Setting up Container Registry for GKE private clusters

This page describes how to configure DNS entries for using Container Registry with a Google Kubernetes Engine private cluster and VPC Service Controls. These steps are only required if your GKE private cluster uses Container Registry.

Note: While private clusters in service perimeters can use Container Registry without the restricted VIP, you should not do so. If the restricted VIP is not used, data can be exfiltrated from a supported service to an unsupported one.

Before you begin

Before you create a service perimeter, set up a new private cluster (https://cloud.google.com/kubernetes-engine/docs/how-to/private-clusters) or identify the existing private clusters that you want to protect.

Also, you must permit egress to 199.36.153.4/30 on port 443. Normally, a VPC network has an implied rule that allows all egress traffic to any destination. However, if you have a rule denying such traffic, you must create an egress firewall rule to allow TCP traffic on port 443 to 199.36.153.4/30.

Configuring DNS

To support GKE private clusters that use Container Registry inside a service perimeter, you first need to configure your DNS server so requests to Container Registry addresses resolve to `restricted.googleapis.com`, the restricted VIP. You can do so using Cloud DNS private DNS zones.

1. Create a managed private zone.

```
gcloud beta dns managed-zones create ZONE_NAME \
    --visibility=private \
    --networks=https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global
    --description=DESCRIPTION \
    --dns-name=gcr.io \
    --project=PROJECT_ID
```

   Where:

   - ***ZONE_NAME*** is a name for the zone you are creating. For example, `gcr`. This name will be used in each of the following steps.
   - ***PROJECT_ID*** is the ID of the project that hosts your GKE private cluster.
   - ***NETWORK*** is the name of the cluster network that you want to redirect requests from. The default network is `default`.
   - ***DESCRIPTION*** is an optional, human-readable description of the managed zone.

2. Start a transaction.

```
gcloud dns record-sets transaction start \
  --zone=ZONE_NAME \
  --project=PROJECT_ID
```

   Where:

   - ***ZONE_NAME*** is the name of the zone you created in the first step.
   - ***PROJECT_ID*** is the ID of the project that hosts your GKE private cluster.

3. Add a CNAME record for `*.gcr.io`.

```
gcloud dns record-sets transaction add \
  --name=*.gcr.io. \
  --type=CNAME gcr.io. \
  --zone=ZONE_NAME \
  --ttl=300 \
  --project=PROJECT_ID
```

Where:

- ***ZONE_NAME*** is the name of the zone you created in the first step.
- ***PROJECT_ID*** is the ID of the project that hosts your GKE private cluster.

4. Add an A record for the restricted VIP.

```
gcloud dns record-sets transaction add \
  --name=gcr.io. \
  --type=A 199.36.153.4 199.36.153.5 199.36.153.6 199.36.153.7 \
  --zone=ZONE_NAME \
  --ttl=300 \
  --project=PROJECT_ID
```

Where:

- ***ZONE_NAME*** is the name of the zone you created in the first step.
- ***PROJECT_ID*** is the ID of the project that hosts your GKE private cluster.

5. Execute the transaction.

```
gcloud dns record-sets transaction execute \
  --zone=ZONE_NAME \
  --project=PROJECT_ID
```

Where:

- ***ZONE_NAME*** is the name of the zone you created in the first step.
- ***PROJECT_ID*** is the ID of the project that hosts your GKE private cluster.

Configuring the service perimeter

After configuring the DNS records (#configure-dns), either create a new service perimeter (https://cloud.google.com/vpc-service-controls/docs/create-service-perimeters) or update an existing perimeter (https://cloud.google.com/vpc-service-controls/docs/manage-service-perimeters), and then add the Container Registry service to the list of services you want to protect using the service perimeter.

Verifying the perimeter works

After configuring the service perimeter (#configure-perimeter), you can follow the PHP Guestbook tutorial (https://kubernetes.io/docs/tutorials/stateless-application/guestbook/) to verify that the perimeter is functioning as expected.

If the configuration is functioning correctly, the pod for the guestbook application web frontend cannot be started.

The following error message is an example of what should be returned if the perimeter is correctly configured:

```
Events:
  Type     Reason                 Age                From
  ----     ------                 ----               ----
  Normal   Scheduled              8m                 default-scheduler
  Normal   SuccessfulMountVolume  8m                 kubelet, gke-netpolicy-default-p
  Normal   Pulling                6m (x4 over 8m)    kubelet, gke-netpolicy-default-p
  Warning  Failed                 6m (x4 over 8m)    kubelet, gke-netpolicy-default-p
```


Last updated August 13, 2019.