Migrate Calico Cloud from Regular to Helm-Based Install/Upgrade (and using a private registry for helm install)

Platform: AKS

K8s version: 1.28.9

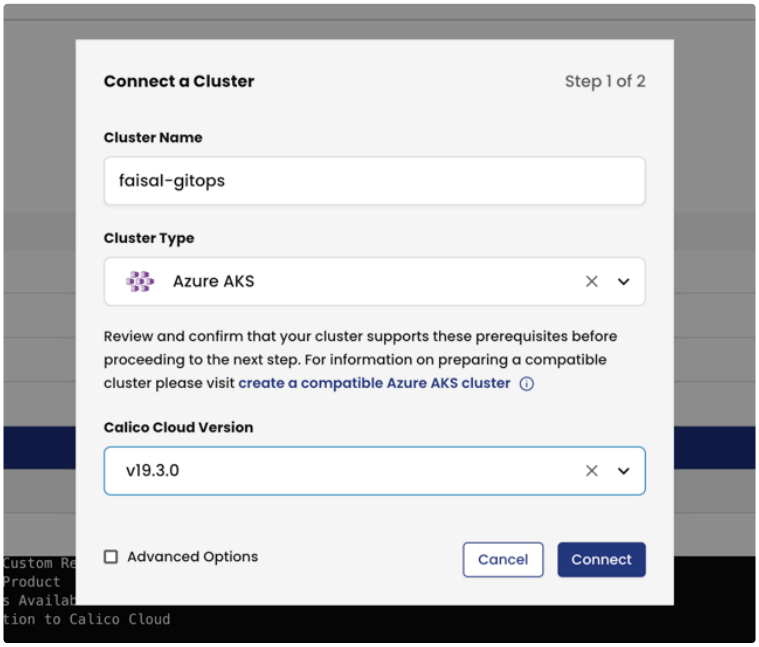
Calico Cloud: 19.3.0

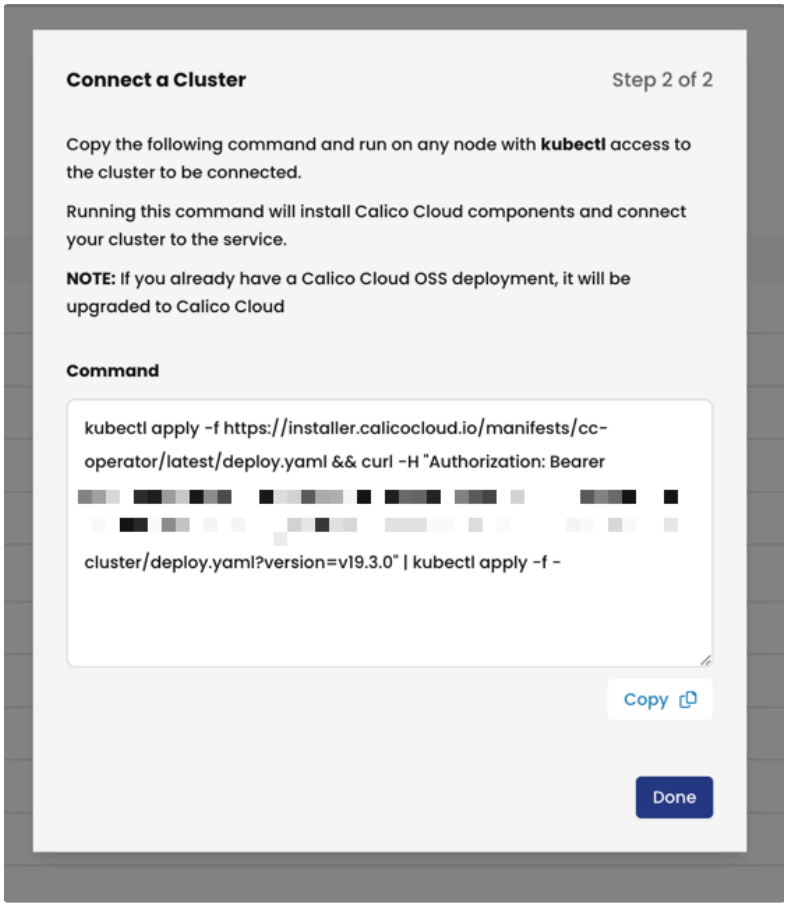
This guide will help you switch your Kubernetes cluster from a **kubectl** installation to a **Helm** installation using a private registry on a AKS cluster. You will retain the same cluster name and avoid switching to Calico OS.

Prerequisites

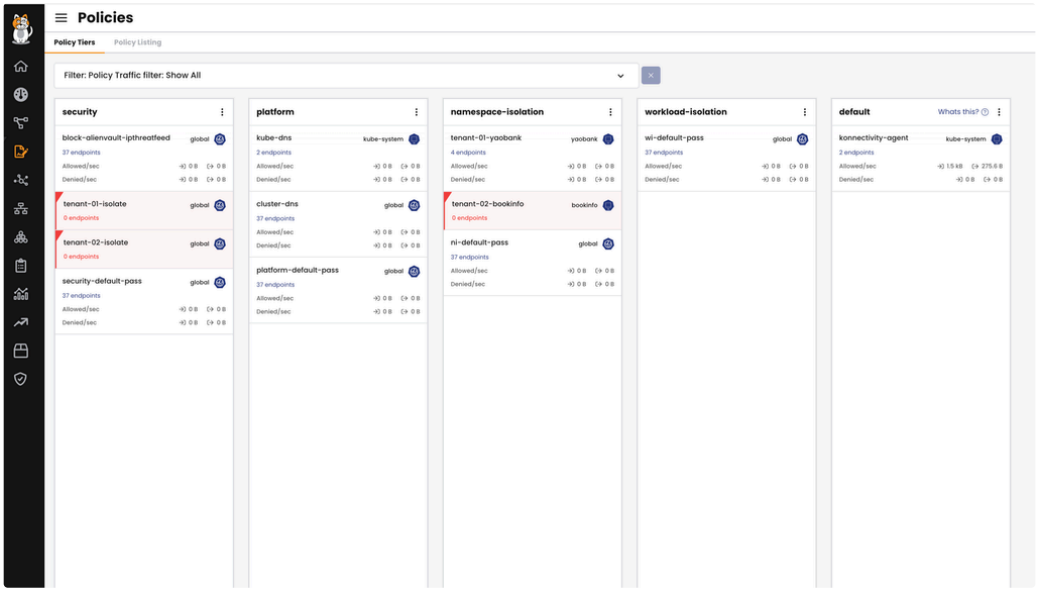
- **Access to the Managed Cluster**: Ensure you have console access.
- **Calico Cloud UI Access**: You'll need to interact with the Calico Cloud web interface.
- **Private Registry**: Set up with Calico Cloud images compatible with your cluster.
- `clean_kubectl_install.sh` **Script**: Obtain this script to clean up the existing installation.
- **Credentials for Private Registry**: If your registry requires authentication.

We will first start with installing CC on an AKS cluster with Azure as the CNI using the regular method where we copy the installation script from the Calico Cloud console and run in the cluster to connect it to the Calico Cloud.

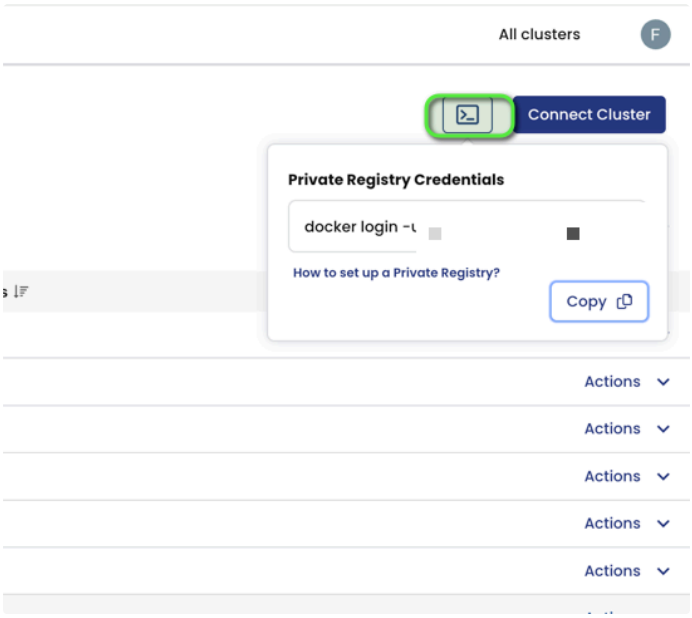




Once we have the cluster connected to Calico cloud, we will then deploy some tiers and network policies in the cluster to secure the cluster and apps.

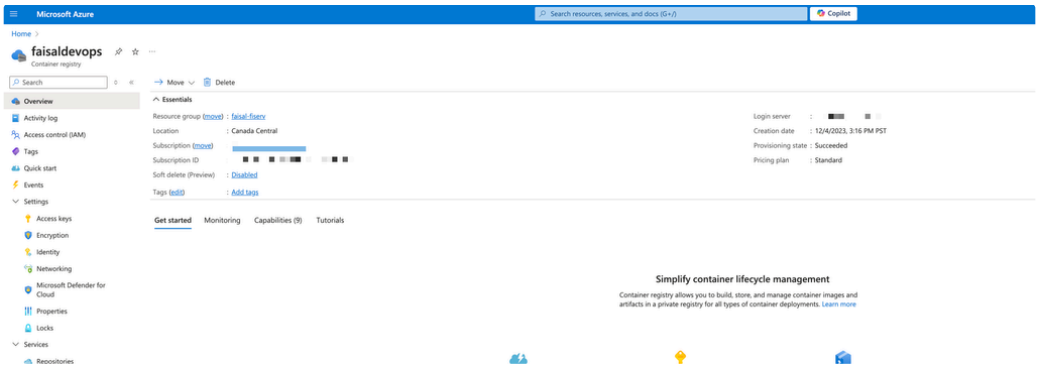


Set up registry credentials (Authenticate to the Source Registry)




```
faisalnaseem@MacBookPro calico-cloud-private-repo-install % docker login -u tigera+calicocloud_images -p 6IVTR8H8KB2HQNTVF quay.io
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
```

Now let's setup our private repo (in this case I'll go with an ACR repo):



Login to Azure:

```
az login
```

Set Variables (in this case I already have an acr repo named - `faisaldevops.azurecr.io`) :

```
RESOURCE_GROUP=faisal-fiserv
LOCATION=canadacentral
ACR_NAME=faisaldevops.azurecr.io
```

Authenticate to Your ACR

```
az acr login --name $ACR_NAME
```

Set ACR Login Server Variable:

```
ACR_LOGIN_SERVER=$(az acr show --name $ACR_NAME --query loginServer --output tsv)
```

Set Up Variables for Image Copying

Define Registry and Image Path:

```
REGISTRY="${ACR_LOGIN_SERVER}"
IMAGEPATH="" # Leave empty or set a custom path (e.g., "calico/")
```

Add Calico Cloud Helm Repository:

```
helm repo add calico-cloud https://installer.calicocloud.io/charts
helm repo update
```

Create a List of Required Images:

```
INSTALLER_IMAGE="quay.io/tigera/cc-operator:$(helm show chart calico-cloud/calico-cloud | grep version: | sed -e 's/version: *//' -e 's/+/-g/')"

IMAGES=(
  $INSTALLER_IMAGE
  quay.io/tigera/operator:v1.35.1
  quay.io/tigera/cnx-apiserver:v3.20.0-1.0
  quay.io/tigera/compliance-benchmarker:v3.20.0-1.0
  quay.io/tigera/compliance-controller:v3.20.0-1.0
  quay.io/tigera/compliance-reporter:v3.20.0-1.0
  quay.io/tigera/compliance-snapshotter:v3.20.0-1.0
  quay.io/tigera/key-cert-provisioner:v3.20.0-1.0
  quay.io/tigera/deep-packet-inspection:v3.20.0-1.0
  quay.io/tigera/fluentd:v3.20.0-1.0
  quay.io/tigera/fluentd-windows:v3.20.0-1.0
  quay.io/tigera/guardian:v3.20.0-1.5
  quay.io/tigera/intrusion-detection-controller:v3.20.0-1.0
  quay.io/tigera/webhooks-processor:v3.20.0-1.0
  quay.io/tigera/packetcapture:v3.20.0-1.0
  quay.io/tigera/policy-recommendation:v3.20.0-1.0
  quay.io/tigera/egress-gateway:v3.20.0-1.0
  quay.io/tigera/l7-collector:v3.20.0-1.0
  quay.io/tigera/envoy:v3.20.0-1.0
  quay.io/tigera/prometheus:v3.20.0-1.0
  quay.io/tigera/prometheus-service:v3.20.0-1.0
  quay.io/tigera/alertmanager:v3.20.0-1.0
  quay.io/tigera/cnx-queryserver:v3.20.0-1.0
  quay.io/tigera/kube-controllers:v3.20.0-1.0
  quay.io/tigera/cnx-node:v3.20.0-1.0
  quay.io/tigera/cnx-node-windows:v3.20.0-1.0
  quay.io/tigera/typha:v3.20.0-1.0
  quay.io/tigera/cni:v3.20.0-1.0
  quay.io/tigera/cni-windows:v3.20.0-1.0
  quay.io/tigera/es-gateway:v3.20.0-1.0
  quay.io/tigera/linseed:v3.20.0-1.0
  quay.io/tigera/dikastes:v3.20.0-1.0
  quay.io/tigera/pod2daemon-flexvol:v3.20.0-1.0
  quay.io/tigera/csi:v3.20.0-1.0
  quay.io/tigera/node-driver-registrar:v3.20.0-1.0
```

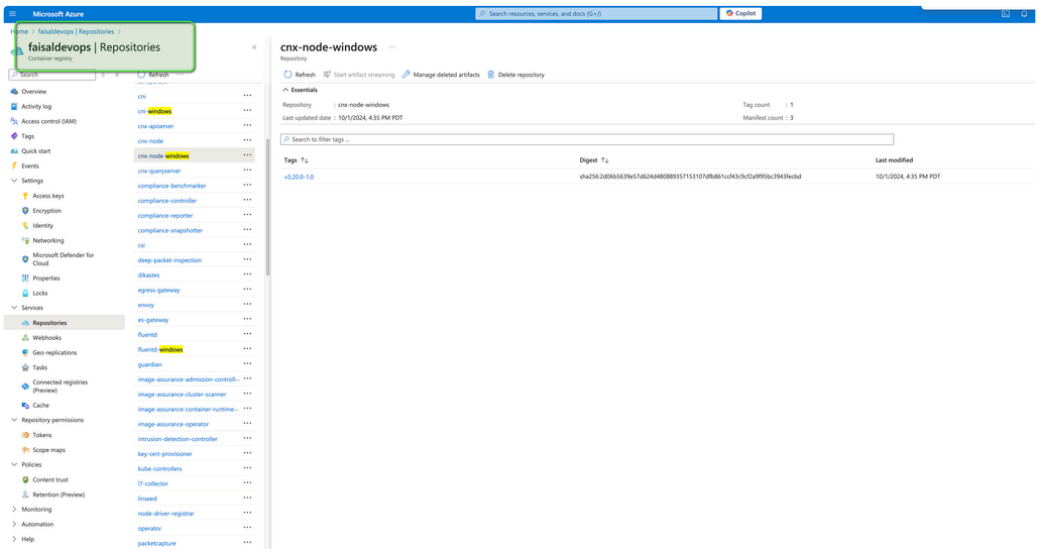
```
  quay.io/tigera/image-assurance-admission-controller:v1.20.3
  quay.io/tigera/image-assurance-operator:v1.20.3
  quay.io/tigera/image-assurance-container-runtime-adaptor:v1.20.3
  quay.io/tigera/image-assurance-cluster-scanner:v1.20.3
  quay.io/tigera/runtime-security-operator:v1.21.0
  quay.io/tigera/skimble:v1.21.0
  quay.io/tigera/cc-core:v0.2.7
  quay.io/tigera/prometheus-operator:v3.20.0-1.0
  quay.io/tigera/prometheus-config-reloader:v3.20.0-1.0
  quay.io/tigera/cc-cni-config-scanner:v0.3
)

```

Copy images to your registry

For Calico Cloud to install images from your registry, copy the images from the standard registries into your own registry.

```
for image in ${IMAGES[@]}; do
  img_base=$(echo ${image} | sed "s#^.*/##")
  crane cp ${image} ${REGISTRY}${IMAGEPATH}/${img_base} || break
done
```

We can now see the images in the repo:



STEP 3: Clean the calico-cloud install

Run the `clean_kubectl_install.sh` Script

- **Download the Script**: Obtain `clean_kubectl_install.sh` from the official source or repository.
- **Access the Managed Cluster**: Open a terminal connected to your cluster.
- **Execute the Script**:

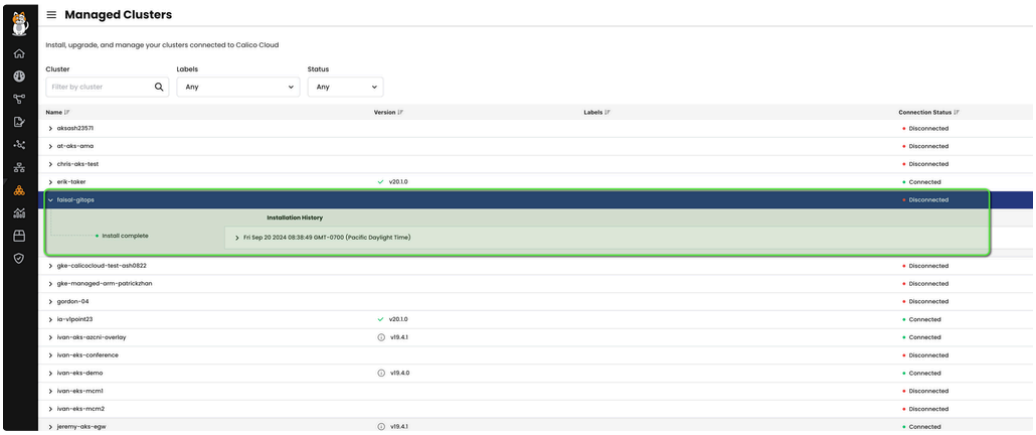
```
chmod +x clean_kubectl_install.sh
./clean_kubectl_install.sh
```

contents of `clean_kubectl_install.sh`

```
#!/bin/bash

kubectl delete crd installers.operator.calicocloud.io

kubectl delete -n calico-cloud deployments calico-cloud-controller-manager

kubectl delete secrets -n calico-cloud api-key

kubectl delete clusterrole calico-cloud-installer-role calico-cloud-installer-tigera-operator-role calico-cloud-installer-sa-creator-role

kubectl delete clusterrolebinding calico-cloud-installer-crb

kubectl delete role -n calico-cloud calico-cloud-installer-ns-role
kubectl delete role -n kube-system calico-cloud-installer-kube-system-role
kubectl delete role -n tigera-prometheus calico-cloud-installer-tigera-prometheus-role
kubectl delete role -n tigera-image-assurance calico-cloud-installer-tigera-image-assurance-role
kubectl delete role -n calico-system calico-cloud-installer-calico-system-role
kubectl delete role -n tigera-risk-system calico-cloud-installer-tigera-risk-system-role
kubectl delete role -n tigera-runtime-security calico-cloud-installer-tigera-runtime-security-role
kubectl delete rolebinding -n calico-cloud calico-cloud-installer-ns-rbac
kubectl delete rolebinding -n kube-system calico-cloud-installer-kube-system-rbac
kubectl delete rolebinding -n tigera-operator calico-cloud-installer-tigera-operator-rbac
kubectl delete rolebinding -n tigera-operator-cloud calico-cloud-installer-tigera-operator-rbac
kubectl delete rolebinding -n tigera-prometheus calico-cloud-installer-tigera-prometheus-rbac
kubectl delete rolebinding -n tigera-image-assurance calico-cloud-installer-tigera-image-assurance-rbac
kubectl delete rolebinding -n tigera-license calico-cloud-installer-tigera-license-rbac
kubectl delete rolebinding -n tigera-access calico-cloud-installer-tigera-access-rbac
kubectl delete rolebinding -n calico-system calico-cloud-installer-calico-system-rbac
kubectl delete rolebinding -n tigera-risk-system calico-cloud-installer-tigera-risk-system-rbac
kubectl delete rolebinding -n tigera-runtime-security calico-cloud-installer-tigera-runtime-security-rbac

# This disconnects the cluster from the Calico Cloud service.
# This is done so that then the Managed Cluster can be deleted in the UI.
kubectl delete managementclusterconnection tigera-secure

```

3. Confirm Cluster Disconnection

- **Verify in Calico Cloud UI**: The cluster should no longer appear in the interface.
- **Logs**: The cluster will stop sending logs to Calico Cloud.



4. Delete the Managed Cluster in Calico Cloud UI

- **Log In to Calico Cloud**: Access your Calico Cloud account.
- **Navigate to Managed Clusters**: Find your list of clusters.
- **Note the Cluster Name**: Write down the exact name of the cluster you're about to delete.
- **Delete the Cluster**: Remove it from the UI to prepare for re-registration.

```
faisalnaseem@MacBookPro calico-cloud-private-repo-install % k get tigerastatus
NAME                   AVAILABLE   PROGRESSING   DEGRADED   SINCE
apiserver              True        False         False      3h7m
calico                 True        False         False      3h8m
cloud-core             True        False         False      11d
compliance             False       False         True       3m30s
image-assurance        True        False         False      3h7m
intrusion-detection    False       False         True       50s
ippools                True        False         False      11d
log-collector          False       False         True       55s
monitor                True        False         False      11d
policy-recommendation  False       False         True       3m30s
tiers                  True        False         False      11d
```

ReInstall Calico-Cloud back with Helm (and using the private repo):

Step 1: Create Namespace and Image Pull Secret (If Required)

If your private registry requires authentication, you need to create an image pull secret in the `calico-cloud` namespace.

1. Check the Namespace:

```
kubectl get namespace
```

Create Image Pull Secret:

- Retrieve ACR Credentials:

```
ACR_NAME=faisaldevops.azurecr.io
ACR_USERNAME=$(az acr credential show -n $ACR_NAME --query username -o tsv)
ACR_PASSWORD=$(az acr credential show -n $ACR_NAME --query "passwords[0].value" -o tsv)
```

Create Secret:

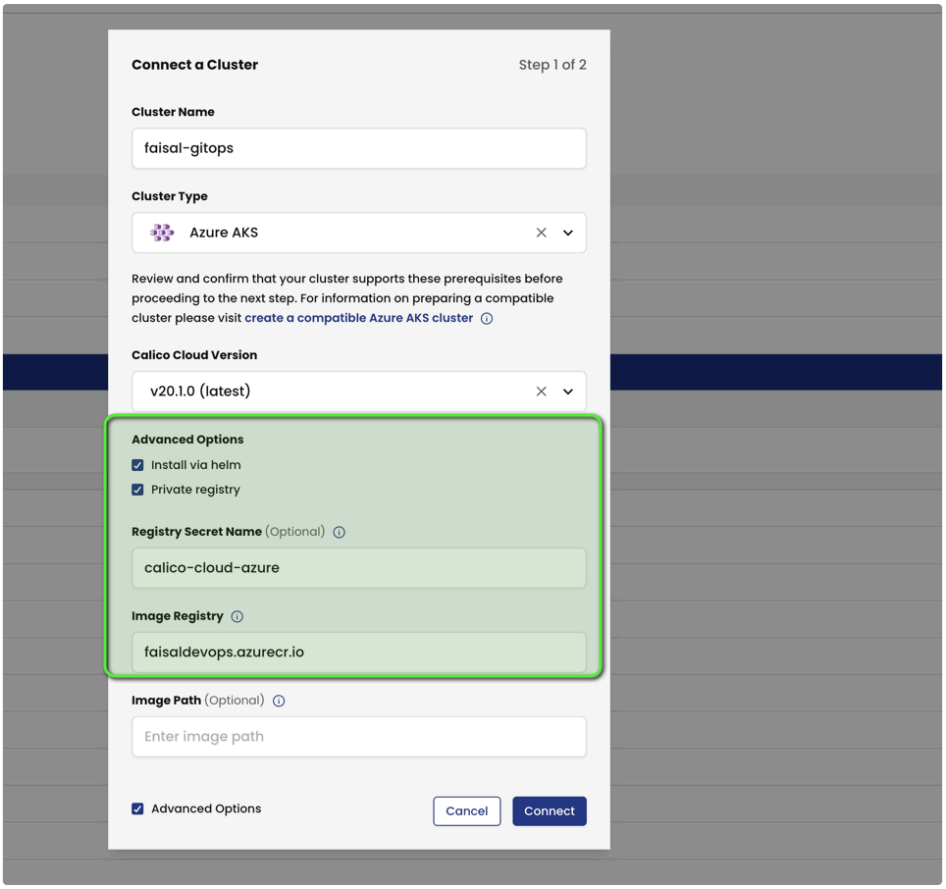
```
kubectl create secret docker-registry calico-cloud-azure \
  --docker-server=faisaldevops.azurecr.io \
  --docker-username=$ACR_USERNAME \
  --docker-password=$ACR_PASSWORD \
  -n calico-cloud
```

Step 4: Access the Calico Cloud Manager UI

1. **Navigate to Managed Clusters:**
   - Log in to the Calico Cloud Manager UI.
   - Go to the **Managed Clusters** page.
2. **Initiate Cluster Connection:**
   - Click on **Connect Cluster**.

Step 5: Generate Helm Installation Command

1. **Enter Cluster Details:**
   - **Cluster Name:** Provide a name for your cluster (e.g., `faisal-gitops`).
   - **Cluster Type:** Select the appropriate cluster type (e.g., **EKS**, **AKS**, **GKE**, or **Other**).
2. **Select Calico Cloud Version (Optional):**
   - If you need to install a specific older release, select it from the dropdown.
   - By default, the latest version is selected, and it's recommended to use it.
3. **Access Advanced Options:**
   - Click on **Advanced Options**.
   - Check the boxes for:
     - **Install via helm**
     - **Private registry**
4. **Provide Private Registry Details:**
   - **Registry Secret Name:** Enter the name of the image pull secret you created earlier (e.g., `calico-cloud-azure`).
   - **Image Registry:** Enter your ACR login server (e.g., faisaldevops.azurecr.io).
   - **Image Path:** Enter the image path if you used one when copying images (e.g., `calico/`). If you didn't set an image path, leave this blank.
5. **Configure Additional Features (Optional):**
   - Under **Advanced Options**, you can enable or disable specific Calico Cloud features during installation.
   - Features you can configure:
     - **Image Assurance**
     - **Container Threat Detection**
     - **Security Posture Dashboard**
     - **Packet Capture**
     - **Compliance Reports**
6. **Generate Helm Command:**
   - After filling in the details, click **Connect**.
   - A unique Helm installation command will be generated.




```
helm repo add calico-cloud https://installer.calicocloud.io/charts --force-update && \
helm upgrade --install calico-cloud-crds calico-cloud/calico-cloud-crds \
  --namespace calico-cloud --create-namespace && \
helm upgrade --install calico-cloud calico-cloud/calico-cloud \
  --namespace calico-cloud \
  --set apiKey=lq4mgljpt:tkskiye8r:4yq2uo47dxkafmlvj \
  --set installer.clusterName=faisal-gitops \
  --set installer.calicoCloudVersion=v20.1.0 \
  --set installer.imageRegistry=faisaldevops.azurecr.io \
  --set installer.imagePath="" \
  --set installer.registrySecret=calico-cloud-azure
```

We should see the cluster reinstalling:

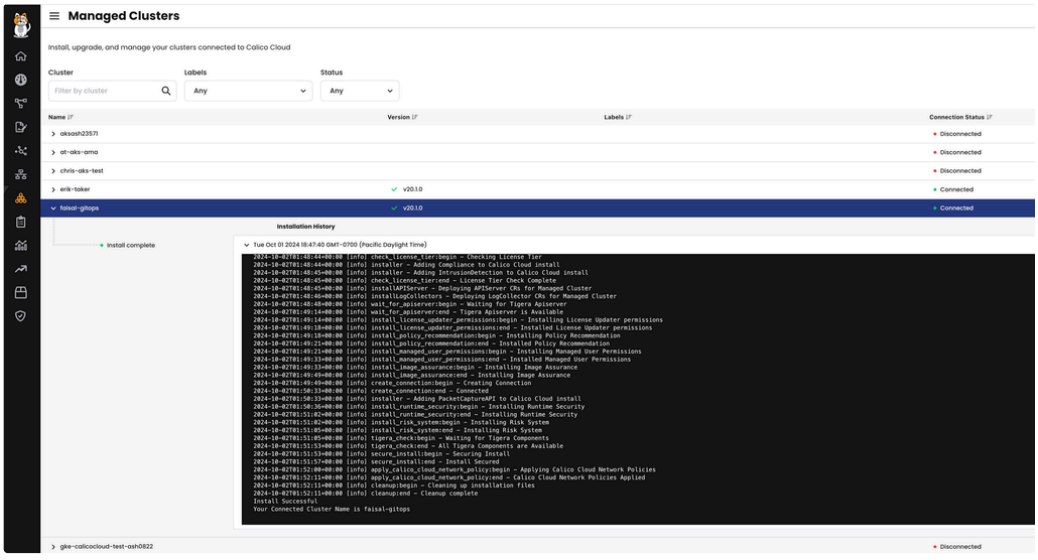
```
"calico-cloud" has been added to your repositories
Release "calico-cloud-crds" has been upgraded. Happy Helming!
NAME: calico-cloud-crds
LAST DEPLOYED: Tue Oct  1 18:46:58 2024
NAMESPACE: calico-cloud
STATUS: deployed
REVISION: 4
TEST SUITE: None
W1001 18:47:18.203408   90358 warnings.go:70] unknown field "spec.imageRegistry"
W1001 18:47:18.203463   90358 warnings.go:70] unknown field "spec.registrySecret"
Release "calico-cloud" has been upgraded. Happy Helming!
NAME: calico-cloud
LAST DEPLOYED: Tue Oct  1 18:47:03 2024
NAMESPACE: calico-cloud
STATUS: deployed
REVISION: 4
```

```
TEST SUITE: None
NOTES:
Thank you for installing Calico Cloud.

Track the status of your install with the following command:

    kubectl get installer default --namespace calico-cloud -o jsonpath --template '{.status}' -w
```

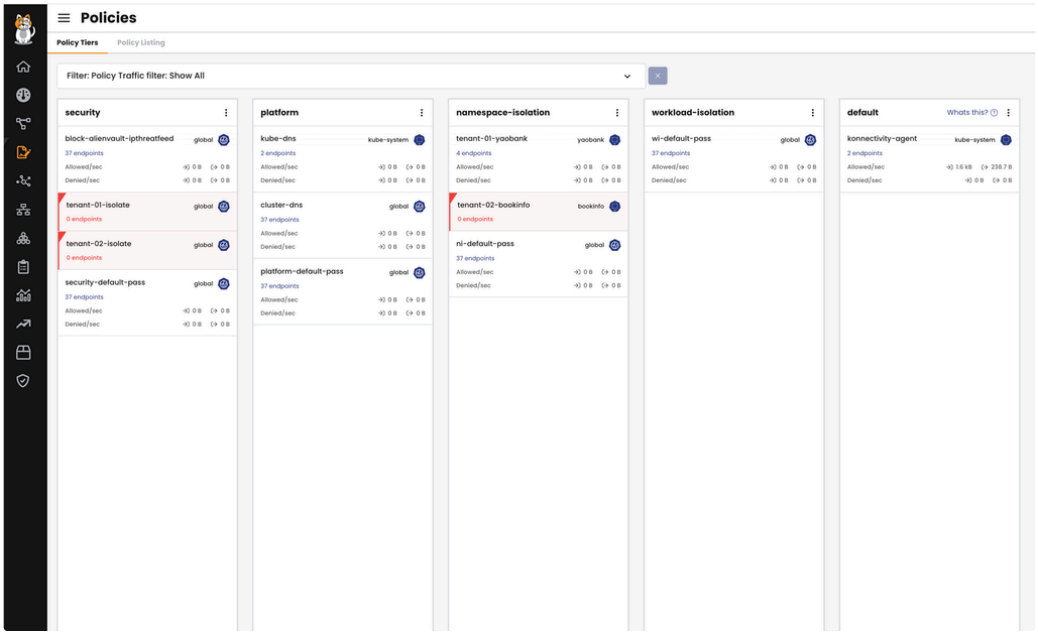
Monitor the progress of the calico-cloud components:

```
faisalnaseem@MacBookPro calico-cloud-private-repo-install % k get tigerastatus
NAME                              AVAILABLE   PROGRESSING   DEGRADED   SINCE
apiserver                         True        False         False      2m54s
calico                            True        False         False      4h
cloud-core                        True        False         False      11d
compliance                        True        False         False      109s
image-assurance                   True        False         False      108s
intrusion-detection               True        False         False      104s
ippools                           True        False         False      11d
log-collector                     True        False         False      14s
management-cluster-connection     True        False         False      94s
monitor                           True        False         False      2m19s
packet-capture                    True        False         False      29s
policy-recommendation             True        False         False      2m14s
tiers                             True        False         False      11d
```

The cluster can be seen connected again (with the same name as before) to the calico-cloud console.



We can also see all the previous network policies are still present in the network policy dashboard:



This completes the migration from a regular kubectl based installation to a Helm chart based installation using a private repository on an AKS cluster.