kubeconfigfile文件添加集群、请将以下行添加到 `cluster` kubeconfig"文件的部分。这样、Astra Control便可添加集群：

```
insecure-skip-tls-verify: true
```

  - *仅Rancher *：在Rancher环境中管理应用程序集群时、请修改Rancher提供的kubeconfig文件中的应用程序集群默认上下文、以使用控制平面上下文、而不是Rancher API服务器上下文。这样可以减少Rancher API 服务器上的负载并提高性能。
- **Astra Control**配置程序要求：要管理集群、您应正确配置Astra Control配置程序(包括其Astra三项功能组件)。
  - 查看**Astra**三端环境要求：在安装或升级Astra Control配置程序之前、请查看 "支持的前端、后端和主机配置"。
  - 启用**Astra Control**配置程序功能：强烈建议您安装Astra Trident 23.10或更高版本并启用 "Asta Control配置程序高级存储功能"。在未来版本中、如果Asta Control配置程序未启用、则Asta Control将不支持Asta Trent。
  - 配置存储后端：必须至少有一个存储后端 "已在Asta Trdent中配置" 在集群上。
  - 配置存储类：必须至少有一个存储类 "已在Asta Trdent中配置" 在集群上。如果配置了默认存储类，请确保该存储类是具有默认标注的*Only"存储类。
  - 配置卷快照控制器并安装卷快照类： "安装卷快照控制器" 以便可以在Astra Control中创建快照。 "创建" 至少一个 `VolumeSnapshotClass` 使用Asta三端到功能。

运行资格检查

运行以下资格检查，以确保您的集群已准备好添加到 Astra 控制中心。

步骤

1. 确定您正在运行的Astra三项目标版本：

```
kubectl get tridentversion -n trident
```

如果存在Asta三项功能、您将看到类似于以下内容的输出：

```
NAME      VERSION
trident   24.02.0
```

如果Astra三端存储不存在、则会显示类似于以下内容的输出：

```
error: the server doesn't have a resource type "tridentversions"
```

2. 执行以下操作之一：
   - 如果您运行的是Asta三端凹凸版23.01或更早版本、请使用这些版本 "说明" 在升级到Asta Control配置程序之前、升级到Asta三端到最新版本。您可以 "执行直接升级" 如果您的Astra三端存储在版本24.02的四个版本的窗口中、则将Astra Control配置程序更新为24.02。例如、您可以直接从Asta三端23.04升级到Asta Control配置程序24.02。
   - 如果您运行的是Astra Trident 23.10或更高版本、请验证Astra Control配置程序是否已启用 "enabled" 。Asta Control配置程序不能用于23.10之前的Asta Control Center版本。 "升级Astra Control配置程序" 以便它与您要升级的Astra Control Center版本相同、以访问最新功能。
3. 确保所有Pod (包括 `trident-acp`)正在运行：

```
kubectl get pods -n trident
```

4. 确定存储类是否正在使用受支持的Asta三端驱动程序。配置程序名称应为 `csi.trident.netapp.io`。请参见以下示例：

```
kubectl get sc
```

响应示例：

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io   Delete          Immediate
true                    5d23h
```

创建集群角色kubeconfig

对于使用kubeconfig"管理的集群、您可以选择为Astra Control Center创建有限权限或扩展权限管理员角色。这不是Astra控制中心设置所需的操作步骤、因为您已在中配置了kubeconfig "安装过程"。

如果您适用场景的环境发生以下任一情况、则此操作步骤可帮助您创建一个单独的kubeconfig:

- 您希望限制Astra Control对其管理的集群的权限
- 您使用多个环境、并且不能使用在安装期间配置的默认Asta Control kubeconfig,否则在您的环境中使用单一环境的有限角色将不起作用

开始之前

在完成操作步骤 步骤之前、请确保您对要管理的集群具有以下信息：

- 已安装kubectl v1.23或更高版本
- kubectl访问要使用Astra控制中心添加和管理的集群

> 对于此操作步骤 、您不需要对运行Astra控制中心的集群进行kubectl访问。

- 要使用活动环境的集群管理员权限管理的集群的活动kubeconfig

步骤

1. 创建服务帐户：
   a. 创建名为`asacontrol service-account.yaml`的服务帐户文件。

```
<strong>astracontrol-service-account.yaml</strong>
```

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

   b. 应用服务帐户：

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

2. 创建以下具有足够权限的集群角色之一、以使集群由Astra Control管理：

集群角色受限

此角色包含由Asta Control管理集群所需的最低权限：

a. 创建 `ClusterRole` 文件、例如、 `astra-admin-account.yaml`。

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
    - horizontalpodautoscalers
    - ingresses
    - jobs
    - namespaces
    - networkpolicies
    - persistentvolumeclaims
    - poddisruptionbudgets
    - pods
    - podtemplates
    - replicasets
    - replicationcontrollers
    - replicationcontrollers/scale
    - rolebindings
    - roles
    - secrets
    - serviceaccounts
    - services
    - statefulsets
    - tridentmirrorrelationships
    - tridentsnapshotinfos
    - volumesnapshots
    - volumesnapshotcontents
    verbs:
    - delete

  # Watch resources
  # Necessary to monitor progress
  - apiGroups:
    - ""
    resources:
    - pods
    - replicationcontrollers
    - replicationcontrollers/scale
    verbs:
    - watch

  # Update resources
  - apiGroups:
    - ""
    - build.openshift.io
    - image.openshift.io
    resources:
    - builds/details
    - replicationcontrollers
    - replicationcontrollers/scale
    - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

b. (仅适用于OpenShift集群)在末尾附加以下内容 `astra-admin-account.yaml` 文件：

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

c. 应用集群角色：

```
kubectl apply -f astra-admin-account.yaml
```

已扩展集群角色

此角色包含要由Asta Control管理的集群的扩展权限。如果您使用多个环境，并且无法使用在安装期间配置的默认Asta Control kubeconfig,则可以使用此角色，否则在您的环境中，只使用一个环境的有限角色将不起作用：

以下内容 `ClusterRole` 步骤是一个常规Kubbernetes示例。有关特定于您的环境的说明、请参见Kubennetes分发版的文档。

a. 创建 `ClusterRole` 文件、例如、 `astra-admin-account.yaml`。

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. 应用集群角色：

```
kubectl apply -f astra-admin-account.yaml
```

3. 为集群角色创建与服务帐户的集群角色绑定：

a. 创建一个 `ClusterRoleBindingm` 文件，该文件名为 `astracontrol – clusterrolebind.YAML` 。

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: astra-admin-account
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

b. 应用集群角色绑定：

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

4. 创建并应用令牌密钥：

   a. 创建名为的令牌机密文件 `secret-astracontrol-service-account.yaml`。

```
<strong>secret-astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. 应用令牌密钥：

```
kubectl apply -f secret-astracontrol-service-account.yaml
```

5. 通过将令牌密钥名称添加到、将其添加到服务帐户 `secrets` 数组(以下示例中的最后一行)：

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. 列出服务帐户密码，将 ` <context>` 替换为适用于您的安装的正确上下文：

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

输出的结尾应类似于以下内容：

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

中每个元素的索引 `secrets` 阵列以0开头。在上面的示例中、是的索引 `astracontrol-service-account-dockercfg-48xhx` 将为0、并为创建索引 `secret-astracontrol-service-account` 将为1。在输出中、记下服务帐户密钥的索引编号。在下一步中、您将需要此索引编号。

7. 按如下所示生成 kubeconfig：
   a. 创建 `create-kubeconfig.sh` 文件
   b. 替换 `TOKEN_INDEX` 在以下脚本的开头、使用正确的值。

```
<strong>create-kubeconfig.sh</strong>
```

```
# 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=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
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
```

c. 获取用于将其应用于 Kubernetes 集群的命令。

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

8. (可选)将kubeconfig重命名为集群的有意义名称。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(技术预览)为受管集群安装Asta Connector

由Asta Control Center管理的集群使用Asta Connector在受管集群和Asta Control Center之间实现通信。您需要在要管理的所有集群上安装Astra Connector。

安装A作用 连接器

您可以使用Kubbernetes命令和自定义资源(Custom Resource、CR)文件安装Astra Connector。

关于此任务

- 执行这些步骤时、请在要使用Astra Control进行管理的集群上执行这些命令。
- 如果使用的是Bastion主机、请从Bastion主机的命令行对这些命令执行问题描述。

开始之前

- 您需要访问要使用Astra Control管理的集群。
- 要在集群上安装Asta Connector操作员、您需要具有Kubbernetes管理员权限。

如果为集群配置了强制实施Pod安全接入(这是Kubernetes 1.25及更高版本集群的默认设置)、则需要对相应的卷空间启用PSA限制。请参见 "使用Astra Control准备用于集群管理的环境" 有关说明，请参见。

步骤

1. 在要使用Asta Control进行管理的集群上安装Asta Connector运算符。运行此命令时、命名空间 `astra-connector-operator` 创建并将配置应用于命名空间：

```
kubectl apply -f https://github.com/NetApp/astra-connector-operator/releases/download/24.02.0-202403151353/astraconnector_operator.yaml
```

2. 确认操作员已安装并准备就绪：

```
kubectl get all -n astra-connector-operator
```

3. 从Asta Control获取API令牌。请参见 "Astra Automation文档" 有关说明，请参见。
4. 使用令牌创建密钥。将<API_TOKEN>替换为您从Astra Control收到的令牌：

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. 创建Docker密钥以提取Astra Connector映像。将括号<>中的值替换为您环境中的信息：

   您可以在Astra Control Web UI中找到<ASTRA_CONTROL_ACCOUNT_ID>。在Web UI中，选择页面右上角的图图标，然后选择*API access*。

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. 创建Astra Connector CR文件并将其命名为 `astra-connector-cr.yaml`。更新方括号<>中的值以匹配您的Astra Control环境和集群配置：
   - <ASTRA_CONTROL_ACCOUNT_ID>：在上一步中从Astra Control Web UI获取。
   - <CLUSTER_NAME>：应在Asta Control中分配此集群的名称。
   - <ASTRA_CONTROL_URL>：Asta Control的Web UI URL。例如：

```
https://astra.control.url
```

```
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. 在您填充之后 `astra-connector-cr.yaml` 使用正确值的文件、应用CR：

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. 验证Asta Connector是否已完全部署：

```
kubectl get all -n astra-connector
```

9. 验证集群是否已注册到Astra Control：

```
kubectl get astraconnectors.astra.netapp.io -A
```

您应看到类似于以下内容的输出：

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e   Registered with Astra
```

10. 验证该集群是否显示在Astra Control Web UI的*集群*页面上的受管集群列表中。

添加集群

要开始管理应用程序，请添加 Kubernetes 集群并将其作为计算资源进行管理。您必须为 Astra 控制中心添加一个集群，才能发现您的 Kubernetes 应用程序。

我们建议，在将其他集群添加到 Astra 控制中心进行管理之前，先由 Astra 控制中心管理其部署所在的集群。要发送 Kubemmetrics 数据和集群关联数据以获取指标和故障排除信息，必须对初始集群进行管理。

开始之前

- 在添加集群之前，请查看并执行必要的操作 "前提条件任务"。
- 如果您使用的是ONTAP SAN驱动程序、请确保在所有Kubbernetes集群上启用了多路径。

步骤

1. 从信息板或集群菜单导航：
   - 从"Resource Summary"的"信息板"中、从"Clusters"窗格中选择"添加"。
   - 在左侧导航区域中、选择*集群*、然后从集群页面中选择*添加集群*。
2. 在打开的 * 添加集群 * 窗口中，上传 `kubeconfig.yaml` 文件或粘贴 `kubeconfig.yaml` 文件的内容。

   `kubeconfig.yaml` 文件应仅包含一个集群的集群凭据 * 。

   创建自己的 `kubeconfig` file中、您只能定义*一*上下文元素。请参见 "Kubernetes 文档" 有关创建的信息 `kubeconfig` 文件。如果您使用为有限集群角色创建了kubeconfig "此过程"、请务必在此步骤中上传或粘贴kubeconfig。

3. 请提供凭据名称。默认情况下，凭据名称会自动填充为集群的名称。
4. 选择 * 下一步 * 。
5. 选择要用于此Kubernetes集群的默认存储类、然后选择*下一步*。

   您应选择一个存储类、该存储类在Astra控件配置程序中进行配置、并由ONTAP存储提供支持。

6. 查看相关信息、如果一切正常、请选择*添加*。

结果

集群将进入*正在发现*状态、然后更改为*运行状况良好*。现在、您正在使用Astra控制中心管理集群。

添加要在 Astra 控制中心中管理的集群后，部署监控操作员可能需要几分钟的时间。在此之前，通知图标将变为红色并记录一个 * 监控代理状态检查失败 * 事件。您可以忽略此问题，因为当 Astra 控制中心获得正确状态时，问题描述将解析。如果问题描述 在几分钟内未解析、请转至集群并运行 `oc get pods -n netapp-monitoring` 作为起点。您需要查看监控操作员日志以调试问题。

在ONTAP存储后端启用身份验证

Astra控制中心提供了两种对ONTAP 后端进行身份验证的模式：

- 基于凭据的身份验证：具有所需权限的ONTAP 用户的用户名和密码。您应使用预定义的安全登录角色(如admin或vsadmin)、以确保与ONTAP 版本的最大兼容性。
- 基于证书的身份验证：Astra控制中心还可以使用后端安装的证书与ONTAP 集群进行通信。您应使用客户端证书、密钥和可信CA证书(如果使用)(建议)。

您可以稍后更新现有后端、以便从一种身份验证类型迁移到另一种身份验证方法。一次仅支持一种身份验证方法。

启用基于凭据的身份验证

ASRA控制中心需要集群范围的凭据 `admin` 与ONTAP 后端通信。您应使用标准的预定义角色、例如 `admin`。这样可以确保与未来的ONTAP 版本向前兼容、这些版本可能会公开功能API、以供未来的Astra控制中心版本使用。

可以创建自定义安全登录角色并将其用于Astra Control Center、但不建议这样做。

示例后端定义如下所示：

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

后端定义是以纯文本格式存储凭据的唯一位置。创建或更新后端是唯一需要了解凭据的步骤。因此、这是一项仅由管理员执行的操作、由Kubornetes或存储管理员执行。

启用基于证书的身份验证

Astra控制中心可以使用证书与新的和现有的ONTAP 后端进行通信。您应在后端定义中输入以下信息。

- `clientCertificate`：客户端证书。

- `clientPrivateKey`:关联的私钥。
- `trustedCACertificate`：可信CA证书。如果使用可信 CA ，则必须提供此参数。如果不使用可信 CA ，则可以忽略此设置。

您可以使用以下类型的证书之一：

- 自签名证书
- 第三方证书

使用自签名证书启用身份验证

典型的工作流包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时、请将公用名(Common Name、CN)设置为ONTAP 用户、以进行身份验证。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. 安装类型为的客户端证书 `client-ca` 和键ONTAP。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. 确认ONTAP 安全登录角色支持证书身份验证方法。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. 使用生成的证书测试身份验证。将<SVM ManagementLIF> and <vserver name> 替换为管理LIF IP和ONTAP 名称。您必须确保LIF的服务策略设置为 `default-data-management`。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. 使用上一步中获得的值、在Astra Control Center UI中添加存储后端。

如果您拥有第三方证书、则可以使用以下步骤设置基于证书的身份验证。

步骤

1. 生成私钥和CSR：

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. 将CSR传递到Windows CA (第三方CA)、然后问题描述 签名证书。
3. 下载签名证书并将其命名为`ONTAP signed_cert.crt
4. 从Windows CA (第三方CA)导出根证书。
5. 为此文件命名 `ca_root.crt`

   现在、您已有以下三个文件：

   - 私钥： `ontap_signed_request.key` (这是ONTAP 中服务器证书对应的密钥。安装服务器证书时需要此证书。)
   - 签名证书： `ontap_signed_cert.crt` (在ONTAP 中也称为_server certificATE _。)
   - 根**CA**证书： `ca_root.crt` (在ONTAP 中也称为_server-ca certific存在_。)
6. 在ONTAP 中安装这些证书。生成并安装 `server` 和 `server-ca` ONTAP 上的证书。

展开**SAMPLE.YAML**

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. 为同一主机创建客户端证书、以实现无密码通信。Asta控制中心使用此过程与ONTAP 进行通信。
8. 在ONTAP 上生成并安装客户端证书：

展开SAMPLE.YAML

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate's CA and serial number for reference:

CA:
serial:
The certificate's generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```


9. 在Asta Control Center UI中添加存储后端、并提供以下值：
   - 客户端证书：ONATP_TEST_client.prom
   - 私钥：ontap_test_client.key
   - 可信**CA**证书：ONATP_signed_cert.crt

添加存储后端

设置凭据或证书身份验证信息后、您可以将现有ONTAP 存储后端添加到Astra控制中心以管理其资源。

通过将 Astra Control 中的存储集群作为存储后端进行管理，您可以在永久性卷（ PV ）和存储后端之间建立链接，并获得其他存储指标。

如果启用了Astra Control配置程序、则在使用NetApp SnapMirror技术时、可以选择在Astra控制中心中添加和管理ONTAP存储后端。

步骤

1. 从左侧导航区域的信息板中、选择*后端*。
2. 选择 * 添加 * 。
3. 在添加存储后端页面的使用现有部分中，选择* ONTAP *。
4. 选择以下选项之一：
   - 使用管理员凭据：输入ONTAP 集群管理IP地址和管理员凭据。凭据必须是集群范围的凭据。

     您在此处输入凭据的用户必须具有 `ontapi` 在ONTAP 集群上的ONTAP 系统管理器中启用用户登录访问方法。如果您计划使用SnapMirror复制、请应用具有"admin"角色的用户凭据、该角色具有访问方法 `ontapi` 和 `http`、在源和目标ONTAP 集群上。请参见 "管理ONTAP 文档中的用户帐户" 有关详细信息 ...

   - 使用证书：上传证书 `.pem` file、证书密钥 `.key` 文件、以及证书颁发机构文件(可选)。
5. 选择 * 下一步 * 。
6. 确认后端详细信息并选择 * 管理 * 。

结果

后端将显示在中 `online` 包含摘要信息的列表中的状态。

您可能需要刷新页面才能显示后端。

添加存储分段

您可以使用Astra Control UI或添加存储分段 "Astra Control API"。如果要备份应用程序和永久性存储，或者要跨集群克隆应用程序，则必须添加对象存储分段提供程序。Astra Control 会将这些备份或克隆存储在您定义的对象存储分段中。

如果您要将应用程序配置和永久性存储克隆到同一集群、则无需在Astra Control中使用存储分段。应用程序快照功能不需要存储分段。

开始之前

- 确保您有一个可从Astra Control Center管理的集群访问的存储分段。
- 确保您具有此存储分段的凭据。
- 确保存储分段为以下类型之一：
  - NetApp ONTAP S3
  - NetApp StorageGRID S3
  - Microsoft Azure
  - 通用 S3

Amazon Web Services (AWS)和Google Cloud Platform (GCP)使用通用S3存储分段类型。

虽然Astra控制中心支持将Amazon S3作为通用S3存储分段提供商、但Astra控制中心可能不支持声称支持Amazon S3的所有对象存储供应商。

步骤

1. 在左侧导航区域中，选择 * 桶 * 。
2. 选择 * 添加 * 。
3. 选择存储分段类型。

   添加存储分段时，请选择正确的存储分段提供程序，并为该提供程序提供正确的凭据。例如， UI 接受 NetApp ONTAP S3 作为类型并接受 StorageGRID 凭据；但是，这将发生原因使使用此存储分段执行所有未来应用程序备份和还原失败。

4. 输入现有存储分段名称和可选的问题描述。

   存储分段名称和问题描述 显示为备份位置、您可以稍后在创建备份时选择该位置。此名称也会在配置保护策略期间显示。

5. 输入 S3 端点的名称或 IP 地址。
6. 在*选择凭据*下、选择*添加*或*使用现有*选项卡。
   - 如果选择*添加*：
     i. 在 Astra Control 中输入凭据名称，以便与其他凭据区分开。
     ii. 通过粘贴剪贴板中的内容来输入访问 ID 和机密密钥。
   - 如果选择*使用现有*：
     i. 选择要用于存储分段的现有凭据。
7. 选择 … `Add`。

   添加存储分段时、Astra Control会使用默认存储分段指示符标记一个存储分段。您创建的第一个存储分段将成为默认存储分段。添加分段时、您可以稍后决定添加 "设置另一个默认存储分段"。

版权信息

版权所有 © 2025 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。