

[Google Kubernetes Engine \(GKE\)](https://cloud.google.com/kubernetes-engine/) (<https://cloud.google.com/kubernetes-engine/>)

[Documentation](https://cloud.google.com/kubernetes-engine/docs/) (<https://cloud.google.com/kubernetes-engine/docs/>) [Guides](#)

Using Google-managed SSL certificates

Beta

This product or feature is in a pre-release state and might change or have limited support. For more information, see the [product launch stages](https://cloud.google.com/products/#product-launch-stages) (<https://cloud.google.com/products/#product-launch-stages>).

Overview

In Google Kubernetes Engine (GKE), you can use [Ingresses](https://cloud.google.com/kubernetes-engine/docs/concepts/ingress) (<https://cloud.google.com/kubernetes-engine/docs/concepts/ingress>) to create [HTTPS load balancers](https://cloud.google.com/load-balancing/docs/https/) (<https://cloud.google.com/load-balancing/docs/https/>) with automatically configured SSL certificates. Google-managed SSL certificates are provisioned, renewed, and managed for your domain names. To learn how to create one, read [Working with Google-managed SSL certificates](https://cloud.google.com/load-balancing/docs/ssl-certificates#managed-certs) (<https://cloud.google.com/load-balancing/docs/ssl-certificates#managed-certs>).

Creating an Ingress with a managed certificate

Note: Managed certificates require clusters with masters running Kubernetes 1.12.6-gke.7 or higher.

To configure a managed SSL certificate and associate it with an Ingress, you need to:

- Create a ManagedCertificate object.
- Associate the ManagedCertificate object to an Ingress by adding an annotation `networking.gke.io/managed-certificates` to the Ingress. This annotation is a comma-separated list of ManagedCertificate resources, `cert1, cert2, cert3` for example.

The ManagedCertificate resource must be created in the same namespace as the Ingress.

Limitations

Google-managed certificates are less flexible than certificates you obtain and manage yourself. Managed certificates support a single, non-wildcard domain. Self-managed certificates can support wildcards and multiple subject alternative names (SANs).

(<https://tools.ietf.org/html/rfc5280#section-4.2.1.6>).

If you require self-managed certificates or if you already own SSL certificates that you would like to configure on your Ingress, refer to the [Ingress](#)

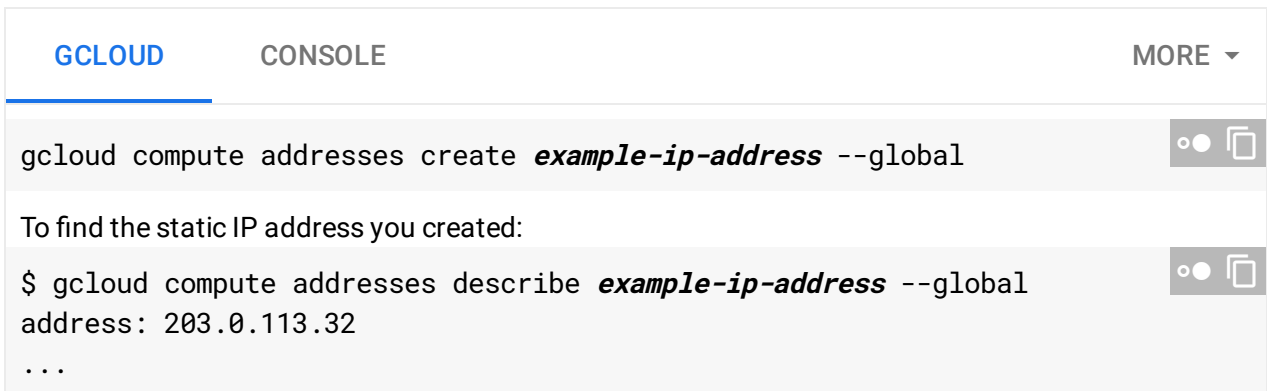
(https://cloud.google.com/kubernetes-engine/docs/concepts/ingress#setting_up_https_tls_between_client_and_load_balancer) documentation.

The number and type of certificates supported by an Ingress are defined by the [limits of Google Cloud managed SSL certificates](#)

(<https://cloud.google.com/load-balancing/docs/ssl-certificates#ssl-certificate-limits>).

Prerequisites

- You must own the domain name. The domain name must be no longer than 63 characters. You can use [Google Domains](#) (<https://domains.google.com/>) or another registrar.
- Create a [reserved \(static\) external IP address](#) (<https://cloud.google.com/compute/docs/ip-addresses/reserve-static-external-ip-address>). Reserving a static IP address guarantees that it will remain yours, even if you delete the Ingress. If you do not reserve an address it may change requiring you to reconfigure your domain's DNS records. Use `gcloud` command-line tool or the Cloud Console to create a reserved IP address, named `example-ip-address`:



```
G CLOUD    CONSOLE    MORE ▾  
  
gcloud compute addresses create example-ip-address --global  
  
To find the static IP address you created:  
$ gcloud compute addresses describe example-ip-address --global  
address: 203.0.113.32  
...
```

Setting up the managed certificate

1. Create a `ManagedCertificate` resource. This resource specifies the domain that the SSL certificate will be created for. Wildcard domains are not supported. The `spec.domains` list must contain only one domain.

Save the following example `ManagedCertificate` manifest to a file named `example-certificate.yaml`. Replace `myapp.example.com` with your domain name:

```
apiVersion: networking.gke.io/v1beta1
kind: ManagedCertificate
metadata:
  name: example-certificate
spec:
  domains:
  - myapp.example.com
```

Use `kubectl` to create the resource:

```
kubectl apply -f example-certificate.yaml
```

2. Create a `NodePort` Service to expose your application to the Internet. The following is an example Service manifest file, `example-service.yaml`.

```
apiVersion: v1
kind: Service
metadata:
  name: example-nodeport-service
spec:
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
```

This example specification doesn't select any Pods to include in the Service. This is sufficient to demonstrate how to configure Managed Certificates. In practical use though, a Service specification should include a selector.

Refer to the [NodePort documentation](#)

(https://cloud.google.com/kubernetes-engine/docs/concepts/service#service_of_type_nodeport) for details on how to configure the Service.

Use `kubectl` to create the Service:

```
kubectl apply -f example-service.yaml
```

3. Create an Ingress, linking it to the ManagedCertificate you created previously.

- Set the `networking.gke.io/managed-certificates` annotation to the name of your certificate.
- For the `spec.backend.serviceName` field, use the name of the service you created in the previous step.
- Set the `spec.backend.servicePort` field to the port you specified in your Service manifest.
- Set the `kubernetes.io/ingress.global-static-ip-name` annotation to the name of your reserved IP address.

The following is an example Ingress manifest, `example-ingress.yaml`:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: example-ip-address
    networking.gke.io/managed-certificates: example-certificate
spec:
  backend:
    serviceName: example-nodeport-service
    servicePort: 80
```

Use `kubectl` to create the Ingress:

```
kubectl apply -f example-ingress.yaml
```

★ **Note:** It may take up to 10-20 minutes for the load balancer to begin functioning.

4. Look up the IP address of the load balancer created in the previous step. Use the following command to get the IP address of the load balancer:

```
$ kubectl get ingress
NAME           HOSTS    ADDRESS          PORTS    AGE
example-ingress *       203.0.113.32    80      54s
```

The load balancer's IP address is listed in the **ADDRESS** column, **203.0.113.32** in this example. If you are using a reserved static IP address that will be the load balancer's address.

If the address is not listed, wait for the Ingress to finish setting up.

5. Configure the DNS records for your domain to point to the IP address of the load balancer. In this example create an A record to point `myapp.example.com` to `203.0.113.32`. If you use Cloud DNS, you can refer to the [Managing Records](https://cloud.google.com/dns/records/) (<https://cloud.google.com/dns/records/>) guide for details.

★6. **Note:** You must wait for the DNS records you configured in the previous step to propagate before continuing.

Wait for the managed certificate to be provisioned. This may take up to 15 minutes. You can check on the status of the certificate with the following command:

```
kubectl describe managedcertificate
```

Once a certificate is successfully provisioned, the value of the `Status.CertificateStatus` field will be `Active`. The following example shows the output of `kubectl describe` after the example certificate is successfully provisioned:

```
Name:          example-certificate
Namespace:     default
Labels:        <none>
Annotations:   <none>
API Version:   networking.gke.io/v1beta1
Kind:          ManagedCertificate
(...)
Spec:
  Domains:
    example.com
Status:
  CertificateStatus: Active
(...)
```

7. Verify that SSL is working by visiting your domain using the `https://` prefix. In this example check `https://myapp.example.com`. Your browser will indicate that the connection is secure and you can view the certificate details.

Migrating to Google-managed certificates from self-managed certificates

When you migrate an Ingress from using self-managed SSL certificates to Google-managed SSL certificates, do not delete any self-managed SSL certificates before the Google-managed SSL certificates are active. After the Google-managed SSL certificates are successfully provisioned, they automatically become active. When the Google-managed SSL certificates are active, you can delete your self-managed SSL certificates.

Use these instructions for migrating from self-managed to Google-managed SSL certificates.

1. Add a new managed certificate to the Ingress, as described in the [Setting up the managed certificate](#) (#setting_up_the_managed_certificate) section.
2. Wait until the status of the Google-managed certificate resource is Active. Check the status of the certificate with `kubectl describe managedcertificate`.
3. When the status is Active, update the Ingress to remove the references to the self-managed certificate.

Removing a managed certificate

To remove a managed certificate from your cluster you must delete the ManagedCertificate resource and remove the Ingress annotation that references it.

1. Delete the ManagedCertificate resource with `kubectl`:

```
kubectl delete -f example-certificate.yaml
```

You will get the following output:

```
managedcertificate.networking.gke.io "example-certificate" deleted
```

2. Remove the annotation from the Ingress:

```
kubectl annotate ingress example-ingress networking.gke.io/managed-certificates-
```

Notice the minus sign, "-", at the end of the command.

3. Release the static IP address that you reserved for your load balancer:

GCLOUD CONSOLE CONFIG CONNECTOR

Using the `gcloud` command-line tool:

```
gcloud compute addresses delete [ADDRESS_NAME] --global
```

where `[ADDRESS_NAME]` is the name of the IP address.

Troubleshooting

ManagedCertificate definitions are validated before the ManagedCertificate resource is created. If validation fails the ManagedCertificate resource is not created and an error message is printed. The different failure reasons are explained below:

`spec.domains` in body should have at most 1 items

Your ManagedCertificate manifests lists more than one domain in the `spec.domains` field. Managed certificates support only one domain.

`spec.domains` in body should match `'^(([a-zA-Z0-9]+|[a-zA-Z0-9]([-a-zA-Z0-9]*[a-zA-Z0-9])\.)+[a-zA-Z]([-a-zA-Z0-9]*[a-zA-Z0-9])\.)?$',`

You specified an invalid domain name or a wildcard domain name in the `spec.domains` field. Managed certificates don't support wildcard domains like `*.example.com`

`spec.domains` in body should be at most 63 chars long

You specified a domain name that is too long. Managed certificates support domain names with at most 63 characters.

What's next

- Learn more about [HTTP\(S\) load balancers](https://cloud.google.com/load-balancing/docs/https/) (<https://cloud.google.com/load-balancing/docs/https/>) on Google Cloud.

- Learn how to use self-managed SSL certificates (<https://cloud.google.com/kubernetes-engine/docs/how-to/ingress-multi-ssl>) on GKE.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated January 6, 2020.