ネットアップとストレージの統合の概要

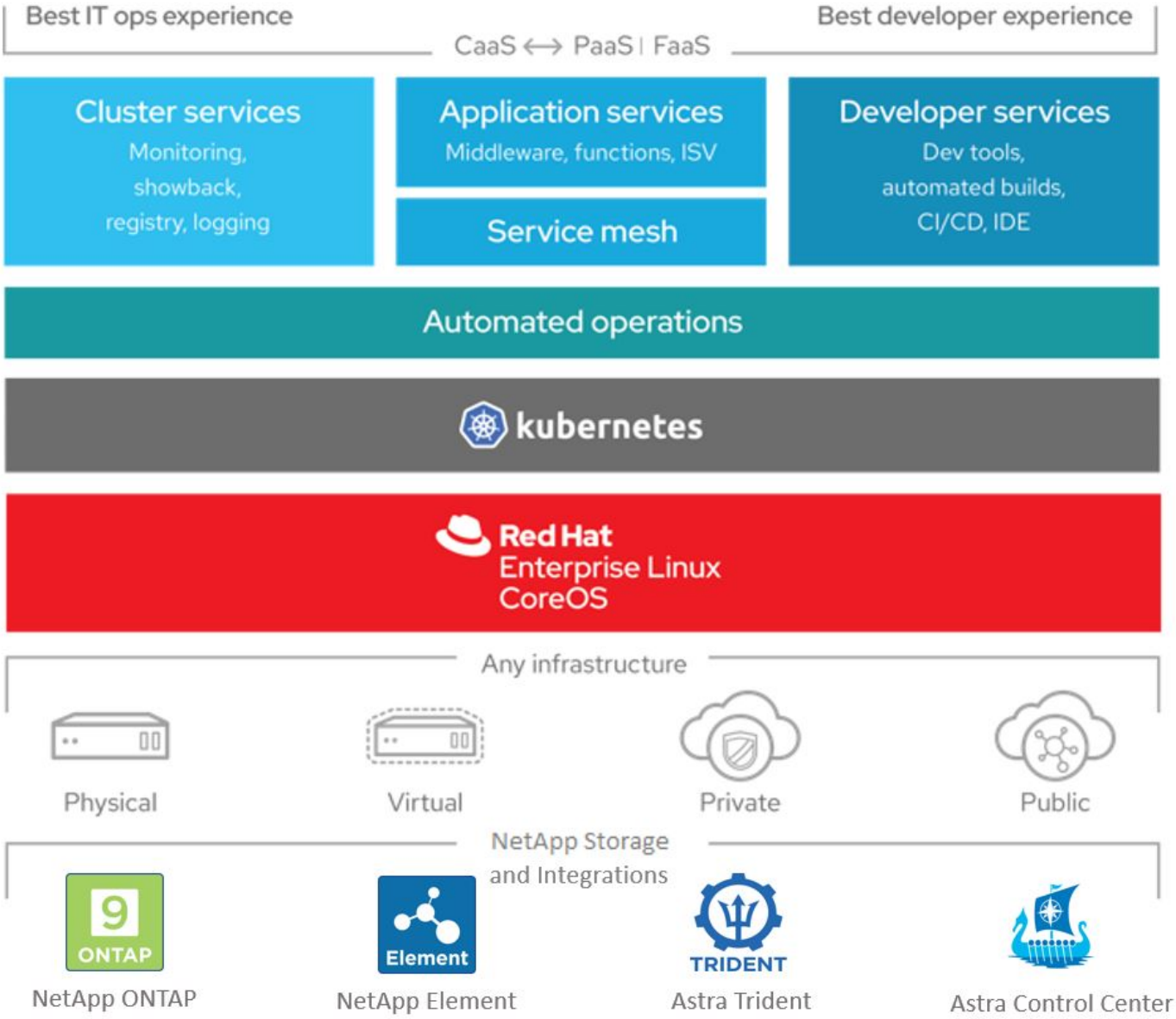
NetApp Solutions

NetApp
September 10, 2024

目次

ネットアップストレージ統合の概要

ネットアップは、 Red Hat OpenShift などのコンテナベースの環境における永続的データのオーケストレーションと管理に役立つさまざまな製品を提供します。



NetApp Astra Control は、ネットアップのデータ保護テクノロジを基盤とするステートフル Kubernetes ワークロード向けの充実したストレージサービスとアプリケーション対応データ管理サービスを提供します。Astra Control Service は、クラウドネイティブの Kubernetes 環境でステートフルワークロードをサポートするために利用できます。Astra Control Center は、 Red Hat OpenShift などのオンプレミス環境でステートフルワークロードをサポートするために使用できます。詳細については、 NetApp Astra Control の Web サイトをご覧ください "こちらをご覧ください"。

NetApp Astra Trident は、コンテナや Kubernetes ディストリビューション向けの、 Red Hat OpenShift などのオープンソースで完全にサポートされているストレージオーケストレーションツールです。詳細については、 Astra Trident の Web サイトをご覧ください "こちらをご覧ください"。

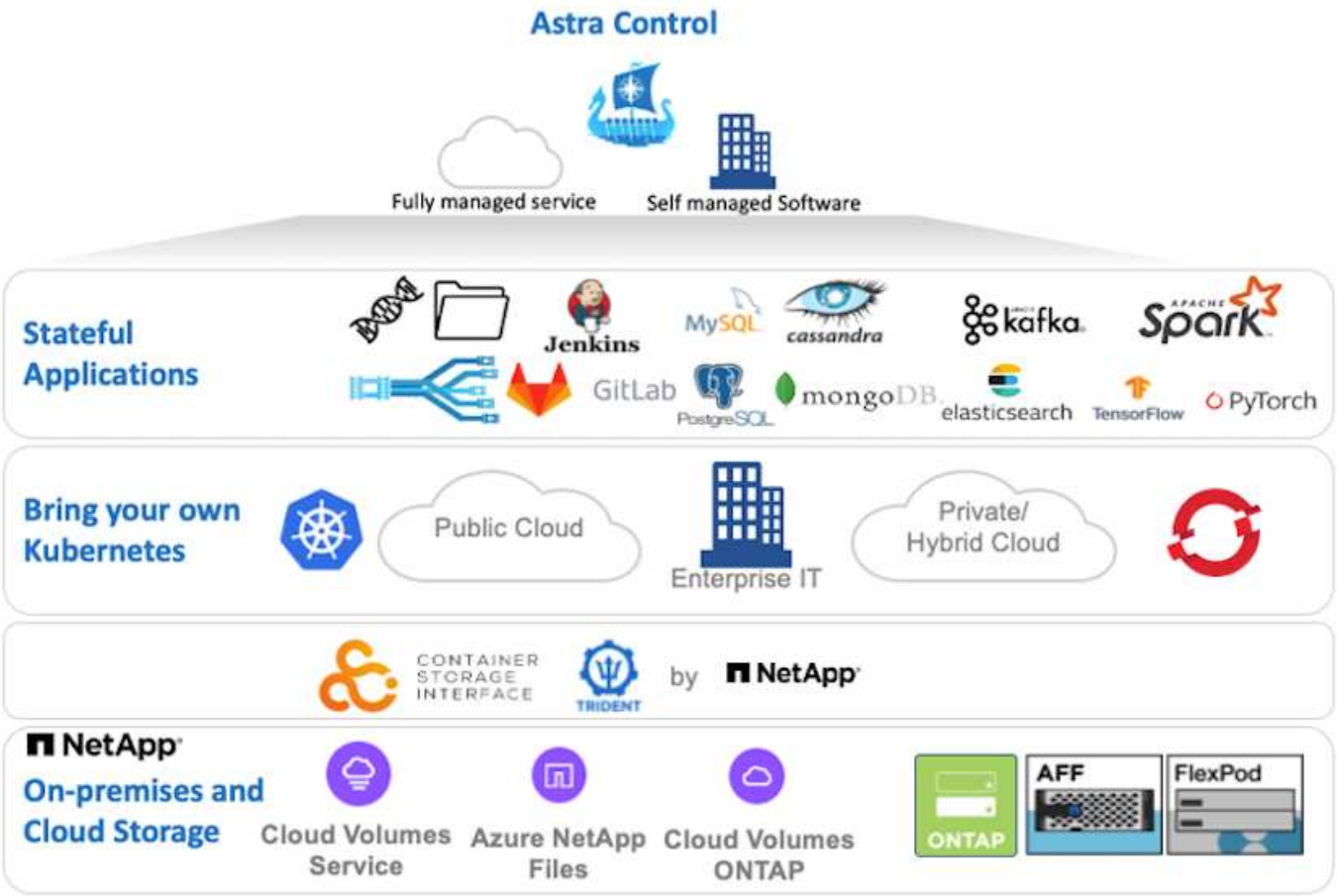
以下のページには、解決策追加情報に実装された Red Hat OpenShift でアプリケーションおよび永続的ストレ

ージ管理のために検証されたネットアップ製品に関する があります。

- "ネットアップアストラコントロールセンター"
- "ネットアップアストラト Trident"

NetApp Astra Control Center の概要

NetApp Astra Control Center は、オンプレミス環境に導入され、ネットアップのデータ保護テクノロジを基盤とするステートフル Kubernetes ワークロード向けの充実したストレージサービスとアプリケーション対応データ管理サービスを提供します。



NetApp Astra Control Center は、 Astra Trident ストレージオーケストレーションツールを導入し、 NetApp ONTAP ストレージシステムにストレージクラスとストレージバックエンドを使用して構成した Red Hat OpenShift クラスタにインストールできます。

Astra Trident のインストールと設定を行い、 Astra Control Center をサポートするには、を参照してください "このドキュメントはこちら"。

クラウド接続環境では、 Cloud Insights を使用して高度なモニタリングとテレメトリを提供します。Cloud Insights 接続がない場合は、限定的な監視と計測（ 7 日間相当の指標）を使用でき、オープン指標エンドポイントを介して Kubernetes の標準の監視ツール（ Prometheus および Grafana ）にエクスポートされます。

Astra Control Centerは、NetApp AutoSupportとActive IQのデジタルアドバイザ（デジタルアドバイザとも呼ばれます）エコシステムに完全に統合されており、ユーザのサポート、トラブルシューティングの支援、使用

状況の統計の表示を提供します。

Astra Control Center の有料版に加え、 90 日間の評価ライセンスも提供されています。評価版は、 E メールとコミュニティ（ Slack チャンネル）を通じてサポートされています。お客様は、これらの記事やその他のナレッジベース記事、および製品サポートダッシュボードから入手可能なドキュメントにアクセスできます。

ネットアップアストラコントロールセンターの利用を開始するには、にアクセスしてください "Astra の Web サイト"。

Astra Control Center のインストールの前提条件

1. 1 つ以上の Red Hat OpenShift クラスタ。バージョン 4.6 EUS および 4.7 が現在サポートされています。
2. 各 Red Hat OpenShift クラスタに Astra Trident をインストールして設定しておく必要があります。
3. ONTAP 9.5 以降を実行している NetApp ONTAP ストレージシステムが 1 つ以上必要です。

   サイトに各 OpenShift インストールを実装し、永続的ストレージ専用の SVM を用意することがベストプラクティスです。マルチサイト環境では、追加のストレージシステムが必要です。

4. Trident ストレージバックエンドは、 ONTAP クラスタがサポートする SVM を含む各 OpenShift クラスタで設定する必要があります。
5. ストレージプロビジョニングツールとして Astra Trident を使用し、各 OpenShift クラスタに設定されたデフォルトのストレージクラス。
6. ロードバランシングや OpenShift Services の公開のために、各 OpenShift クラスタにロードバランサをインストールして構成する必要があります。

   リンクを参照してください "こちらをご覧ください" この目的で検証済みのロードバランサに関する情報。

7. NetApp アストラ Control Center イメージをホストするには、プライベートイメージのレジストリを設定する必要があります。

   リンクを参照してください "こちらをご覧ください" この目的のために OpenShift プライベートレジストリをインストールして構成します。

8. Red Hat OpenShift クラスタにクラスタ管理者アクセス権限が必要です。
9. NetApp ONTAP クラスタへの管理者アクセスが必要です。
10. Docker または podman 、 tridentctl 、 OC または kubectl ツールがインストールされ、 $path に追加された管理ワークステーション。

    Docker をインストールする場合は、 20.10 よりも前のバージョンの Docker 、 Podman をインストールする場合は、バージョン 3.0 よりも前の podman が必要です。

Astra Control Center をインストールします

OperatorHub を使用する

1. ネットアップサポートサイトにログインし、最新バージョンの NetApp Astra Control Center をダウンロードします。そのためには、ネットアップアカウントにライセンスを関連付ける必要があります。tarball をダウンロードしたら、 admin ワークステーションに転送します。

   > Astra Control の試用版ライセンスの使用を開始するには、にアクセスしてください "Astra 登録サイト"。

2. tar ボールを開梱し、作業ディレクトリを作成されたフォルダに変更します。

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-
21.12.60.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.12.60
```

3. インストールを開始する前に、 Astra Control Center イメージをイメージレジストリにプッシュします。この手順では、 Docker または Podman のいずれかを使用して実行します。両方の手順については、この手順で説明します。

ポドマン

a. レジストリ FQDN を、組織 / 名前空間 / プロジェクト名とともに環境変数「管理」としてエクスポートします。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. レジストリにログインします。

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password
--tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインしている場合は、「 podman login -u OCP -user -p token --tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com` 」の代わりにトークンを使用します。

> または、サービスアカウントのトークンを使用して、サービスアカウントを作成し、（プッシュアクセスまたはプルアクセスが必要かどうかに応じて）レジストリエディタまたはレジストリビューアロールを割り当て、レジストリにログインすることもできます。

c. シェルスクリプトファイルを作成し、次の内容を貼り付けます。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

> レジストリに信頼されていない証明書を使用している場合は、シェルスクリプトを編集し、 podman push コマンドに「 --tls-verify=false 」を使用します。「 podman push $registry/ $ 」（ echo $astraaIlmage | sed’s /[\\/]\\/'/')--tls-verify=false 」）。

d. ファイルを実行可能にします

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. シェルスクリプトを実行します。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

Docker です

a. レジストリ FQDN を、組織 / 名前空間 / プロジェクト名とともに環境変数「管理」としてエクスポートします。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. レジストリにログインします。

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password
astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、「 password - `d Occker login -u OCP-user-p token astra-registry.apps.ocp-vmw.cie.netapp.com` 」の代わりにトークンを使用します。

> または、サービスアカウントのトークンを使用して、サービスアカウントを作成し、（プッシュアクセスまたはプルアクセスが必要かどうかに応じて）レジストリエディタまたはレジストリビューアロールを割り当て、レジストリにログインすることもできます。

c. シェルスクリプトファイルを作成し、次の内容を貼り付けます。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed
's/Loaded image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

d. ファイルを実行可能にします

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. シェルスクリプトを実行します。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. 公開されていないプライベートイメージレジストリを使用する場合は、イメージレジストリ TLS 証明書を OpenShift ノードにアップロードします。そのためには、 TLS 証明書を使用して OpenShift -config ネームスペースに ConfigMap を作成し、クラスタイメージ構成にパッチを適用して証明書を信頼できるようにします。

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-ca -n
openshift-config --from-file=astra-registry.apps.ocp
-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"default-ingress-
ca"}}}' --type=merge
```

ルートとともに入力オペレータからのデフォルト TLS 証明書を含む OpenShift 内部レジストリを使用している場合は、前の手順に従って、ルートホスト名に証明書をパッチする必要があります。入力オペレータから証明書を抽出するには、コマンド「 oc extract secret/router-ca --keys=tls.crt-n OpenShift ingress-operator 」を使用します。

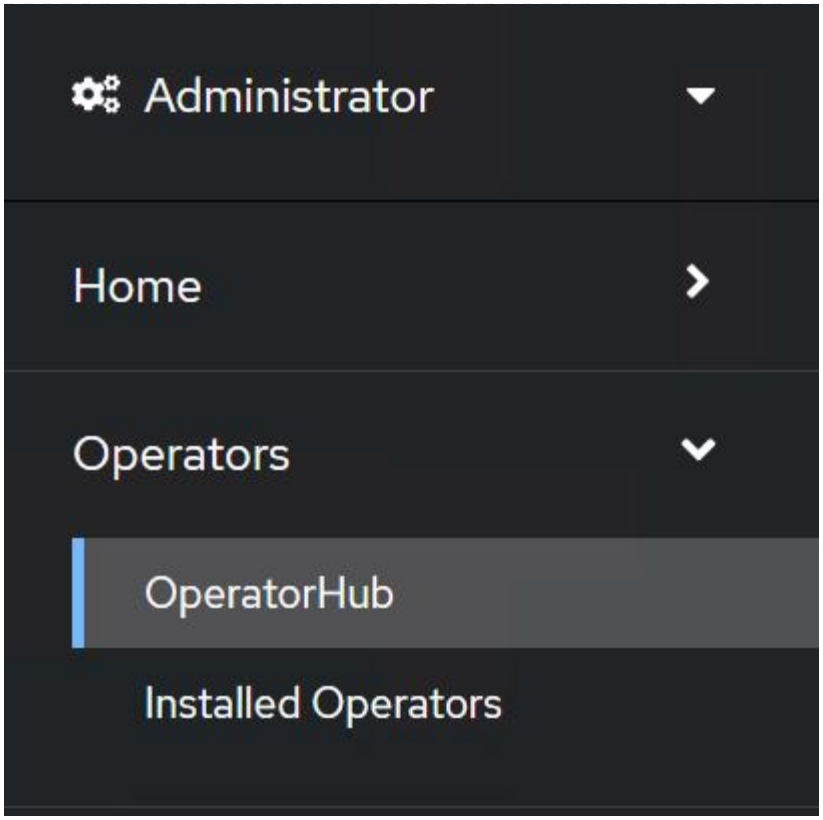
5. Astra Control Center 用の名前空間 NetApp-acc-operator' を作成します

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator

namespace/netapp-acc-operator created
```

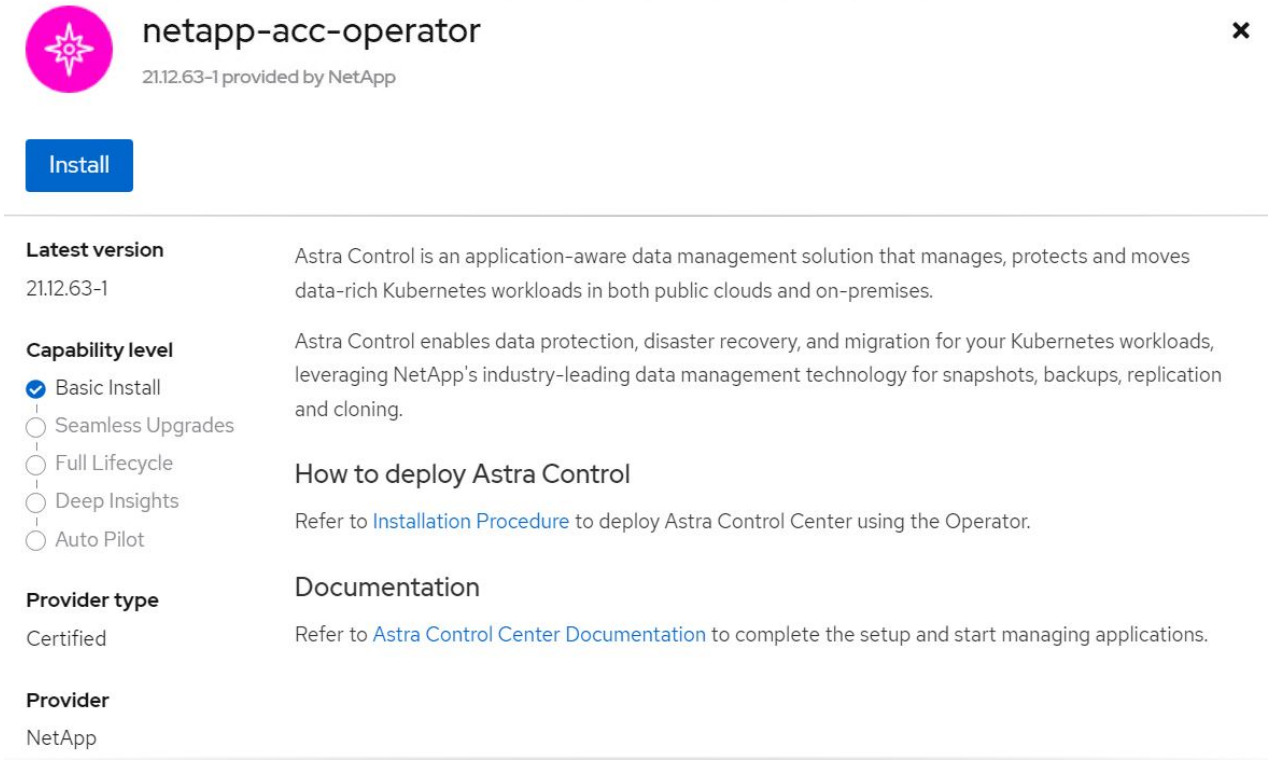
6. NetApp-acc-operator ネームスペースのイメージレジストリにログインするためのクレデンシャルを含むシークレットを作成します。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-
registry-cred --docker-server=astra-registry.apps.ocp
-vmw.cie.netapp.com --docker-username=ocp-user --docker
-password=password -n netapp-acc-operator

secret/astra-registry-cred created
```

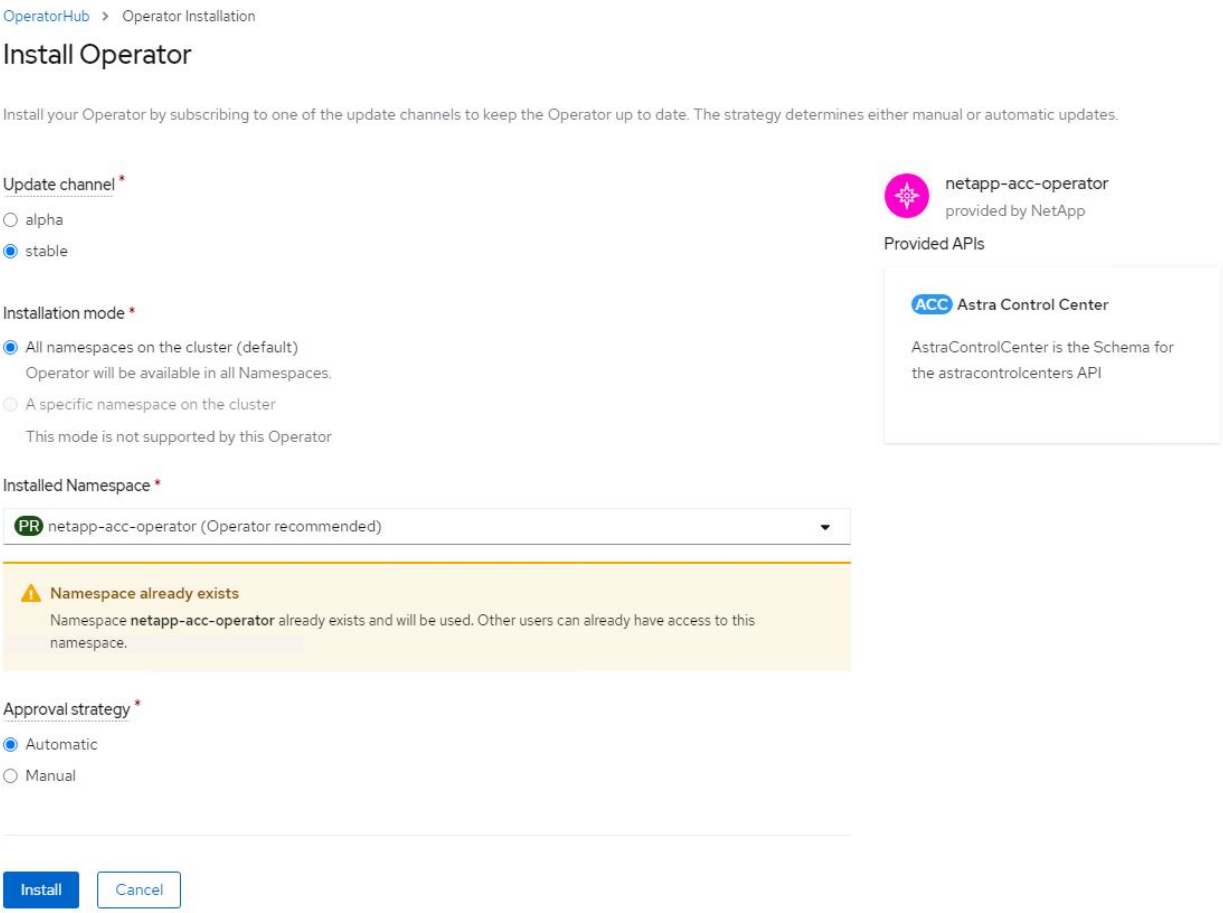
7. クラスタ管理者アクセスで Red Hat OpenShift GUI コンソールにログインします。
8. Perspective ドロップダウンから Administrator を選択します。
9. [演算子]>[演算子ハブ] の順に移動し、 Astra を検索します。



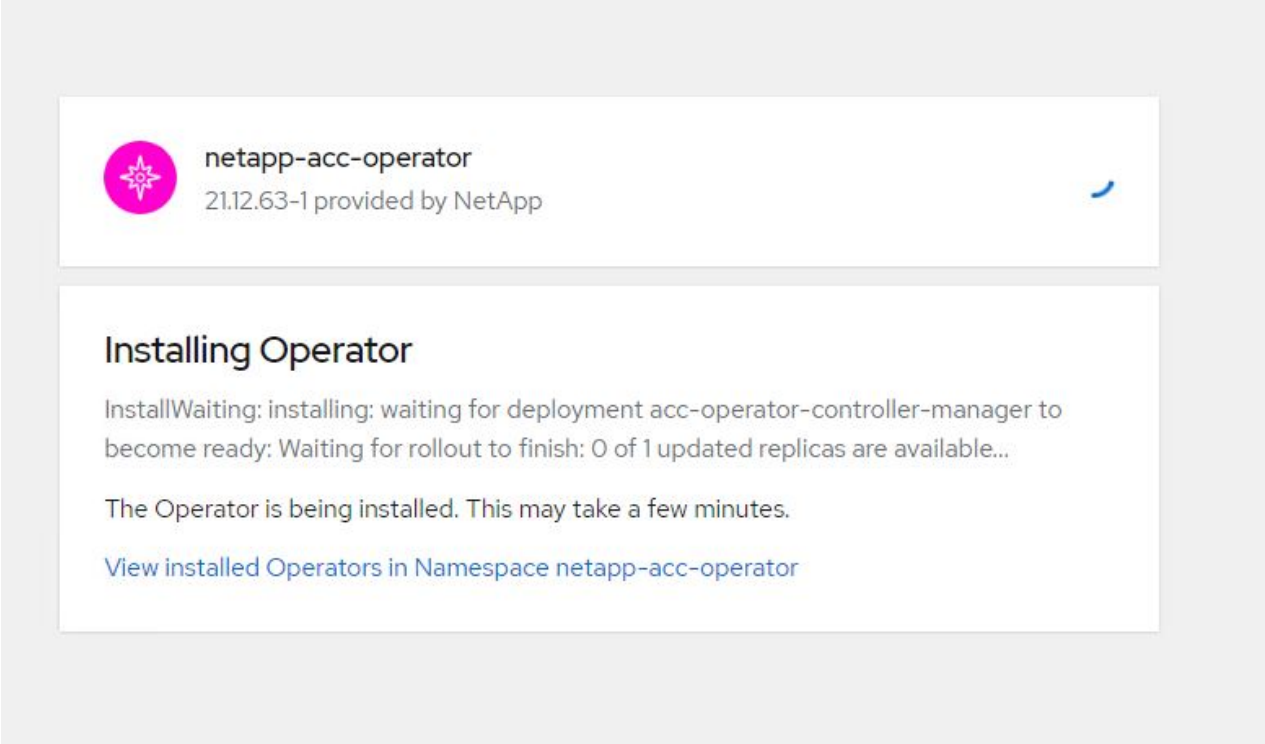
10. NetApp-acc-operator' タイルを選択し、 [インストール] をクリックします。



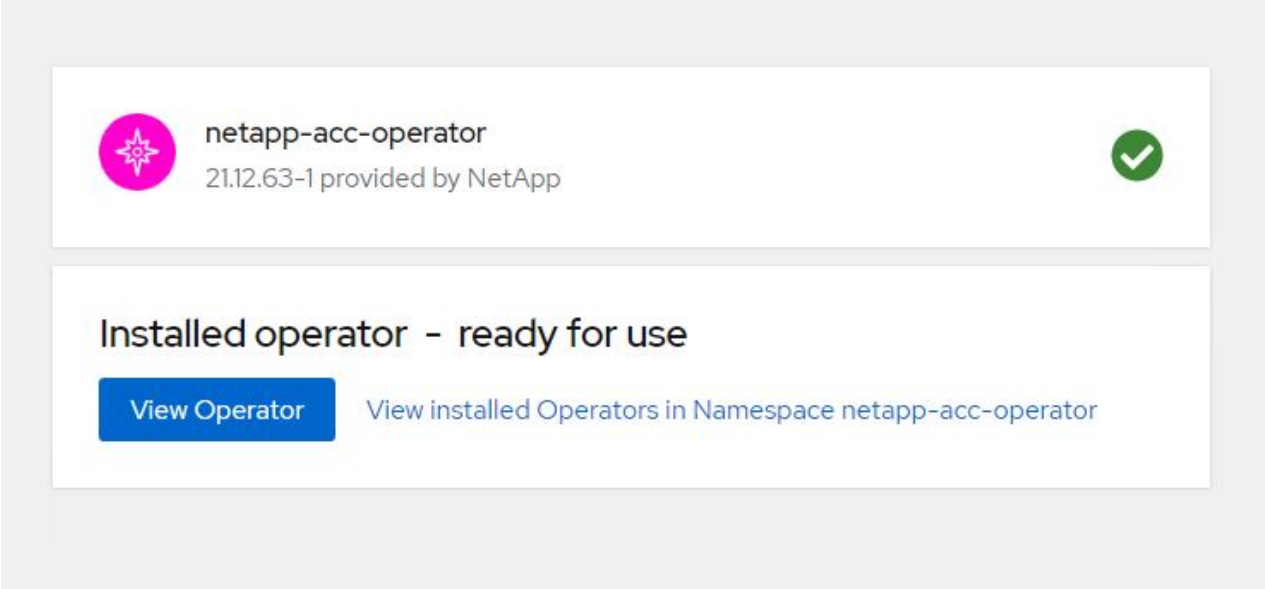
11. インストールオペレータ画面で、デフォルトのパラメータをすべて受け入れて、「インストール」をクリックします。



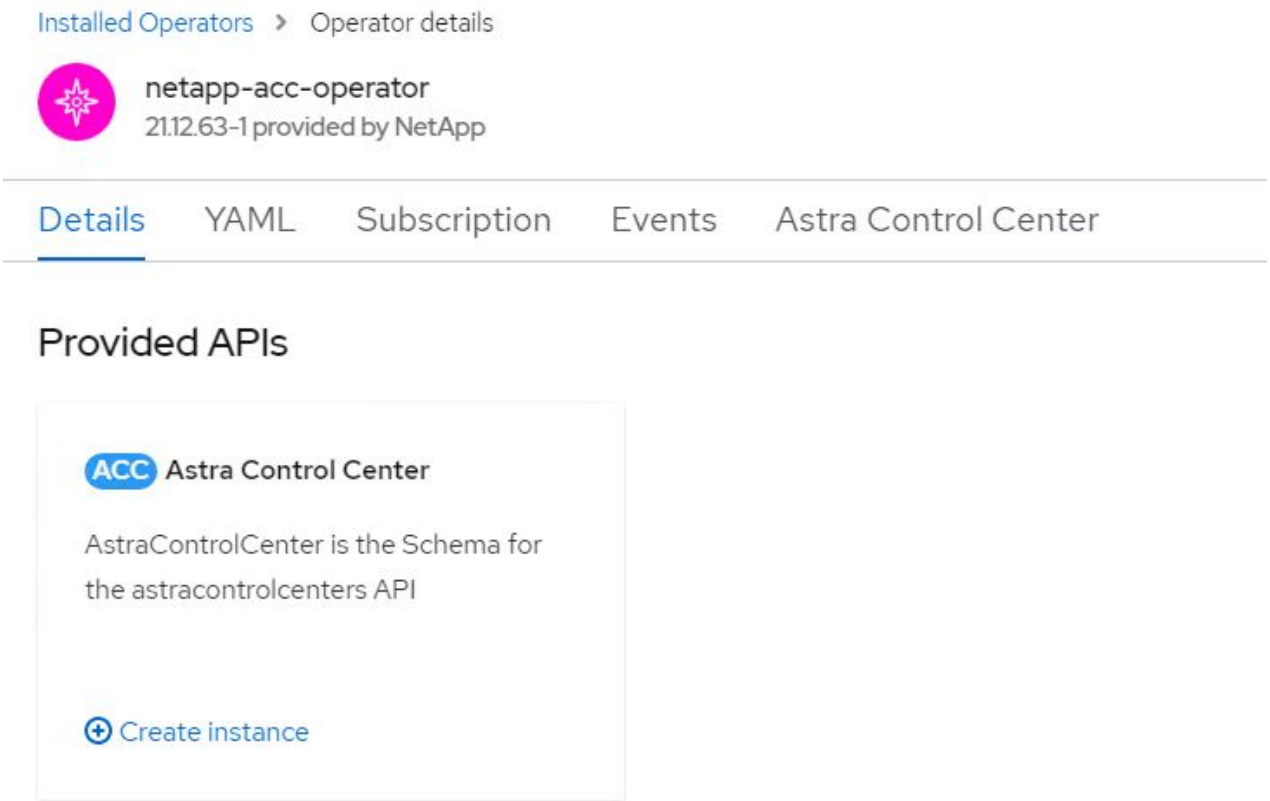
12. オペレータによるインストールが完了するまで待ちます。



13. オペレータのインストールが完了したら、 [View Operator] をクリックします。



14. 次に、オペレーターの Astra Control Center タイルで [Create Instance] をクリックします。



15. [Create AstraeControl] フォームフィールドに入力し '[Create] をクリックします

    a. 必要に応じて、 Astra Control Center インスタンス名を編集します。
    b. 必要に応じて、 AutoSupport を有効または無効にします。Auto Support 機能の保持を推奨します。
    c. Astra Control Center の FQDN を入力します。
    d. Astra Control Center のバージョンを入力します。デフォルトで最新のバージョンが表示されま

す。

e. Astra Control Center のアカウント名を入力し、管理者の詳細（名、姓、メールアドレスなど）を入力します。

f. ボリューム再利用ポリシーを入力します。デフォルトは Retain です。

g. Image Registry に、レジストリの FQDN と、イメージをレジストリにプッシュする際に指定した組織名を入力します（この例では「 astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra`」）。

h. 認証が必要なレジストリを使用する場合は、 [イメージレジストリ] セクションにシークレット名を入力します。

i. Astra Control Center のリソース制限のスケーリングオプションを設定します。

j. デフォルト以外のストレージクラスに PVC を配置する場合は、ストレージクラス名を入力します。

k. CRD 処理の環境設定を定義します。

Project: netapp-acc-operator

Name *

astra

Labels

app=frontend

Account Name *

HCG Solutions Engineering

Astra Control Center account name

Astra Address *

astra-control-center.cie.netapp.com

AstraAddress defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center. Example - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control Center

Astra Version *

21.12.60

Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch

Email *

solutions_tme@netapp.com

EmailAddress will be notified by Astra as events warrant.

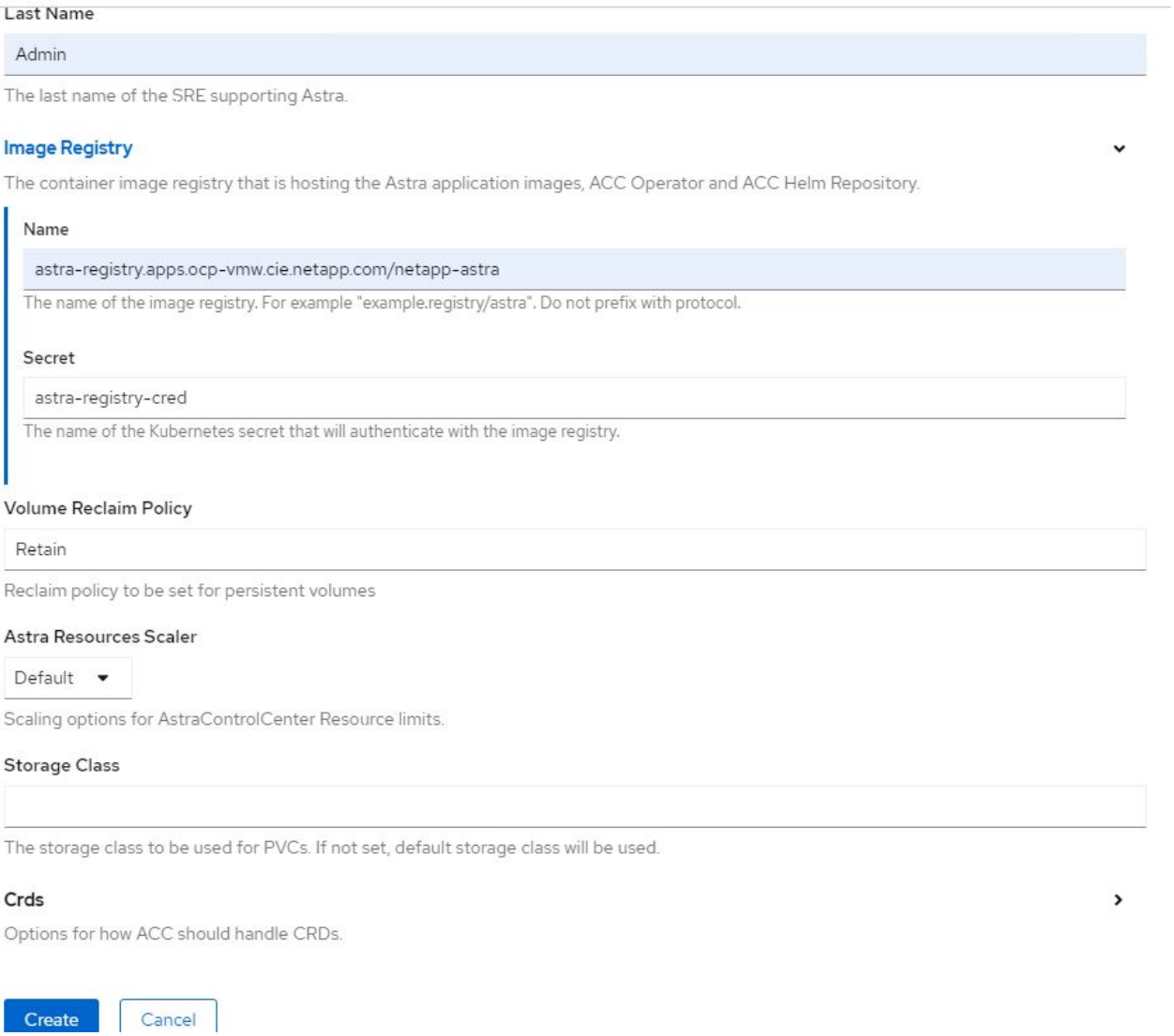
Auto Support *

AutoSupport indicates willingness to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and indicates support data will be sent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should enter false.

First Name

HCG

The first name of the SRE supporting Astra.



自動化された **[Ansible]**

1. Ansibleプレイブックを使用してAstra Control Centerを導入するには、Ansibleがインストールされたユbuntu / RHELマシンが必要です。手順に従います "こちらをご覧ください" UbuntuおよびRHELの場合。
2. Ansible コンテンツをホストする GitHub リポジトリをクローニングします。

```
git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
```

3. ネットアップサポートサイトにログインし、最新バージョンのNetApp Astra Control Centerをダウンロードします。そのためには、ネットアップアカウントにライセンスを関連付ける必要があります。tar ファイルをダウンロードしたら、ワークステーションに転送します。

> Astra Control の試用版ライセンスの使用を開始するには、にアクセスしてください "Astra 登録サイト"。

4. Astra Control CenterをインストールするOpenShiftクラスタにadminとしてアクセスし、kubeconfig

ファイルを作成または取得します。

5. ディレクトリを na_Astra_control_site に変更します。

```
cd na_astra_control_suite
```

6. 「vars/vars.yml」ファイルを編集し、必要な情報を変数に入力します。

```
#Define whether or not to push the Astra Control Center images to
your private registry [Allowed values: yes, no]
push_images: yes

#The directory hosting the Astra Control Center installer
installer_directory: /home/admin/

#Specify the ingress type. Allowed values - "AccTraefik" or
"Generic"
#"AccTraefik" if you want the installer to create a LoadBalancer
type service to access ACC, requires MetalLB or similar.
#"Generic" if you want to create or configure ingress controller
yourself, installer just creates a ClusterIP service for traefik.
ingress_type: "AccTraefik"

#Name of the Astra Control Center installer (Do not include the
extension, just the name)
astra_tar_ball_name: astra-control-center-22.04.0

#The complete path to the kubeconfig file of the
kubernetes/openshift cluster Astra Control Center needs to be
installed to.
hosting_k8s_cluster_kubeconfig_path: /home/admin/cluster-
kubeconfig.yml

#Namespace in which Astra Control Center is to be installed
astra_namespace: netapp-astra-cc

#Astra Control Center Resources Scaler. Leave it blank if you want
to accept the Default setting.
astra_resources_scaler: Default

#Storageclass to be used for Astra Control Center PVCs, it must be
created before running the playbook [Leave it blank if you want the
PVCs to use default storageclass]
astra_trident_storageclass: basic

#Reclaim Policy for Astra Control Center Persistent Volumes [Allowed
```

```
values: Retain, Delete]
storageclass_reclaim_policy: Retain

#Private Registry Details
astra_registry_name: "docker.io"

#Whether the private registry requires credentials [Allowed values:
yes, no]
require_reg_creds: yes

#If require_reg_creds is yes, then define the container image
registry credentials
#Usually, the registry namespace and usernames are same for
individual users
astra_registry_namespace: "registry-user"
astra_registry_username: "registry-user"
astra_registry_password: "password"

#Kuberenets/OpenShift secret name for Astra Control Center
#This name will be assigned to the K8s secret created by the
playbook
astra_registry_secret_name: "astra-registry-credentials"

#Astra Control Center FQDN
acc_fqdn_address: astra-control-center.cie.netapp.com

#Name of the Astra Control Center instance
acc_account_name: ACC Account Name

#Administrator details for Astra Control Center
admin_email_address: admin@example.com
admin_first_name: Admin
admin_last_name: Admin
```

7. プレイブックを実行して Astra Control Center を導入します。Playbookには、特定の構成用のroot権限が必要です。

   このプレイブックを実行しているユーザがrootである場合、またはパスワードを使用しないsudoが設定されている場合は、次のコマンドを実行してプレイブックを実行します。

   ```
   ansible-playbook install_acc_playbook.yml
   ```

   ユーザにパスワードベースのsudoアクセスが設定されている場合は、次のコマンドを実行してこのPlaybookを実行し、sudoパスワードを入力します。

```
ansible-playbook install_acc_playbook.yml -K
```

インストール後の手順

1. インストールが完了するまでに数分かかることがあります。NetApp-AstrA-cc' ネームスペース内のすべてのポッドとサービスが稼働していることを確認します

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. 「 acc-operator-controller-manager 」ログをチェックし、インストールが完了したことを確認します。

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n
netapp-acc-operator -c manager -f
```

> 次のメッセージは、 Astra Control Center のインストールが正常に完了したことを示します。

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraContro
lCenter","msg":"Successfully Reconciled AstraControlCenter in
[seconds]s","AstraControlCenter":"netapp-astra-
cc/astra","ae.Version":"[21.12.60]"}
```

3. Astra Control Center にログインするためのユーザ名は、 CRD ファイルに提供された管理者の電子メールアドレスで、パスワードは Astra Control Center UUID に付加された文字列「 ACC-` 」です。次のコマンドを実行します。

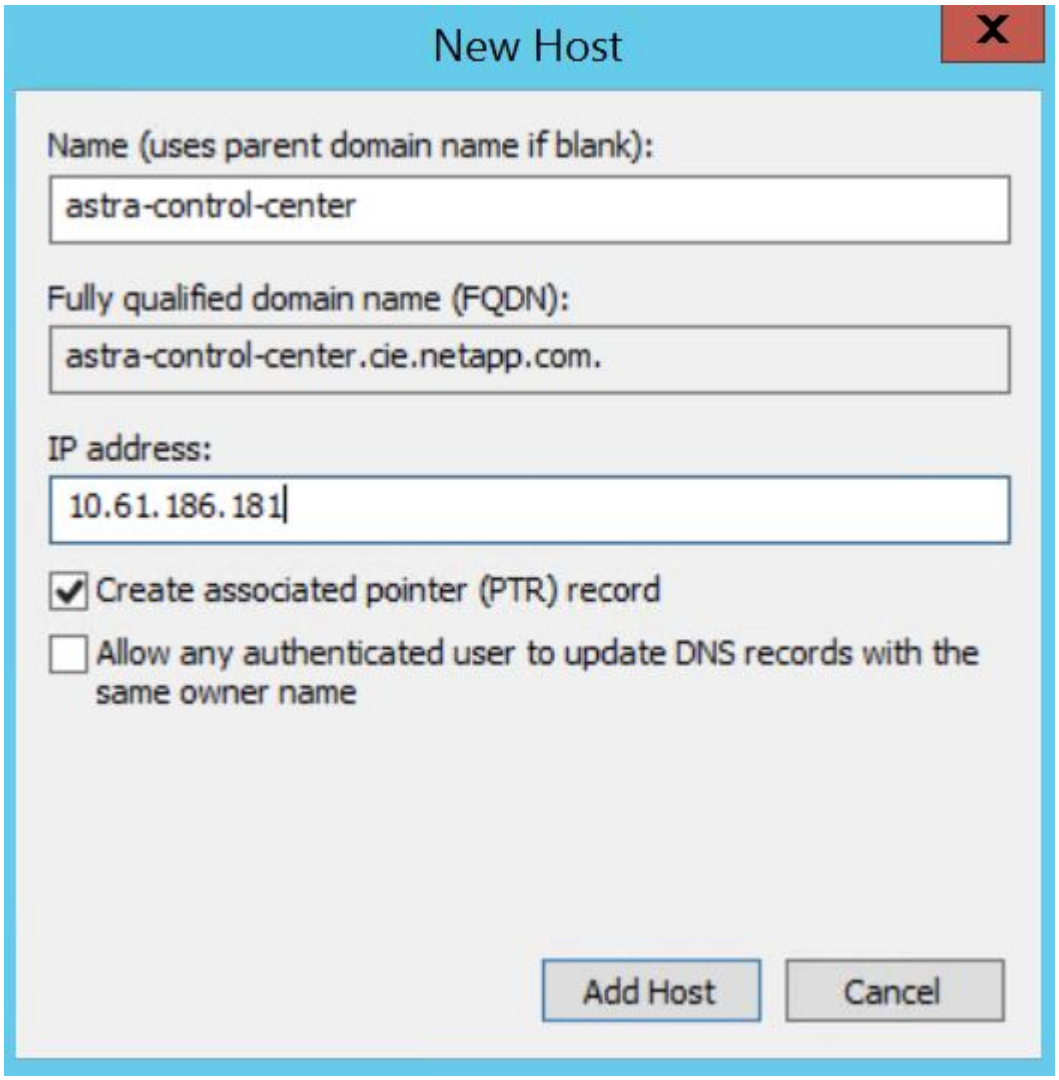
```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME    UUID
astra   345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

> この例では、パスワードは「 ACC-345c55a5 -bf2e-21f0 -84b8 -b6f2bce5e95f 」です。

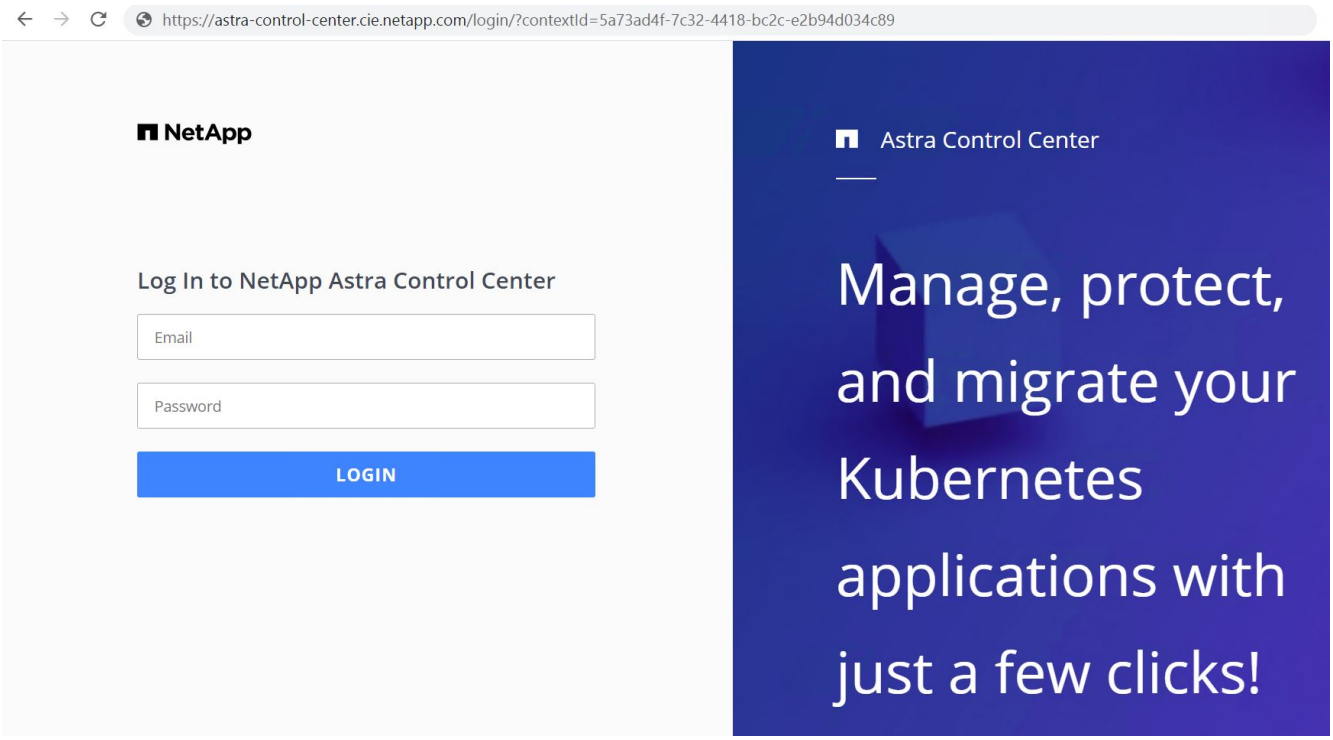
4. traefik サービスのロードバランサ IP を取得します。

```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep
'EXTERNAL|traefik'

NAME                                        TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)
AGE
traefik                                     LoadBalancer   172.30.99.142
10.61.186.181   80:30343/TCP,443:30060/TCP
16m
```

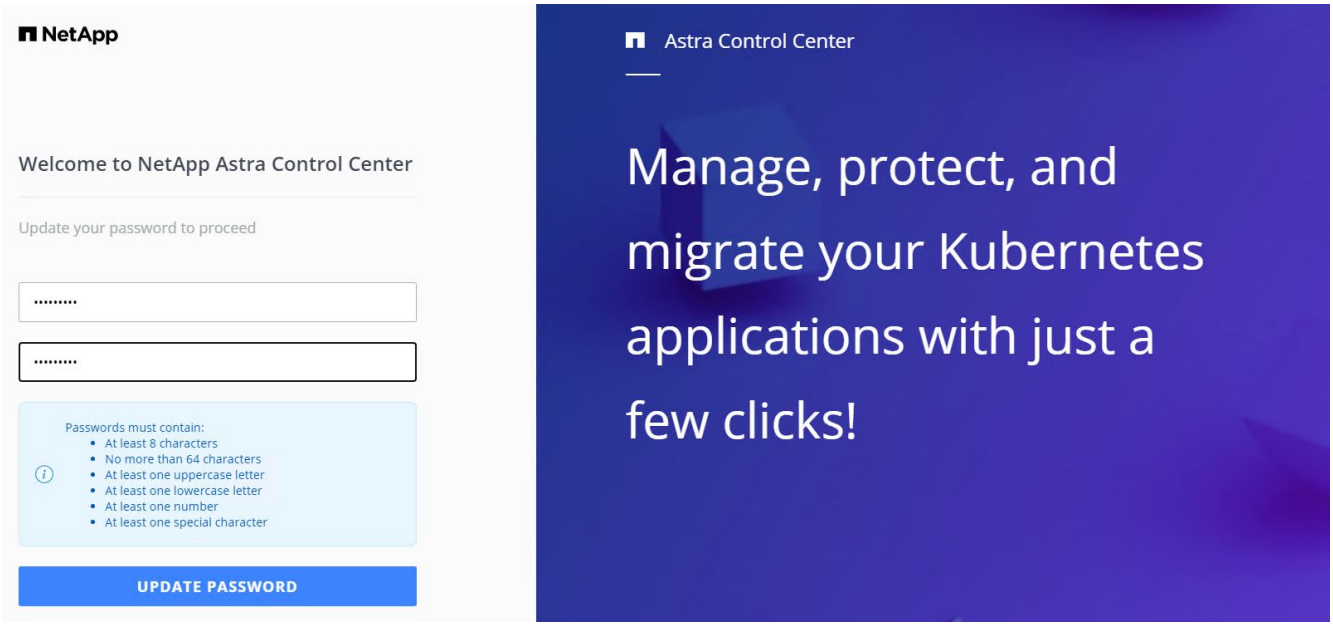
5. Astra Control Center CRD ファイルに指定された FQDN を指す DNS サーバーのエントリを、 traefik サービスの「 external-IP 」に追加します。



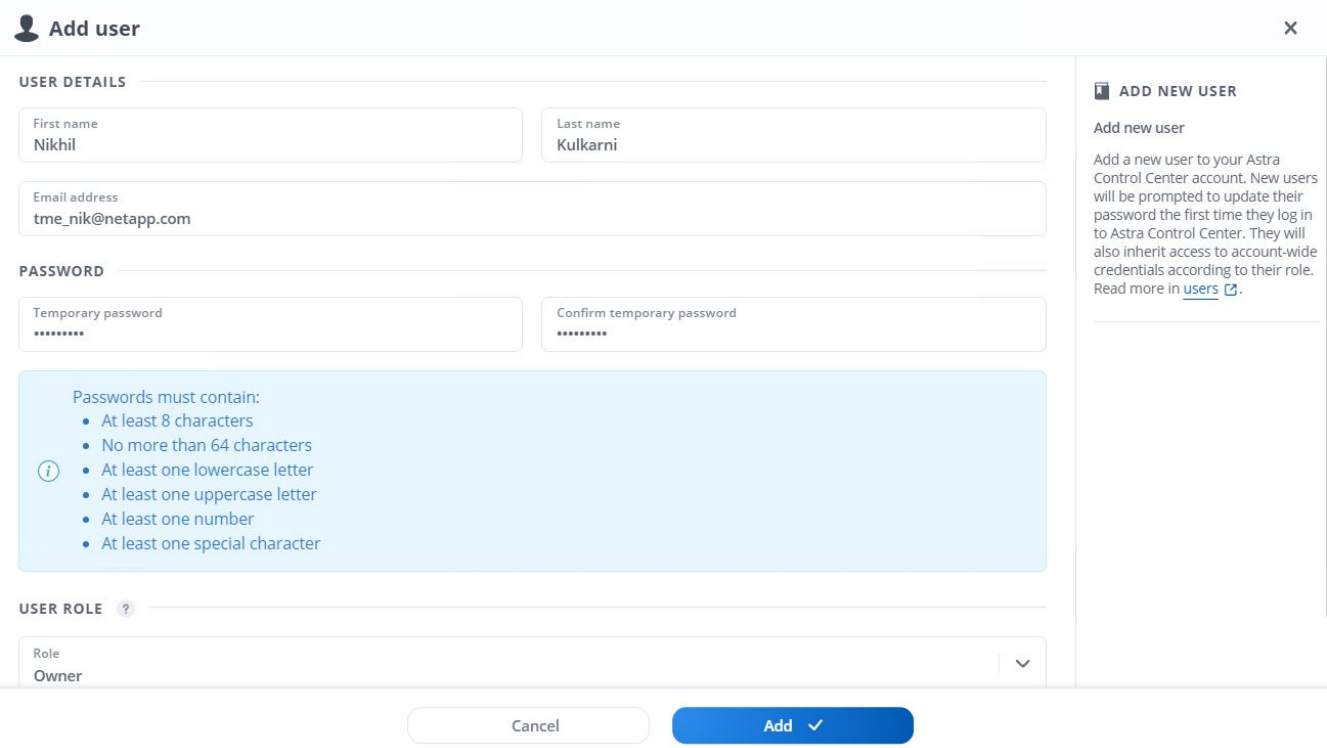
6. Astra Control Center GUI に、 FQDN を参照してログインします。



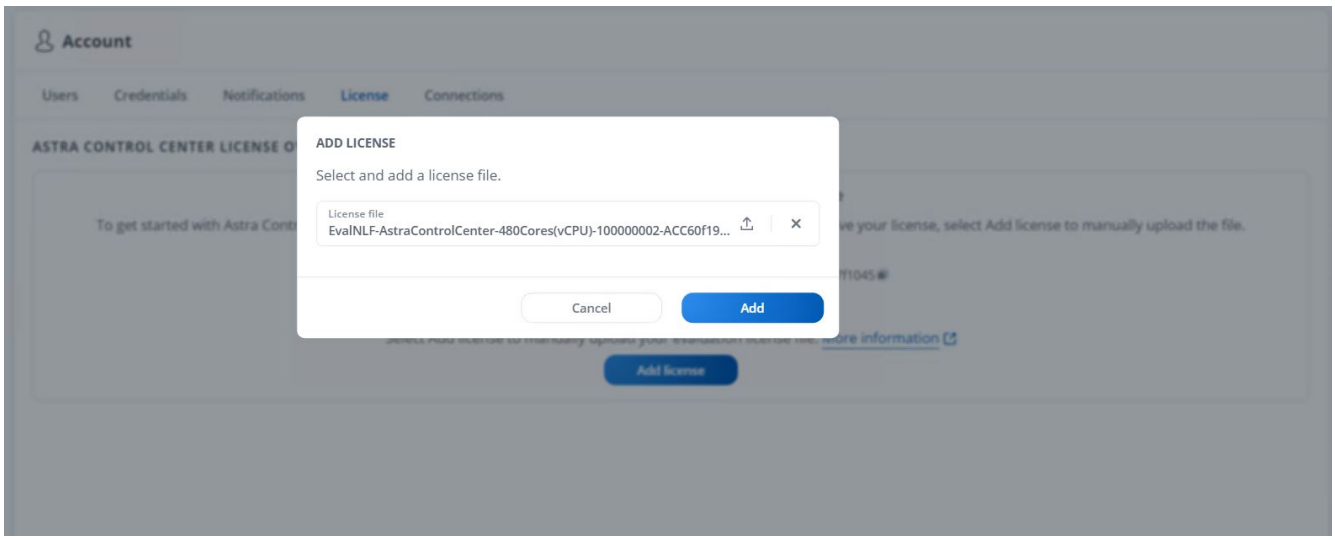
7. CRD で提供された管理者メールアドレスを使用して初めて Astra Control Center GUI にログインする場合は、パスワードを変更する必要があります。



8. ユーザーを Astra Control Center に追加する場合は、 [アカウント]>[ユーザー] の順に選択し、 [追加] をクリックしてユーザーの詳細を入力し、 [追加] をクリックします。



9. Astra Control Center では、すべての機能が動作するためにライセンスが必要です。ライセンスを追加するには、 ［アカウント］ > ［ライセンス］ の順に選択し、 ［ライセンスの追加］ をクリックして、ライセンスファイルをアップロードします。



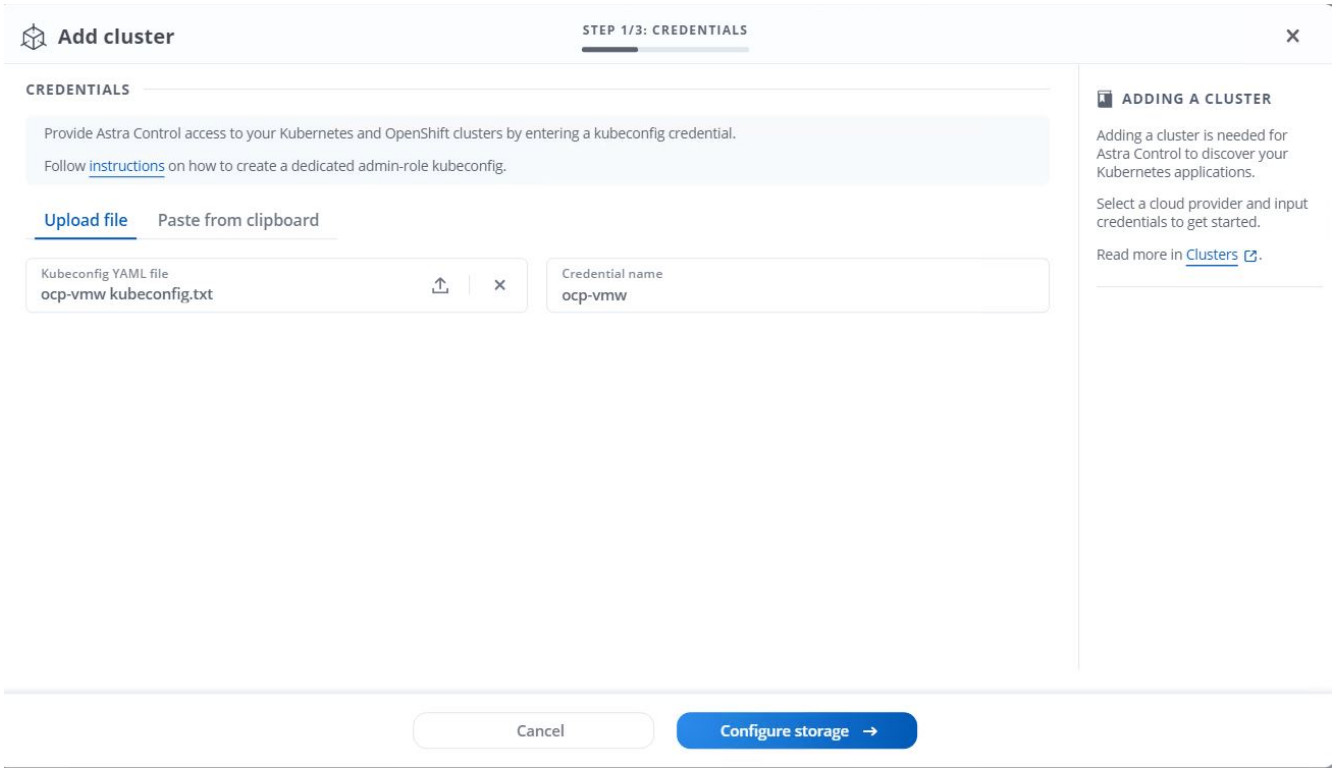
NetApp Astra Control Center のインストールまたは設定で問題が発生した場合は、既知の問題のナレッジベースを利用できます "こちらをご覧ください"。

Red Hat OpenShift クラスタを Astra Control Center に登録します

Astra Control Center でワークロードを管理できるようにするには、まず Red Hat OpenShift クラスタを登録する必要があります。

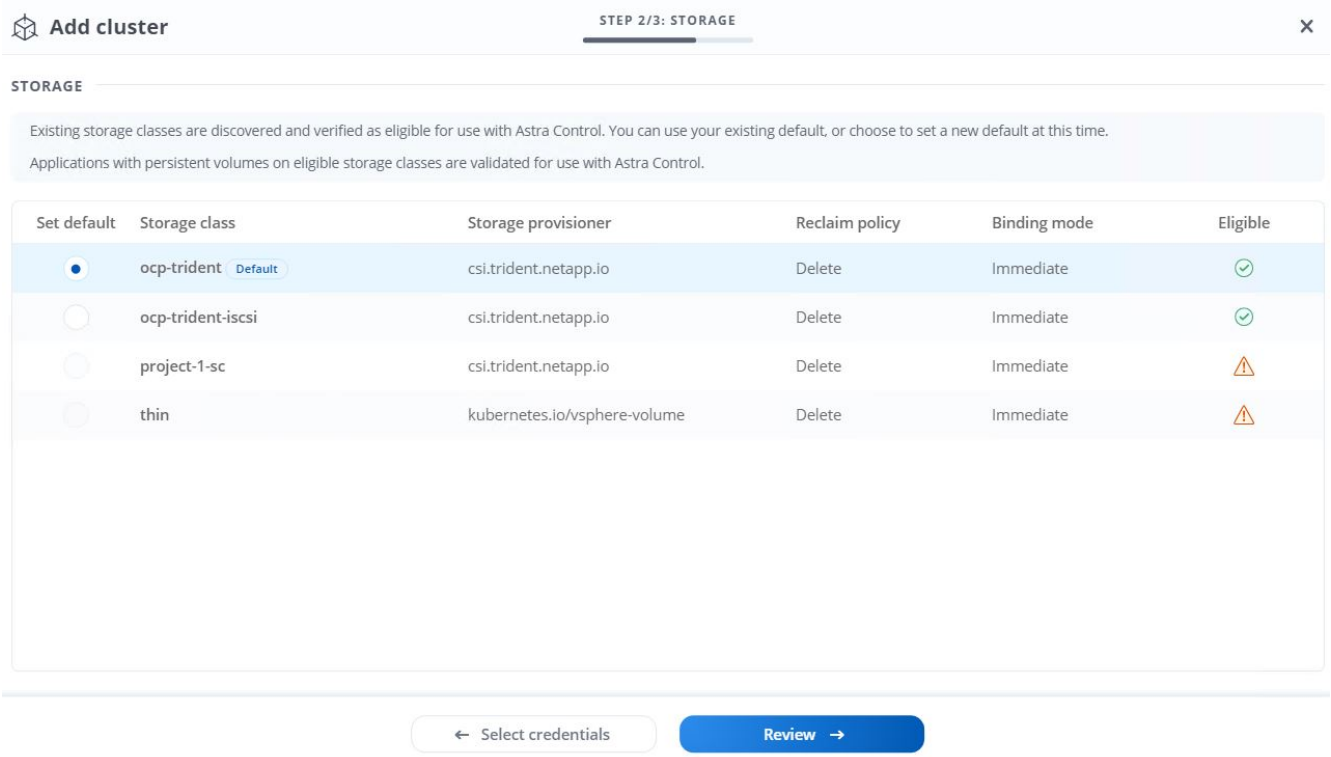
Red Hat OpenShift クラスタを登録します

1. 最初のステップでは、 OpenShift クラスタを Astra Control Center に追加して管理します。クラスタに移動してクラスタの追加をクリックし、 OpenShift クラスタの kubeconfig ファイルをアップロードして、ストレージの選択をクリックします。

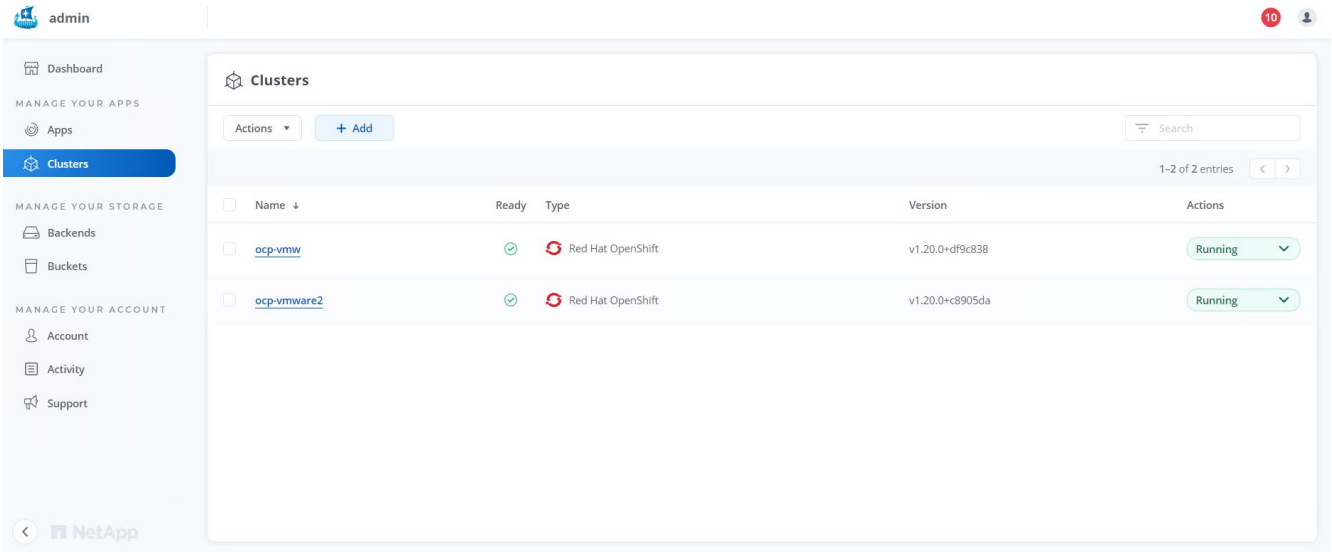


> ユーザ名とパスワードまたはトークンを使用して認証するために kubeconfig ファイルを生成できます。トークンが期限切れになるまでの時間は制限されており、登録されたクラスタに到達できなくなる可能性があります。ネットアップでは、 OpenShift クラスタを Astra Control Center に登録するために、ユーザ名とパスワードを付けた kubeconfig ファイルを使用することを推奨します。

2. Astra Control Center で、対象となるストレージクラスが検出される。次に、ストレージクラスが NetApp ONTAP 上の SVM がサポートする Trident を使用してボリュームをプロビジョニングする方法を選択し、Review （確認）をクリックします。次のペインで詳細を確認し、 Add Cluster をクリックします。

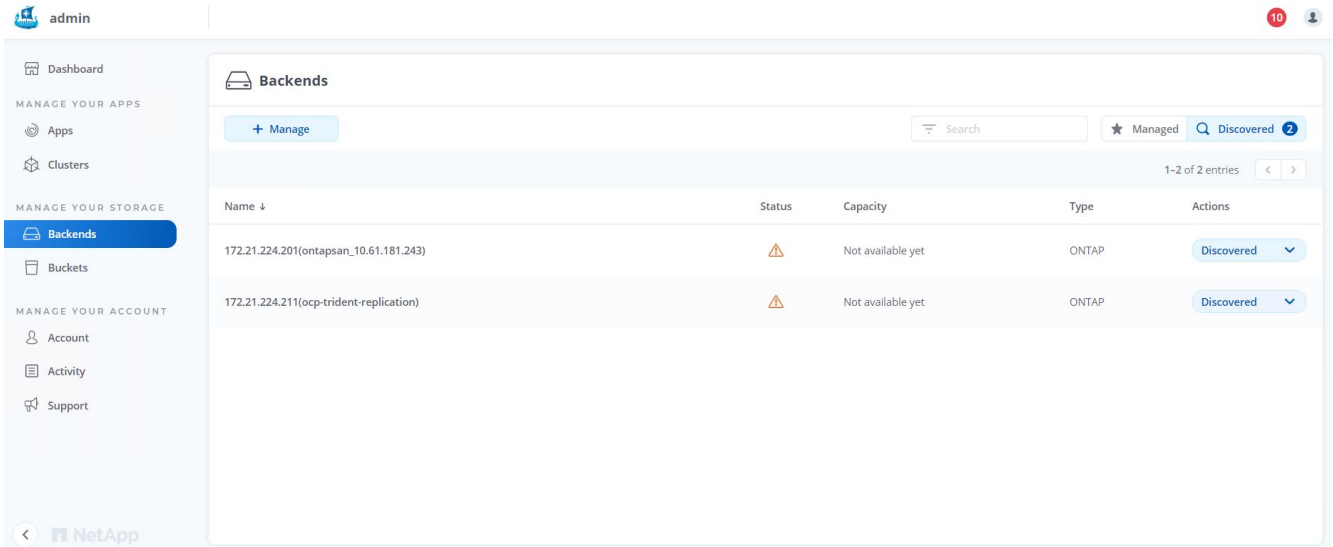


3. 手順 1 の説明に従って、両方の OpenShift クラスタを登録します。追加すると、 Astra Control Center がクラスタを検査して必要なエージェントをインストールしながら、クラスタは Discovering ステータスに移行します。クラスタが登録されると、クラスタのステータスが「 Running 」に変わります。

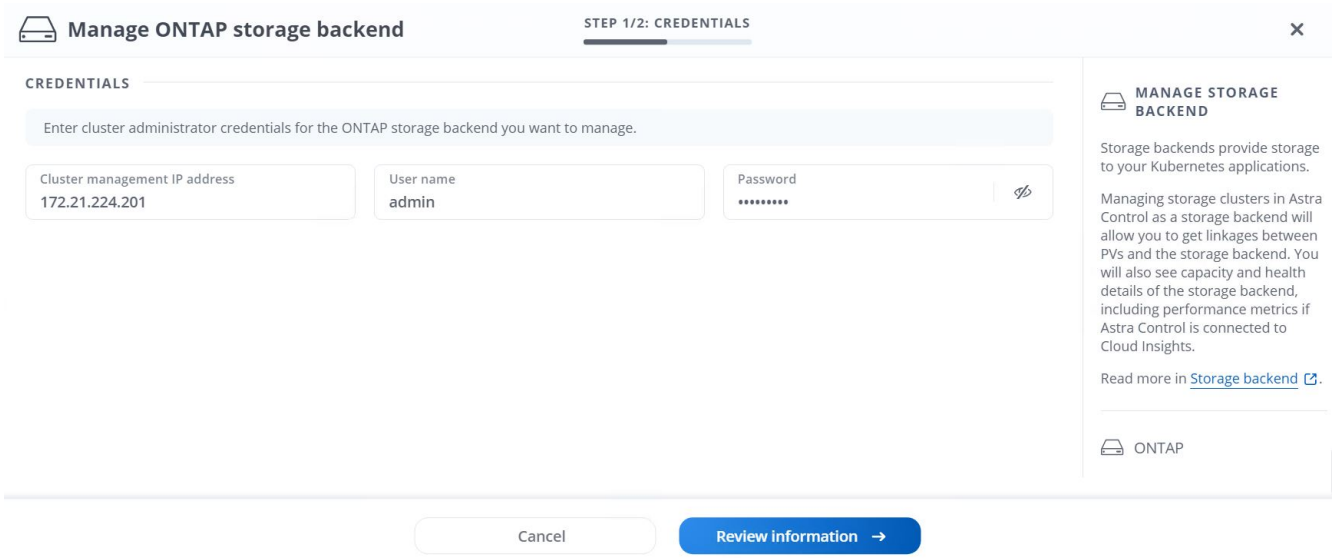


Astra Control Center で管理するすべての Red Hat OpenShift クラスタは、管理対象クラスタにインストールされたエージェントとしてインストールに使用されたイメージレジストリにアクセスできる必要があります。このレジストリからイメージがプルされます。

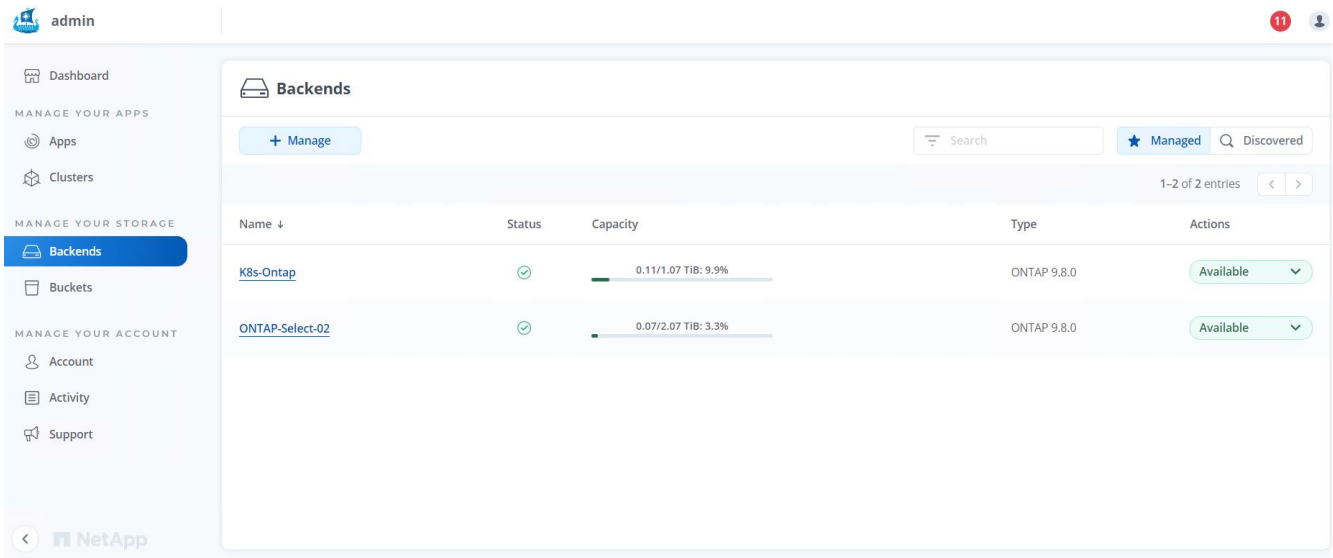
4. ONTAP クラスタをストレージリソースとして Astra Control Center でバックエンドとして管理するようにインポートします。ストレージクラスが設定されている Astra に OpenShift クラスタが追加されると、ストレージクラスをサポートする ONTAP クラスタが自動的に検出されて検査されますが、 Astra コントロールセンターにインポートされて管理されません。



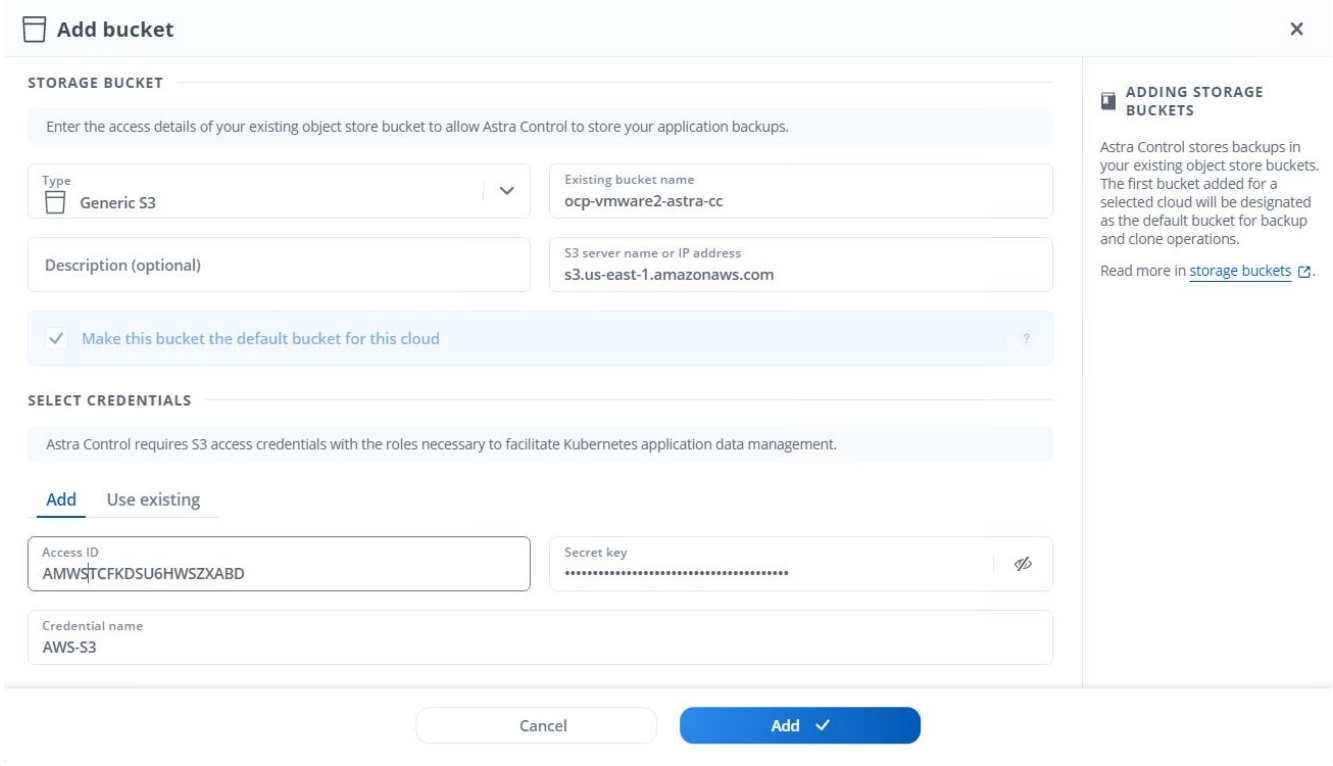
5. ONTAP クラスタをインポートするには、バックエンドに移動し、ドロップダウンをクリックして、管理対象の ONTAP クラスタの横にある Manage を選択します。ONTAP クラスタの資格情報を入力し、 [情報の確認] をクリックして、 [ストレージバックエンドのインポート] をクリックします。



6. バックエンドを追加すると、ステータスが Available に変わります。このバックエンドには、 OpenShift クラスタ内の永続ボリュームと ONTAP システム上の対応するボリュームに関する情報が含まれます。



7. Astra Control Center を使用して OpenShift クラスタ間でバックアップとリストアを行うには、 S3 プロトコルをサポートするオブジェクトストレージバケットをプロビジョニングする必要があります。現在サポートされているオプションは、 ONTAP S3 、 StorageGRID 、および AWS S3 です。このインストールのために、 AWS S3 バケットを設定します。バケットに移動し、バケットの追加をクリックして、汎用 S3 を選択します。S3 バケットの詳細とアクセスするためのクレデンシャルを入力し、「 Make this bucket the default bucket for the cloud 」のチェックボックスをオンにして、 Add をクリックします。

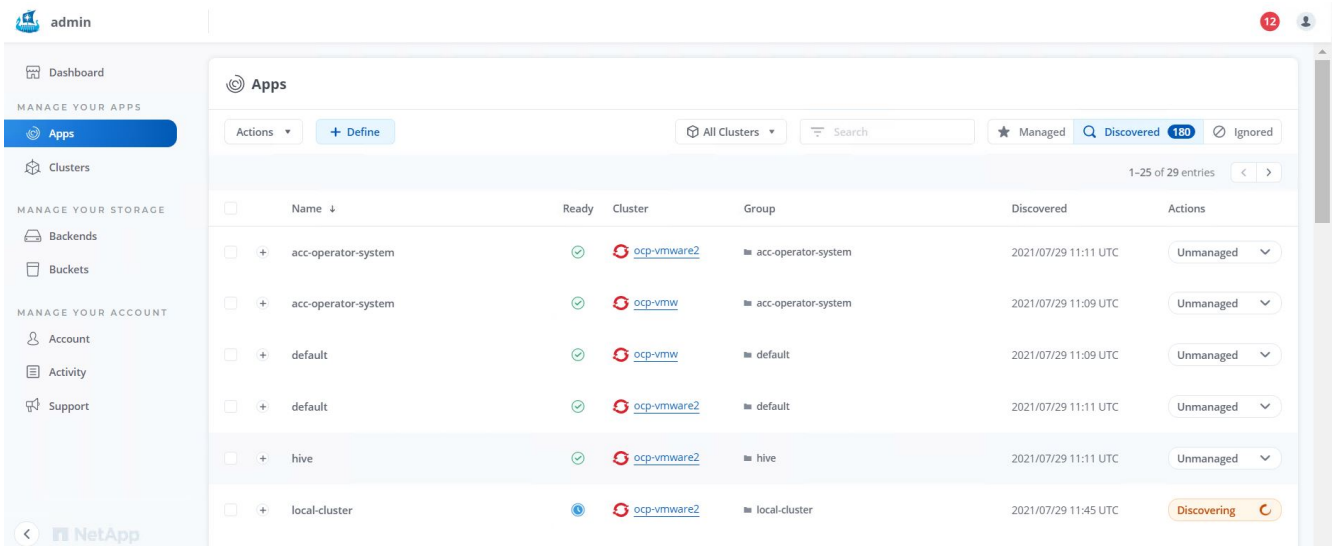


保護するアプリケーションを選択します

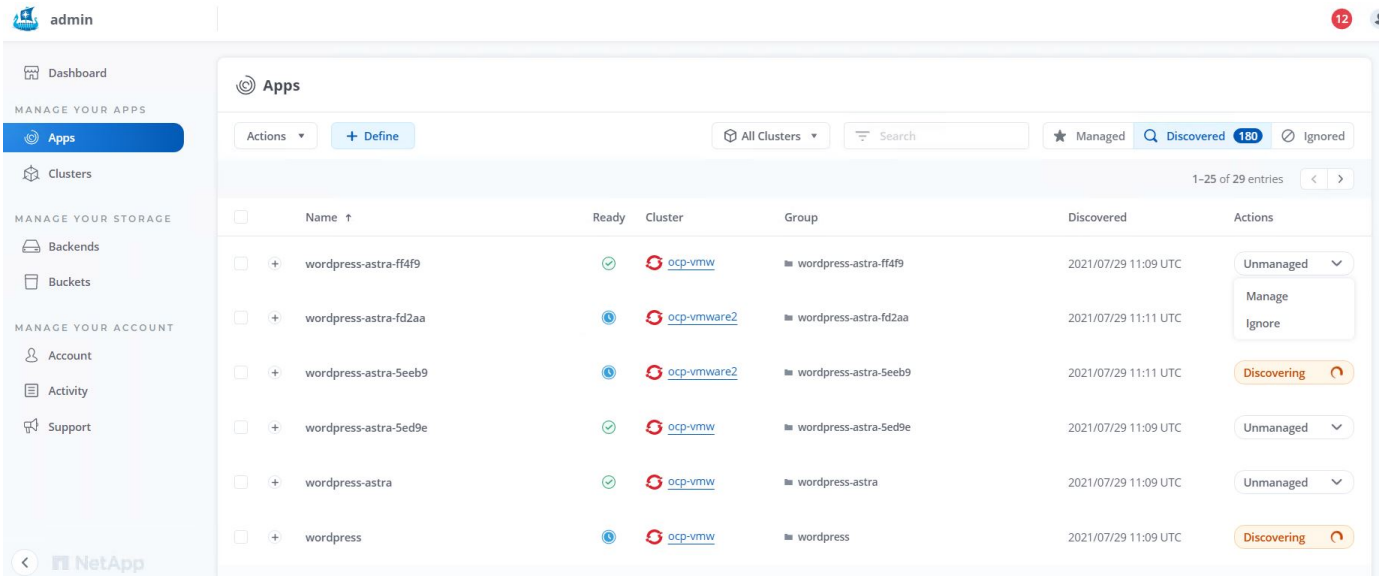
Red Hat OpenShift クラスタを登録したら、 Astra Control Center を使用して導入および管理するアプリケーションを検出できます。

アプリケーションを管理します

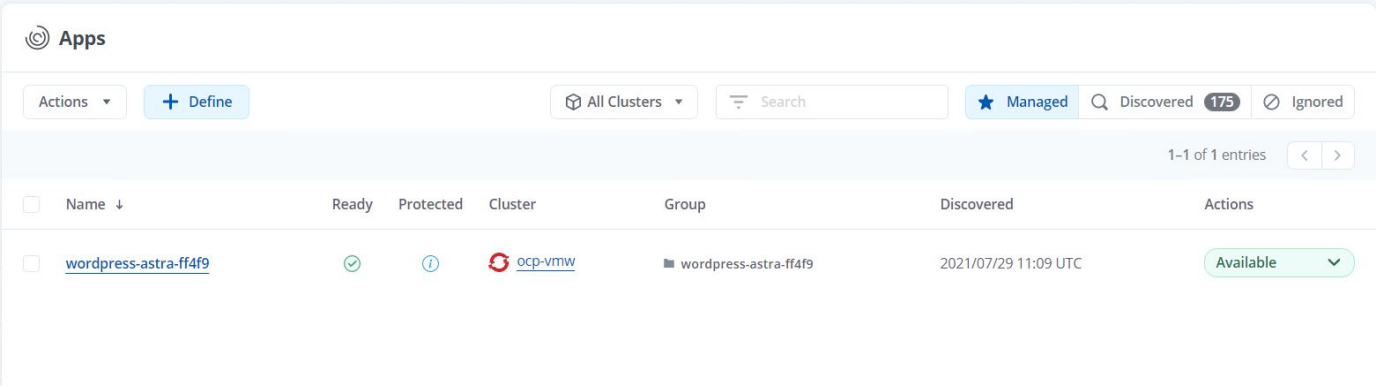
1. OpenShift クラスタと ONTAP バックエンドが Astra Control Center に登録されると、コントロールセンターは、指定した ONTAP バックエンドで構成されたストレージクラスを使用するすべてのネームスペース内のアプリケーションを自動的に検出します。



2. [アプリケーション]>[検出済み] の順に移動し、 Astra を使用して管理するアプリケーションの横にあるドロップダウンメニューをクリックします。[管理] をクリックします。



1. アプリケーションが [使用可能（ Available ）] 状態になり、 [アプリケーション（ Apps ）] セクションの [管理（ Managed ）] タブで表示できます。



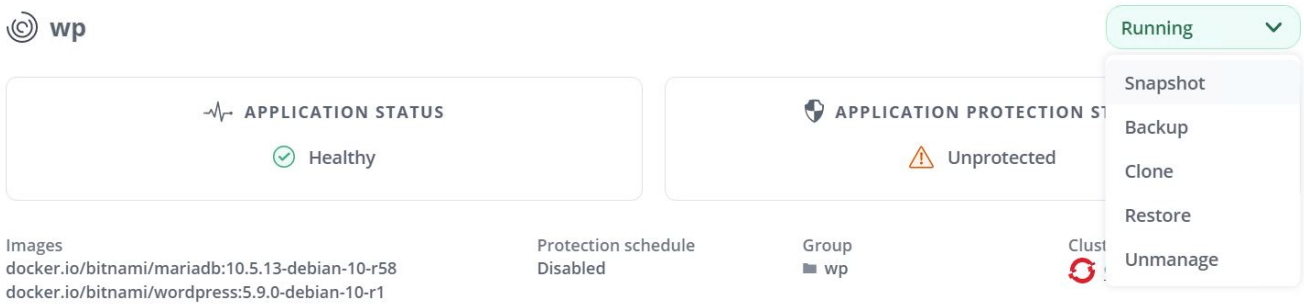
アプリケーションを保護

アプリケーションワークロードを Astra Control Center で管理した後、それらのワークロードの保護設定を構成できます。

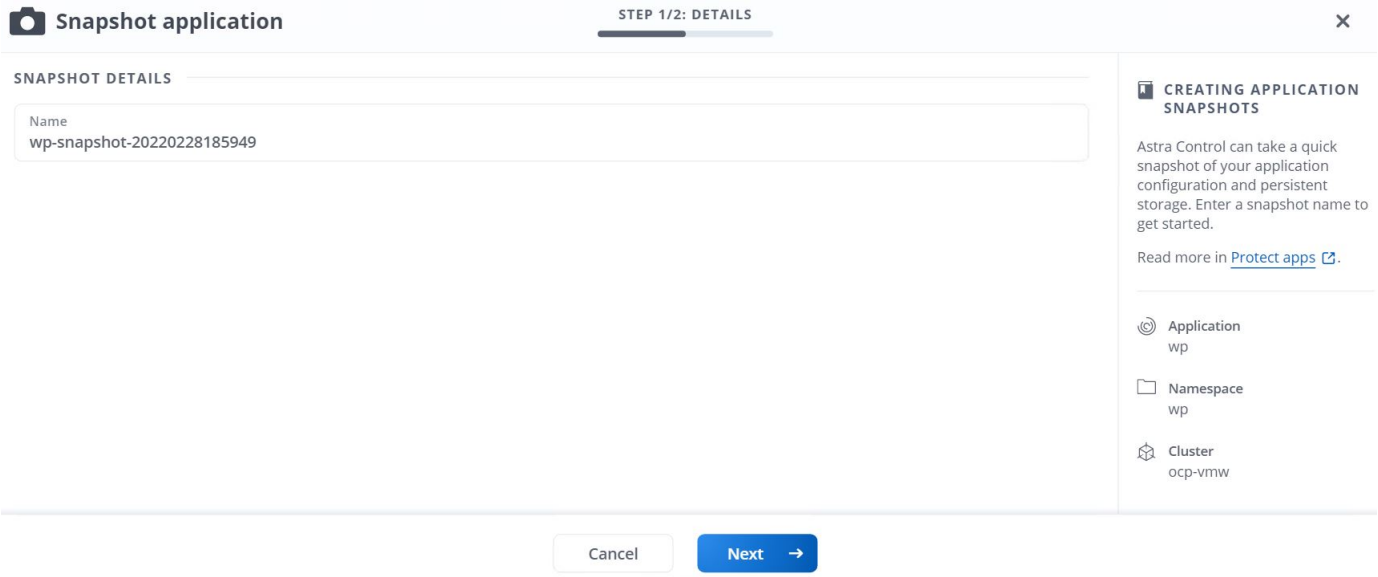
アプリケーションスナップショットを作成しています

アプリケーションの Snapshot コピーを作成すると、 ONTAP Snapshot コピーが作成されます。 Snapshot コピーに基づいて、アプリケーションを特定の時点にリストアまたはクローニングできます。

1. アプリケーションのスナップショットを作成するには、 [アプリ] > [管理] タブに移動し、 Snapshot コピーを作成するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 Snapshot をクリックします。



2. スナップショットの詳細を入力し、 [次へ] をクリックして、 [スナップショット] をクリックします。Snapshot の作成には約 1 分かかり、作成が完了するとステータスを確認できるようになります。



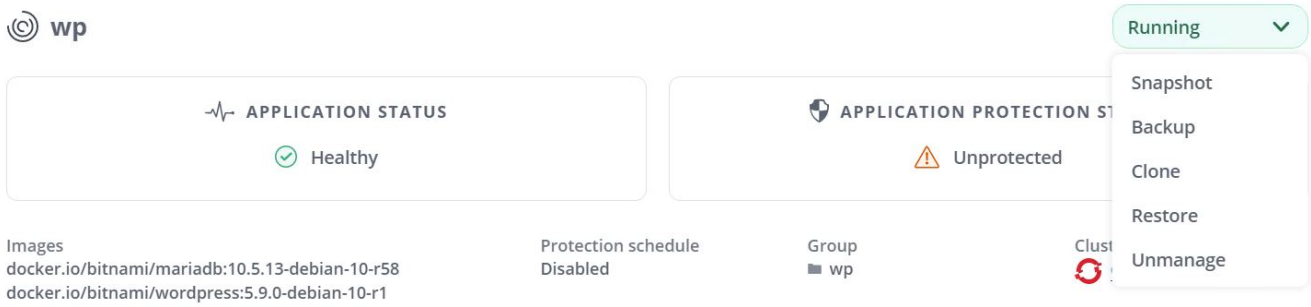
アプリケーションのバックアップを作成しています

アプリケーションのバックアップは、アプリケーションのアクティブな状態とそのリソースの設定をキャプチャしてファイルに変換し、リモートのオブジェクトストレージバケットに格納します。

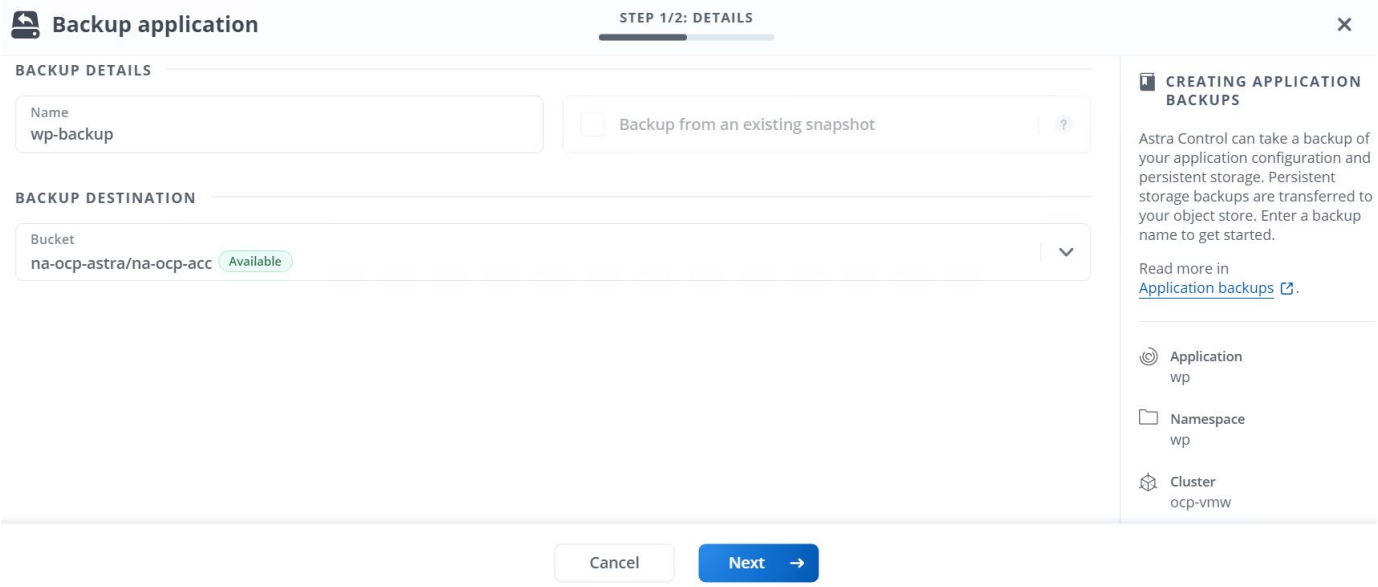
Astra Control Center で管理対象アプリケーションのバックアップとリストアを行うには、バッキング ONTAP システムのスーパーユーザ設定を前提条件として設定する必要があります。そのためには、次のコマンドを入力します。

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. Astra Control Center で管理対象アプリケーションのバックアップを作成するには、 [アプリ] > [管理] タブに移動し、バックアップを作成するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 [バックアップ] をクリックします。



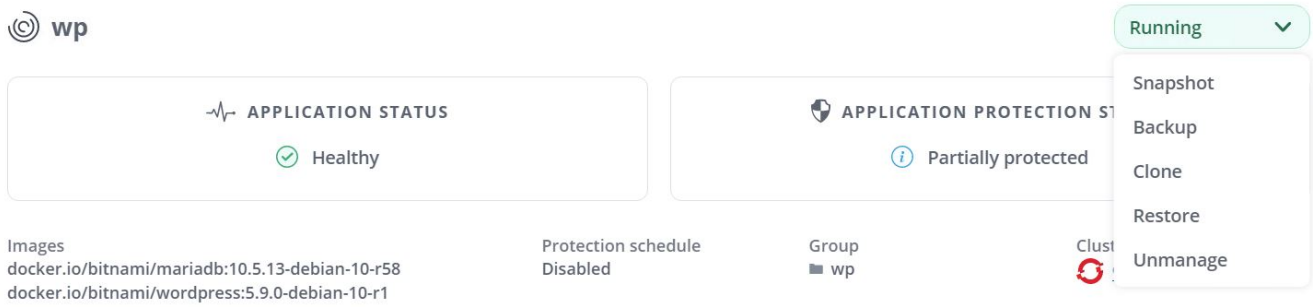
2. バックアップの詳細を入力し、バックアップファイルを保存するオブジェクトストレージバケットを選択して次へをクリックします。詳細を確認したら、バックアップをクリックします。アプリケーションのサイズとデータによっては、バックアップに数分かかることがあり、バックアップが正常に完了したあとでバックアップのステータスを確認できるようになります。



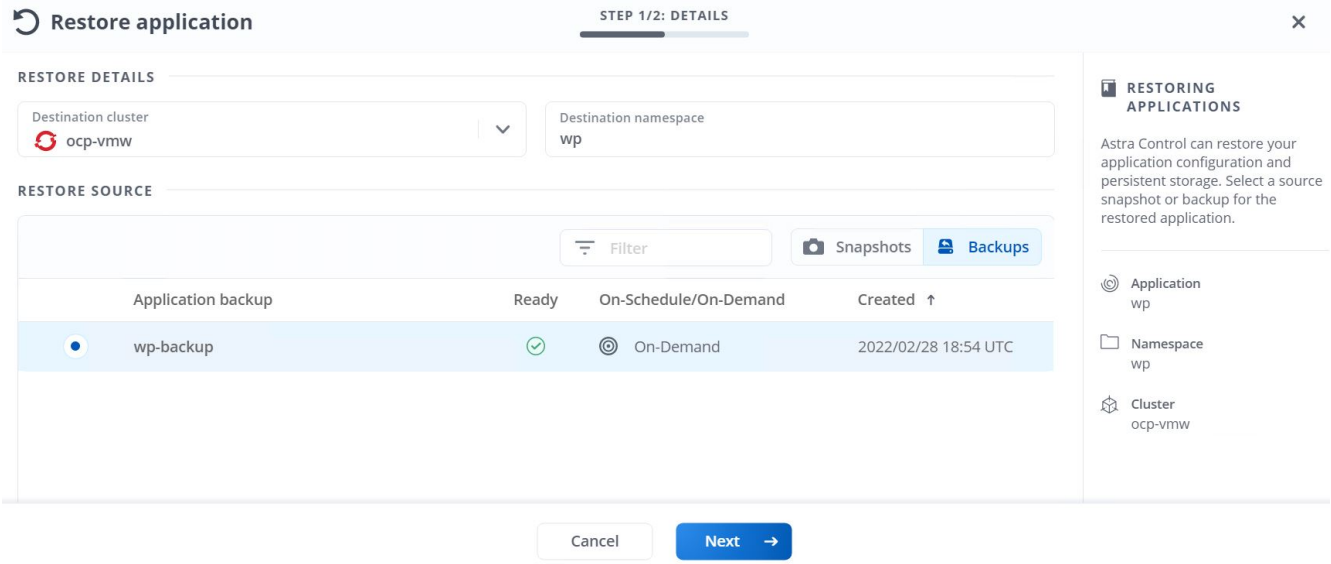
アプリケーションのリストア

ボタンを押すだけで、アプリケーションを同じクラスタ内の元のネームスペースまたはリモートクラスタにリストアし、アプリケーションを保護してディザスタリカバリに使用できます。

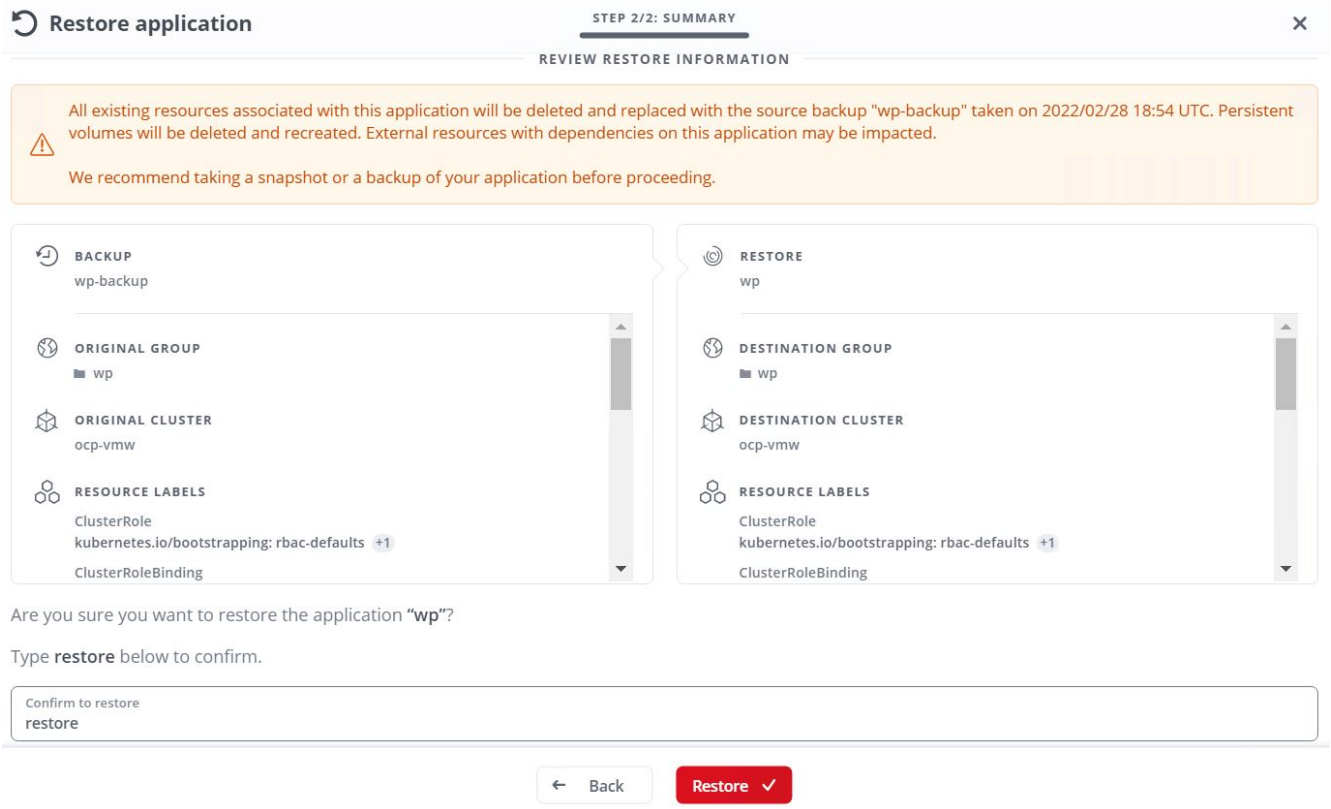
1. アプリケーションを復元するには、 [アプリ] > [管理] タブに移動し、該当するアプリをクリックします。アプリケーション名の横にあるドロップダウン・メニューをクリックし '[リストア] をクリックします



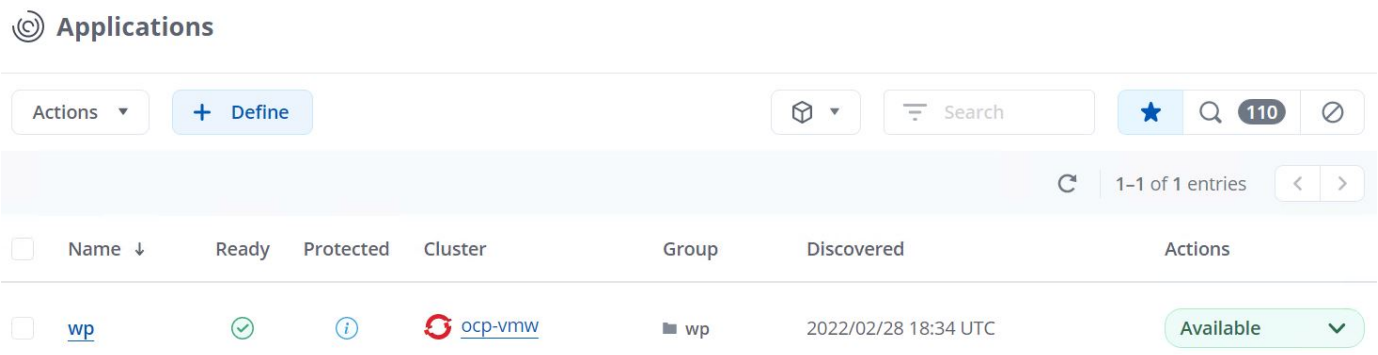
2. リストアネームスペースの名前を入力し、リストア先のクラスタを選択して、既存の Snapshot からリストアするかアプリケーションのバックアップからリストアするかを選択します。次へをクリックします。



3. レビューペインで「 restore 」と入力し、詳細を確認した後で「 Restore 」をクリックします。



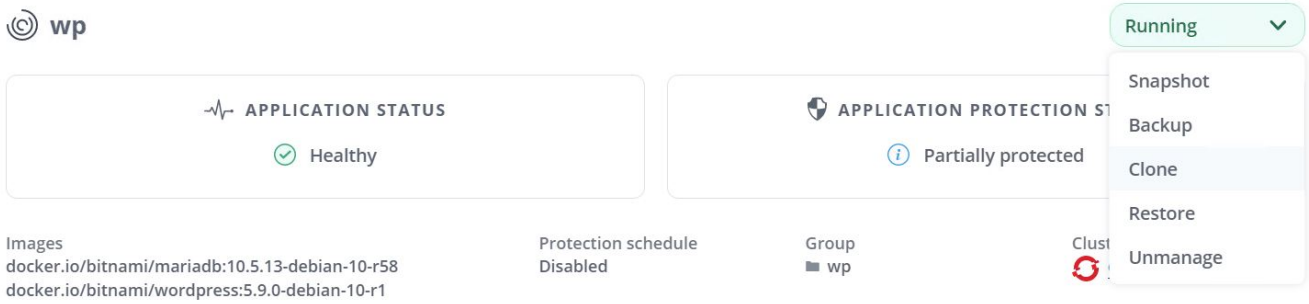
4. 新しいアプリケーションは、 Astra Control Center が選択したクラスタ上のアプリケーションを復元している間、 Restoring 状態になります。アプリケーションのすべてのリソースが Astra によってインストールおよび検出されると、アプリケーションは Available 状態になります。



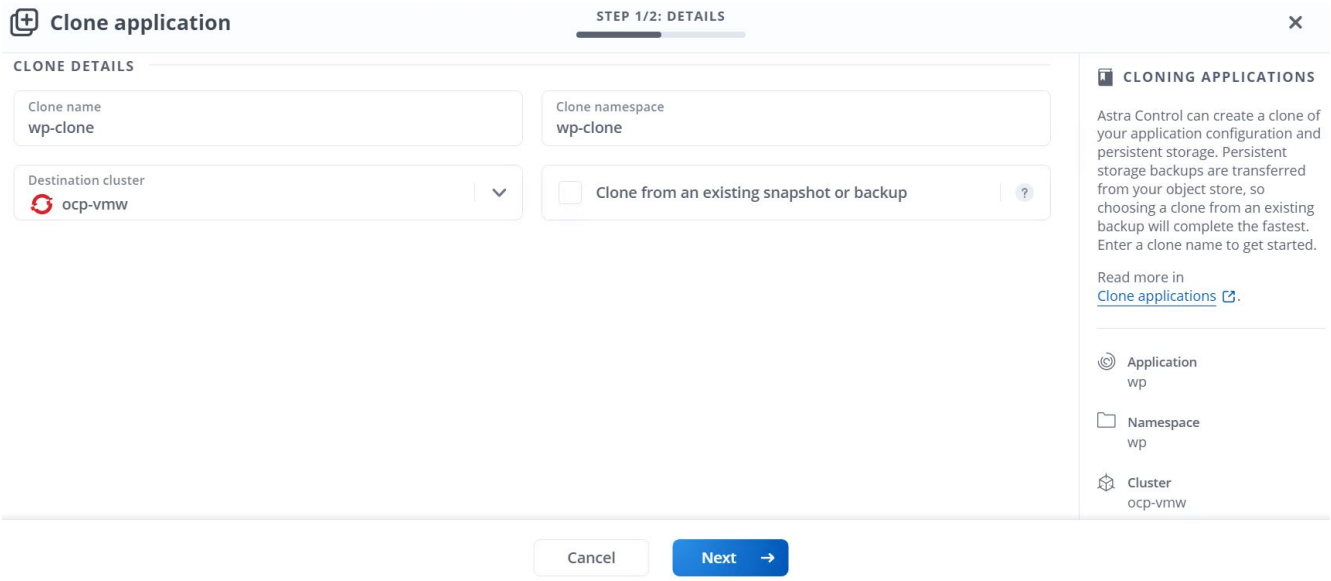
アプリケーションのクローニング

アプリケーションは、開発 / テストやアプリケーションの保護およびディザスタリカバリ目的で、元のクラスタまたはリモートクラスタにクローニングできます。同じストレージバックエンドで同じクラスタ内にあるアプリケーションをクローニングする場合、 NetApp FlexClone テクノロジを使用します。 FlexClone テクノロジを使用すると、 PVC のクローンを瞬時に作成し、ストレージスペースを節約できます。

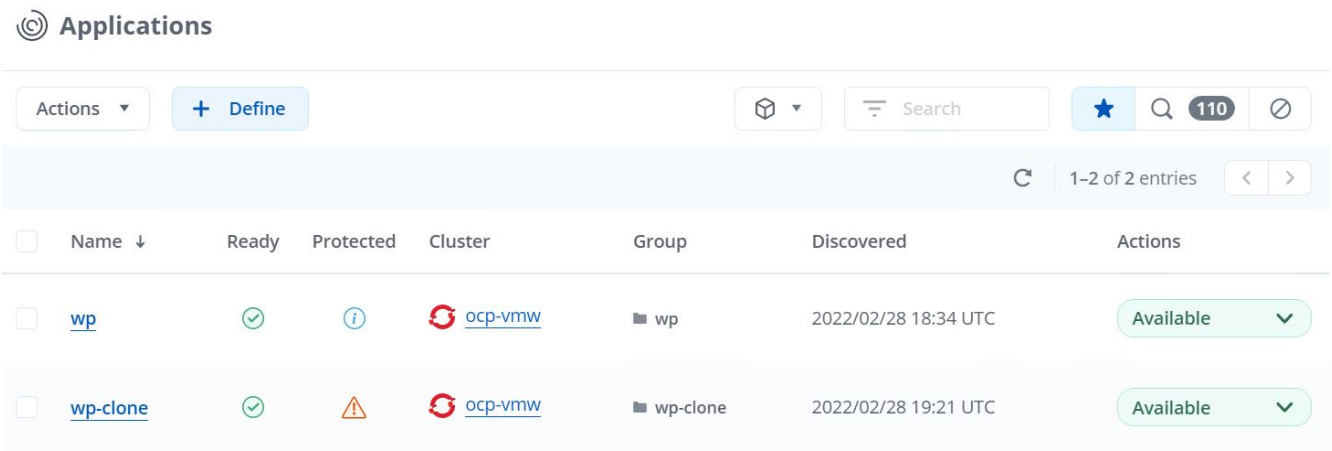
1. アプリケーションをクローンするには、 [アプリケーション（ Apps ）] > [管理（ Managed ）] タブに移動し、該当するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 Clone をクリックします。



2. 新しいネームスペースの詳細を入力し、クローニング先のクラスタを選択します。クローンを既存の Snapshot 、バックアップ、またはアプリケーションの現在の状態から作成するかどうかを選択します。詳細を確認したら、 [次へ] をクリックして、 [レビューペインに複製] をクリックします。



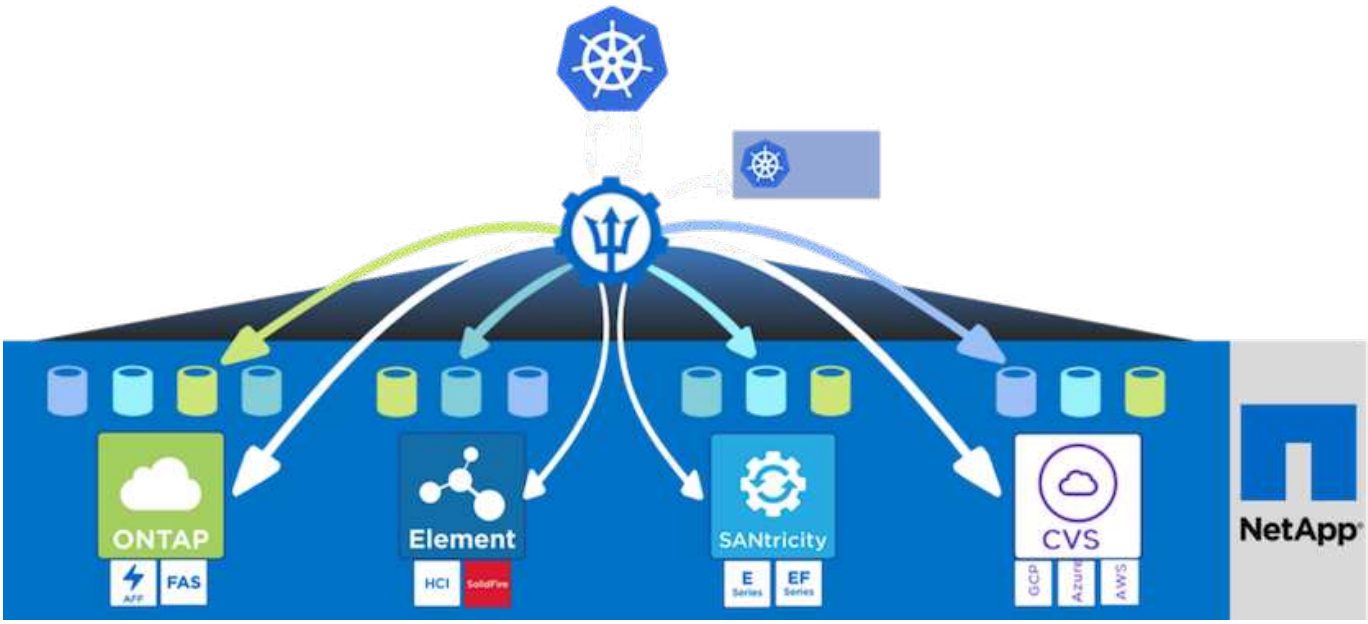
3. 新しいアプリケーションは Discovering 状態になり、 Astra Control Center は選択したクラスタにアプリケーションを作成します。アプリケーションのすべてのリソースが Astra によってインストールおよび検出されると、アプリケーションは Available 状態になります。



Astra Trident の概要

Astra Trident は、コンテナや Kubernetes ディストリビューション向けの、 Red Hat OpenShift などのオープンソースで完全にサポートされているストレージオーケストレーションツールです。Trident は、 NetApp ONTAP や Element ストレージシステムを含むネットアップストレージポートフォリオ全体と連携し、 NFS 接続と iSCSI 接続もサポートします。Trident を使用すると、ストレージ管理者の手を煩わせることなく、エンドユーザがネットアップストレージシステムからストレージをプロビジョニングして管理できるため、 DevOps ワークフローが高速化されます。

管理者は、プロジェクトのニーズやストレージシステムモデルに基づいて複数のストレージバックエンドを構成し、圧縮、特定のディスクタイプ、 QoS レベルなどの高度なストレージ機能を有効にして一定のレベルのパフォーマンスを保証できます。定義されたバックエンドは、プロジェクトの開発者が永続的ボリューム要求（ PVC ）を作成し、永続的ストレージをオンデマンドでコンテナに接続するために使用できます。



Astra Trident は、 Kubernetes と同様、迅速な開発サイクルを 1 年に 4 回リリースしています。

2022 年 1 月にリリースされた最新バージョンの Astra Trident は 22.01 です。Trident のどのバージョンがサポートされているかを確認できます Kubernetes ディストリビューションのテストに使用 "こちらをご覧ください"。

20.04 リリース以降、 Trident のセットアップは Trident オペレータによって実行されます。オペレータが大規模な導入を容易にし、 Trident インストールの一部として導入されたポッドの自己修復などの追加サポートを提供します。

21.01 リリースでは、 Trident Operator のインストールを容易にするために Helm チャートを使用できるようになりました。

Astra Trident をダウンロード

導入したユーザクラスタに Trident をインストールし、永続ボリュームをプロビジョニングするには、次の手順を実行します。

1. インストールアーカイブを管理ワークステーションにダウンロードし、内容を展開します。Trident の最新バージョンは 22.01 で、ダウンロードできます "こちらをご覧ください"。

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
```

```
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-22.01.0.tar.gz'

100%[===================================================================
==============================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]
```

2. ダウンロードしたバンドルから Trident のインストールを解凍します。

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

Helm を使用して Trident Operator をインストールします

1. Trident にはこのファイルを渡すオプションがないため、まず、ユーザクラスタの「 kubeconfig 」ファイルの場所を環境変数として設定します。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. Helm コマンドを実行し、ユーザークラスタに trident 名前空間を作成しながら、 Helball ディレクトリ内の tarball から Trident 演算子をインストールします。

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. Trident が正しくインストールされていることを確認するには、ネームスペースで実行されているポッドを確認するか、 tridentctl バイナリを使用してインストールされているバージョンを確認します。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

場合によっては、お客様の環境で Trident の導入のカスタマイズが必要になることもあります。このような場合は、 Trident のオペレータを手動でインストールし、含まれているマニフェストを更新して配置をカスタマイズすることもできます。

Trident Operator を手動でインストールします

1. Trident にはこのファイルを渡すオプションがないため、まず、ユーザクラスタの「 kubeconfig 」ファイルの場所を環境変数として設定します。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. 'trident-installer' ディレクトリには ' 必要なすべてのリソースを定義するマニフェストが含まれています適切なマニフェストを使用して、「 TridentOrchestrator 」カスタムリソース定義を作成します。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

3. 存在しない場合は、指定されたマニフェストを使用して、クラスタ内に Trident ネームスペースを作成します。

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. トライデントオペレータの配備に必要なリソースを作成しますたとえば ' オペレータ用のサービスアカウ

ント 'ClusterRole' および 'ClusterRoleBind' を 'ServiceAccount' 専用の 'PodSecurityPolicy' またはオペレータ自体に割り当てます

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.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
```

5. 次のコマンドを使用すると、展開後にオペレータのステータスを確認できます。

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. オペレータが導入したら、 Trident をインストールできます。これには 'TridentOrchestrator を作成する必要があります

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
```

```
    Operation:    Update
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportimage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentimage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport image:          netapp/trident-autosupport:21.01
    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:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                        Message
  ----    ------      ----  ----                        -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. Trident が正しくインストールされていることを確認するには、ネームスペースで実行されているポッドを確認するか、 tridentctl バイナリを使用してインストールされているバージョンを確認します。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0          | 22.01.0          |
+----------------+----------------+
```

ワーカーノードをストレージ用に準備する

NFS

ほとんどの Kubernetes ディストリビューションには、 Red Hat OpenShift など、デフォルトでインストールされる NFS バックエンドをマウントするパッケージとユーティリティが付属しています。

ただし NFSv3 については、クライアントとサーバ間の同時処理をネゴシエートするメカニズムはありませ

ん。したがって ' サーバが接続のウィンドウ・サイズを小さくしなくても NFS 接続の最適なパフォーマンスを確保できるように ' クライアント側 sunrpc スロット・テーブル・エントリーの最大数は ' サーバ上でサポートされている値と手動で同期する必要があります

ONTAP でサポートされる sunrpcslot table エントリの最大数は 128 です。つまり、 ONTAP は、一度に 128 個の NFS 要求を同時に処理できます。ただし、 Red Hat CoreOS / Red Hat Enterprise Linux では、接続ごとに最大 65 、 536 の sunrpc スロットテーブルエントリがデフォルトでサポートされます。この値を 128 に設定する必要があります。これは OpenShift のマシン構成オペレータ（ MCO ）を使用して実行できます。

OpenShift ワーカーノードで最大 sunrpc スロットテーブルエントリを変更するには、次の手順を実行します。

1. OCP Web コンソールにログインし、 [計算]>[マシン構成] に移動します。[マシン構成の作成] をクリックします。YAML ファイルをコピーして貼り付け、 [作成] をクリックします。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8; base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. MCO が作成されたら、すべてのワーカーノードに設定を適用し、 1 つずつ再起動する必要があります。プロセス全体には約 20~30 分かかります。「 OC GET MCP 」を使用してマシン構成が適用されているかどうかを確認し、ワーカーのマシン構成プールが更新されていることを確認します。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

iSCSI プロトコルによるブロックストレージボリュームのマッピングを許可するようにワーカーノードを準備するには、その機能をサポートするために必要なパッケージをインストールする必要があります。

Red Hat OpenShift では、 MCO （マシン構成オペレータ）を展開後にクラスタに適用することによって処理されます。

ワーカーノードで iSCSI サービスを実行するように設定するには、次の手順を実行します。

1. OCP Web コンソールにログインし、 [計算]>[マシン構成] に移動します。[マシン構成の作成] をクリックします。YAML ファイルをコピーして貼り付け、 [作成] をクリックします。

   マルチパスを使用しない場合：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-element-iscsi
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
  osImageURL: ""
```

   マルチパスを使用する場合：

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8; base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 構成の作成後、約 20~30 分で設定がワーカーノードに適用され、再ロードされます。「 OC GET MCP 」を使用してマシン構成が適用されているかどうかを確認し、ワーカーのマシン構成プールが更新されていることを確認します。ワーカーノードにログインして、 iscsid サービスが実行されている（マルチパスを使用している場合、 multipathd サービスが実行されている）ことを確認することもできます。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                     UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

また、適切なフラグを指定して「 OC debug 」コマンドを実行することにより、MachineConfig が正常に適用され、サービスが正常に開始されたことを確認することもできます。

ストレージシステムバックエンドを作成

Astra Trident Operator のインストールが完了したら、使用するネットアップストレージプラットフォームに合わせてバックエンドを設定する必要があります。Astra Trident のセットアップと設定を続行するには、次のリンクを参照してください。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI の略"
- "NetApp Element iSCSI の略"

NetApp ONTAP の NFS 構成

Trident を NetApp ONTAP ストレージシステムと統合するには、ストレージシステムとの通信を可能にするバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブのサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。NFS を提供している NetApp ONTAP システムの場合は、「 backend-ontap/nas.json 」ファイルを作業ディレクトリにコピーし、ファイルを編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. backendName 、 managementLIF 、 dataLIF 、 SVM 、ユーザ名を編集します。 パスワードの値を入力します。

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> カスタムの backendName 値は、簡単に識別できるように NFS を提供するストレージ DriverName とデータ LIF を組み合わせて定義することを推奨します。

3. このバックエンドファイルを設定した状態で、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. バックエンドを作成したら、次にストレージクラスを作成する必要があります。バックエンドと同様に、sample_inputs フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成したバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

> このファイルに定義されているオプションのフィールド「 fsType 」があります。この行は NFS バックエンドで削除できます。

6. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. ストレージクラスを作成したら、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. このファイルに対して行う必要がある唯一の編集は ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義をさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI 構成

Trident を NetApp ONTAP ストレージシステムと統合するには、ストレージシステムとの通信を可能にするバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブのサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。iSCSI を提供している NetApp ONTAP システムの場合は、「 backend-ontap -san.json 」ファイルを作業ディレクトリにコピーし、ファイルを編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. このファイルで管理 LIF 、データ LIF 、 SVM 、ユーザ名、パスワードの値を編集します。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. このバックエンドファイルを設定した状態で、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. バックエンドを作成したら、次にストレージクラスを作成する必要があります。バックエンドと同様に、sample_inputs フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成したバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

このファイルに定義されているオプションのフィールド「 fsType 」があります。iSCSI バックエンドでは、この値を特定の Linux ファイルシステムタイプ（ XFS 、 ext4 など）に設定することも、 OpenShift が使用するファイルシステムを決定できるようにするために削除することもできます。

6. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. ストレージクラスを作成したら、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. このファイルに対して行う必要がある唯一の編集は ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義をさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 構成

Trident を NetApp Element ストレージシステムと統合するには、 iSCSI プロトコルを使用してストレージシステムと通信できるバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブのサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。iSCSI を提供している NetApp Element システムの場合、「 backend-solidfire.json 」ファイルを作業ディレクトリにコピーし、ファイルを編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. エンドポイント行のユーザ、パスワード、および MVIP 値を編集します。

   b. 「仮想 IP 」の値を編集します。

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. このバックエンドファイルを設定したら、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                  UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. バックエンドを作成したら、次にストレージクラスを作成する必要があります。バックエンドと同様に、sample_inputs フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成したバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

> このファイルに定義されているオプションのフィールド「 fsType 」があります。iSCSI バックエンドでは、この値を特定の Linux ファイルシステムタイプ（ XFS 、 ext4 など）に設定するか、 OpenShift で使用するファイルシステムを決定できるようにするためにこの値を削除できます。

5. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. ストレージクラスを作成したら、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. このファイルに対して行う必要がある唯一の編集は ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義をさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```