Services and Networking

This chapter contains the following topics:

Load Balancing Kubernetes Services using NGINX

Cisco Container Platform uses NGINX to offer advanced layer 7 load balancing solutions. NGINX can handle a large number of requests and at the same time, it can be run on Kubernetes containers.

The NGINX load balancer is automatically provisioned as part of Kubernetes cluster creation. Each Kubernetes cluster is provisioned with a single L7 NGINX load balancer. You can access the load balancer using its virtual IP address, which can be found by running the command `kubectl get svc -n ccp`.

To use the NGINX load balancer, you must create an Ingress resource. Ingress is a Kubernetes object that allows you to define HTTP load balancing rules to allow inbound connections to reach the cluster services. You can configure Ingress to create external URLs for services, load balance traffic, terminate SSL, offer name-based virtual hosting, and so on.

L7 Ingress

Cisco Container Platform supports the following types of L7 Ingresses:

- **Simple fanout**

  It enables you to access the website using http.

  Example

  ```
  cafe.test.com ->   10.1.1.1   ->   /tea      tea-svc:80
                                                /coffee   coffee-svc:80
  ```

  For this type of Ingress, you need to create a yaml file that defines the Ingress rules.

  Sample yaml file

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
name: cafe-ingress
spec:
rules:
-host: cafe.test.com
http:
    paths:
    -path:/
    backend:
    serviceName: tea-svc
    servicePort: 80
    -path:/
    backend:
    serviceName: tea-svc
    servicePort: 80
```

- **Simple fanout with SSL termination**

  It enables you to access the website using https.

  Example

```
https://cafe.test.com   ->   10.1.1.1   ->   /tea       tea-svc:80
                                              /coffee    coffee-svc:80
```

  For this type of Ingress, you need to create the following yaml files:

  - A yaml file that defines the Secret

    Sample yaml file

```
apiVersion: v1
kind: Secret
metadata:
  name: cafe-secret
type: Opaque
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
```

  - A yaml file that defines the Ingress rules

    Sample yaml file

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  tls:
  -hosts:
  -cafe.test.com
   secretName: cafe-secret
  rules:
  -host: cafe.example.com
  http:
   paths:
   -path:/
   backend:
     serviceName: tea-svc
     sevicePort: 80
   -path:/
   backend:
```

```
        serviceName: coffee-svc
        servicePort: 80
```

- **Name based virtual hosting**

  It enables you to access the website using multiple host names.

  Example

```
tea.test.com    --|           |-> tea.test.com      s1:80
                  |10.1.1.1   |
coffee.test.com --|           |-> coffee.test.com   s2:80
```

  For this type of Ingress, you need to create a yaml file that defines the Ingress rules.

  Sample yaml file

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata
name: cafe-ingress
spec:
 rules:
 -host: tea.test.com
 http:
    paths:
    -path:/
    backend:
    serviceName: tea-svc
    servicePort: 80
-host: coffee.test.com
http:
paths:
-path:/
backend:
serviceName: coffee-svc
servicePort: 80
```

Note You can download the yaml files that are shown in this topic from the following link:

https://github.com/nginxinc/kubernetes-ingress/tree/master/examples/complete-example

For more information on a sample scenario of implementing Ingress, see Deploying Cafe Application with Ingress.

L4 Ingress

NGINX supports L4 TCP and UDP Ingress load balancing. It uses the NGINX helm chart that contains the TCP or UDP service mappings, instead of the Ingress resources as in the case of L7 support.

Configuring L4 Load Balancing

Note NGINX supports either TCP or UDP L4 load balancing, but not both simultaneously.

Step 1 Access the Kubernetes cluster master node using ssh.

```
ssh -l <username> <IP address of master node>
```

Note Once you create a Kubernetes cluster, it may take a few minutes for the necessary services to start. If ssh to a cluster fails, we recommend that you try again after a few minutes.

Step 2 Get the current helm configuration values.

```
helm get values --all nginx-ingress > l4.yaml
```

Step 3 Edit the `l4.yaml` file.

You can search for *tcp* or *udp* in the `l4.yaml` file, and then add your L4 services.

The following example shows adding the tcp-test-svc TCP service that uses port 3333.

```
tcp:
    "9000": default/tcp-test-svc:3333
```

The following example shows adding the udp-test-svc UDP service that uses port 5005.

```
udp:
    "9001": default/udp-test-svc:5005
```

Step 4 Update the NGINX helm chart with the L4 service mappings.

```
helm upgrade --install nginx-ingress /opt/ccp/charts/nginx-ingress.tgz -f l4.yaml
```

Note You need to restart the NGINX Ingress controller pods for the new configuration to take effect.

Step 5 Verify that ingress has successfully mapped the port.

```
kubectl get services -o wide -w nginx-ingress-controller
```

Ingress CA

Cisco Container Platform by default creates an L7 Ingress service in order to support Monitoring Health of Cluster Deployments, Monitoring Logs from Cluster Deployments, and Setting up Kubernetes Dashboard. All of these services are exposed with TLS enabled, and the certificate authority (CA) that is used to sign the Ingress controller server certificate is self-signed and per cluster based.

In order to reach the services without triggering SSL warning, you can either add the CA as part of your application that needs to interact with services behind Cisco Container Platform ingress (preferred), or add the CA to your system trusted CA list.

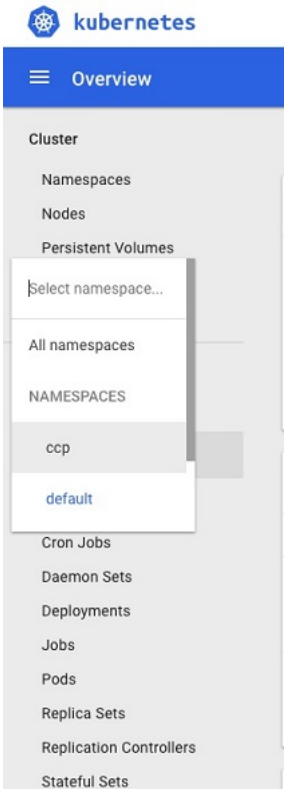
Obtaining CA Certificate for Nginx Ingress Controller

To obtain a CA certificate.

Step 1 Log in to the Kubernetes dashboard from browser as described in Setting up Kubernetes Dashboard section, download the kubeconfig file, and then use it to login to the Kubernetes dashboard.

Step 2 From the right pane, click the dropdown box under **Namespace**, click the **ccp** namespace.

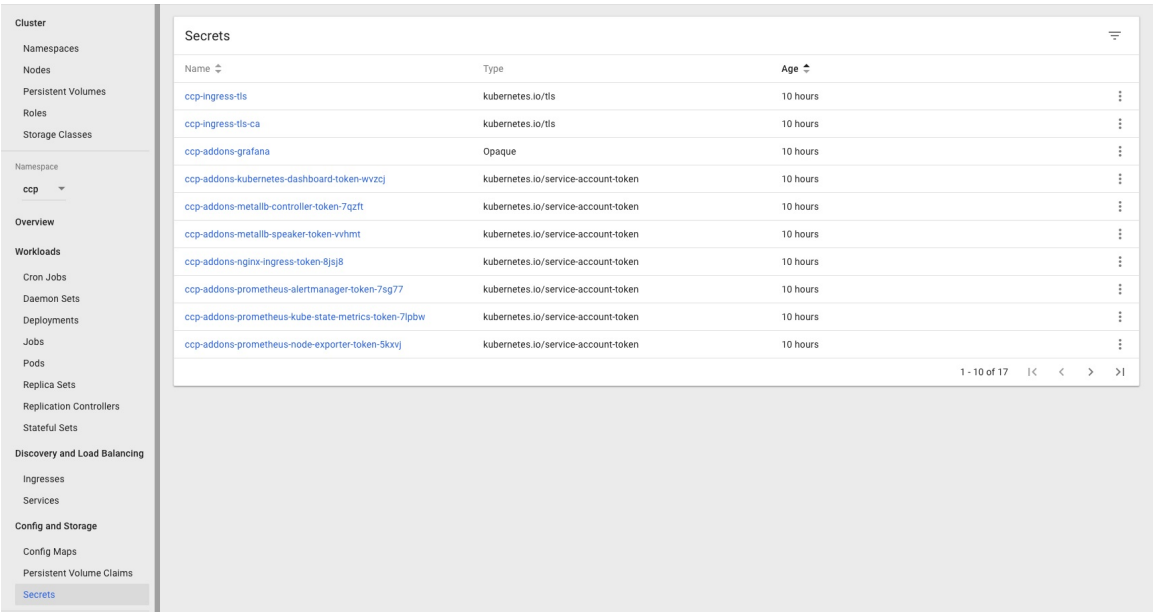
Figure 1: Kubernetes Dashboard



Step 3 Click the **Secrets** tab.

The **Secrets** pane appears.

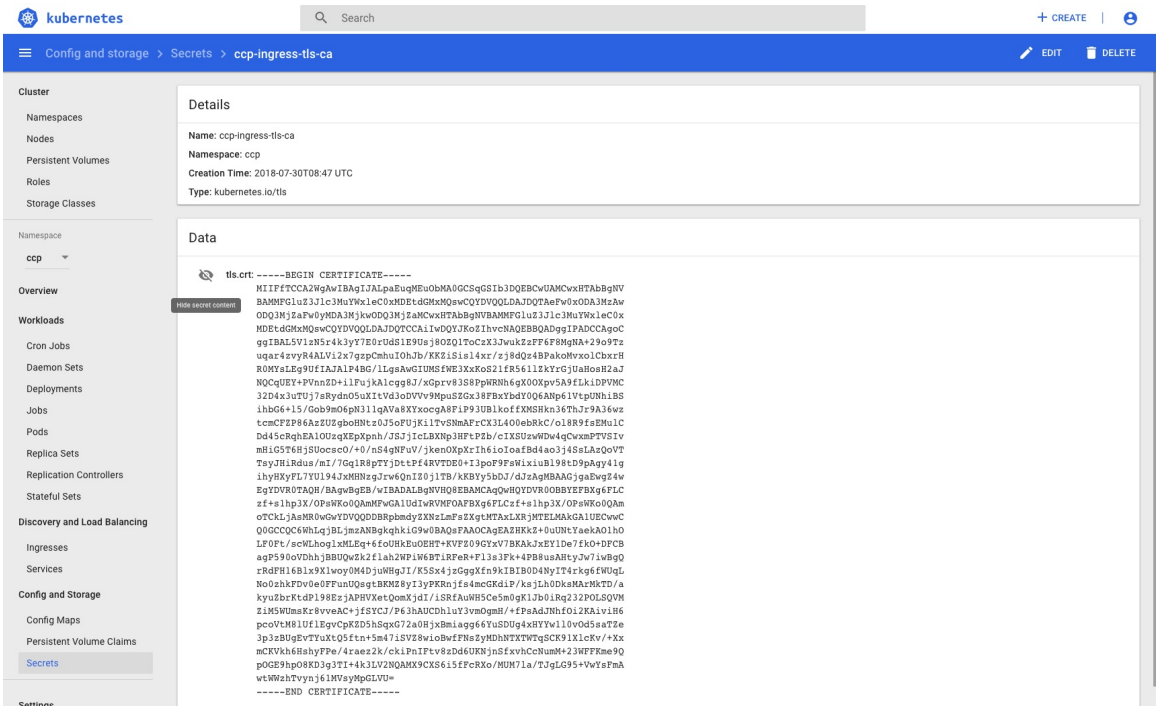
Figure 2: Secrets Pane



Step 4 Open the `ccp-ingress-tls-ca` secret and find the data for `tls.crt`.

Step 5 Click the **Eye** icon to view the details of a `tls.crt`.

Figure 3: Secrets Pane Showing Details of tls.crt



You can save the CA data into a file, and use it when a client is trying to connect to the Ingress service.

The following example uses **curl** to get to the dashboard using the saved CA certificate.

```
curl --cacert ./ca.crt -I https://10.10.99.185/dashboard
 HTTP/1.1 200 OK
 Server: nginx/1.13.12
 Date: Mon, 30 Jul 2018 19:08:11 GMT
 Content-Type: text/html; charset=utf-8
 Connection: keep-alive
 Vary: Accept-Encoding
 Accept-Ranges: bytes
 Cache-Control: no-store
 Strict-Transport-Security: max-age=15724800; includeSubDomains
```

Updating CA Certificates for Nginx Ingress Controller

To update certificates configured for the Nginx Ingress controller with custom certificates that are not provisioned using Cisco Container Platform:

Step 1 On the control plane node of your Cisco Container Platform instance, backup the existing daemonset ingress-nginx-controller resources.

```
kubectl get daemonset ingress-nginx-controller -o yaml > ds_ingress_nginx_controller_ccp.yaml
```

Step 2 Create or copy the TLS key file and TLS crt file of the new certificate to a known location on the control plane node.

Step 3 Create a new TLS secret in the default namespace configured for the new custom certificates.

```
kubectl create secret tls custom-certificate-secret --key=/path/to/tls.key --cert=/path/to/tls.crt
```

Step 4 Update the existing daemonset to use the new certificates.

```
kubectl patch daemonset ingress-nginx-controller --type='json'
-p='[{"op":"replace","path":"/spec/template/spec/containers/0/args/6","value":"--default-ssl-certificate=default/custom-certificate-secret"}]'
```

Step 5 Check the status of the daemonset.

```
ccpuser@ccp800-master80bcc3ccdc:~$ kubectl get ds ingress-nginx-controller

NAME                       DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
AGE

ingress-nginx-controller   3         3         3       3            3           <none>
73d
```

Note Ensure that the **UP-TO-DATE** count and **READY** count are equal.

Step 6 Verify the certificates used by the Cisco Container Platform dashboard in one of the following ways:

- Using the browser.

  This depends on the type of browser you are using. Check the browser settings and instructions.

- Using the CLI.

  Run the following curl commands to verify if the new custom certificates have been installed for your Cisco Container Platform instance.

  ```
  $ curl --insecure -vvI https://<master_vip_address>
  * Rebuilt URL to: https://10.10.96.6/
  ...
  ```

```
      * Server certificate:
      *  subject: CN=ingress.ccp800; OU=server
      *  start date: Jan 15 20:27:17 2021 GMT
      *  expire date: Jan 15 20:27:17 2023 GMT
      *  issuer: CN=ingress.ccp800; OU=CA
      *  SSL certificate verify result: unable to get local issuer certificate (20), continuing
anyway.
      * Using HTTP2, server supports multi-use
      * Connection state changed (HTTP/2 confirmed)
      * Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
      ...
      * Connection #0 to host 10.10.96.6 left intact
```

Reverting CA Certificates for Nginx Ingress Controller

To revert to the original certificate settings that you configured when you installed Cisco Container Platform:

Step 1 When Cisco Container Platform is installed, it configures an ingress-nginx-controller with self-signed certificates that is stored in the `default/ccp-ingress-tls` Kubernetes secret. Ensure that this secret exists on the control plane.

```
kubectl get secret ccp-ingress-tls
```

Step 2 Update the daemonset ingress-nginx-controller with this secret.

```
kubectl patch daemonset ingress-nginx-controller --type='json'
-p='[{"op":"replace","path":"/spec/template/spec/containers/0/args/6","value":"--default-ssl-certificate=default/ccp-ingress-tls"}]'
```

Step 3 Wait for the daemonset to be ready.

```
ccpuser@ccp800-master80bcc3ccdc:~$ kubectl get ds ingress-nginx-controller

NAME                       DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
AGE

ingress-nginx-controller   3         3         3       3            3           <none>
73d
```

Note Ensure that the **UP-TO-DATE** count and **READY** count are equal.

The certificate configuration will now be restored to the original certificate settings that you configured when you installed Cisco Container Platform.

Step 4 Verify the certificates used by the Cisco Container Platform dashboard in one of the following ways:

- Using the browser.

  This depends on the type of browser you are using. Check the browser settings and instructions

- Using the CLI.

  Run the following curl commands to verify if the new custom certificates have been installed for your Cisco Container Platform instance.

```
$ curl --insecure -vvI https://<master_vip_address>
* Rebuilt URL to: https://10.10.96.6/
...
* Server certificate:
*  subject: CN=ingress.ccp800; OU=server
```

```
*  start date: Jan 15 20:27:17 2021 GMT
*  expire date: Jan 15 20:27:17 2023 GMT
*  issuer: CN=ingress.ccp800; OU=CA
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing
anyway.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
...
* Connection #0 to host 10.10.96.6 left intact
```

Network Policies

Cisco Container Platform supports Kubernetes NetworkPolicies. The NetworkPolicies are independent of the underlying container network plugin.

Load Balancer Services

Cisco Container Platform supports load balancer services on tenant clusters.

While creating a tenant cluster, you need to choose the number of load balancer IP addresses that you want to allocate for a tenant cluster from a VIP pool that you want to use.

Note The cluster creation operation fails if the number of requested load balancer IP addresses is more than the available IP addresses in the pool.

For more information, see Creating Clusters on vSphere.

Once load balancer IP addresses are allocated for a tenant cluster, externally reachable load balancer IP addresses are automatically provisioned for the load balancer services.

The following code provides an example of creating a service of type **LoadBalancer**.

```
apiVersion: v1
kind: Service
metadata:
     name: frontend
     labels:
             app: guestbook
             tier: frontend
     type: LoadBalancer
```

You can update the number of available load balancer IP addresses from the **Edit Cluster** screen. You need to be aware of the number of used addresses in order to update the number of allocated load balancer IP addresses.

For example:

Suppose the current tenant is allocated with five load balancer IP addresses. If there are three load balanced services running, you cannot reduce the number of load balancer IP addresses to three or less as there are services using those IP addresses already.

Note When you delete a tenant cluster, the allocated load balancer IP addresses are recycled to the VIP pool.