We'll create a private cluster named gke-deep-dive that has private nodes, and has no client access to the public endpoint. As part of the same command, we'll also create

- A subnet named gke-deep-dive-subnet

Step 1: gcloud config set compute/zone us-west1-a

Step 2: **gcloud container clusters create** gke-deep-dive --num-nodes=1 --disk-type=pd-standard --disk-size=10 **--create-subnetwork name=gke-deep-dive-subnet --enable-ip-alias --enable-private-nodes --enable-private-endpoint --master-ipv4-cidr 172.16.0.32/28**

where:

- `--create-subnetwork name=gke-deep-dive-subnet` **causes GKE to automatically create a subnet named** `gke-deep-dive-subnet`**.**
- `--enable-ip-alias` **makes the cluster VPC-native**
- `--enable-private-nodes` **indicates that the cluster's nodes do not have external IP addresses.**
- `--enable-private-endpoint` **indicates that the cluster is managed using the private IP address of the control plane API endpoint.**
- `--master-ipv4-cidr 172.16.0.32/28` **specifies an internal IP address range for the control plane (optional for Autopilot).**

Step 4: Verify that the cluster's nodes do not have external IP addresses.

gcloud container clusters describe gke-deep-dive

Step 5: Check the new subnet that's created on console

At this point, these are the only IP address ranges that have access to the control plane:

- **The primary range of** `gke-deep-dive-subnet`**.**
- **The secondary range used for Pods.**

Let's try to access the control plane from outside `gke-deep-dive-subnet`**, using**

cloud shell. We must authorize the public IP address of our cloud shell instance to have access to the cluster endpoint.

Step 6: dig +short myip.opendns.com @resolver1.opendns.com

Step 7: gcloud container clusters describe gke-deep-dive --zone us-west1-a --format "flattened(masterAuthorizedNetworksConfig.cidrBlocks[])"

Step 8: gcloud container clusters update gke-deep-dive --zone us-west1-a --enable-master-authorized-networks --master-authorized-networks <IP address from step 6>

Step 9: Error because this cluster doesn't have public endpoint

Create another cluster without private endpoint, limited access to public endpoint

Step 10: gcloud container clusters create gke-deep-dive-public --num-nodes=1 --disk-type=pd-standard --disk-size=10 --enable-master-authorized-networks --subnetwork gke-deep-dive-subnet --enable-private-nodes --enable-ip-alias --master-ipv4-cidr 172.16.0.16/28

At this point, these are the only IP addresses that have access to the cluster control plane:

- **The primary range of `gke-deep-dive-subnet`.**
- **The secondary range used for Pods.**

Let's try to access the control plane from outside `gke-deep-dive-subnet`, using cloud shell. We must authorize the public IP address of our cloud shell instance to have access to the cluster endpoint.

Step 11: gcloud container clusters describe gke-deep-dive-public --zone us-west1-a --format "flattened(masterAuthorizedNetworksConfig.cidrBlocks[])"

Step 12: gcloud container clusters update gke-deep-dive-public --zone us-west1-a --enable-master-authorized-networks --master-authorized-networks <*IP address from step 6*>

Step 13: gcloud container clusters describe gke-deep-dive-public --zone us-west1-a --format "flattened(masterAuthorizedNetworksConfig.cidrBlocks[])"

Now these are the only IP addresses that have access to the control plane:

- **The primary range of** `gke-deep-dive-subnet`**.**
- **The secondary range used for Pods.**
- **Address ranges that we have authorized for the cloud shell public IP**

Step 14: gcloud container clusters get-credentials gke-deep-dive-public *--internal-ip*

Step 15: Kubectl get nodes

It might take sometime

Step 16: If connection error, try: gcloud container clusters update gke-deep-dive-public *--enable-master-global-access*

Step 17: Delete cluster: gcloud container clusters delete -q gke-deep-dive-public gke-deep-dive --zone us-west1-a