Guided Exercise: Protect External Traffic with TLS

- Expose an application that is secured by TLS certificates.

Outcomes

- Deploy an application and create an unencrypted route for it.
- Create an OpenShift edge route with encryption.
- Update an OpenShift deployment to support a new version of the application.
- Create an OpenShift TLS secret and mount it to your application.
- Verify that the communication to the application is encrypted.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

The command ensures that the cluster API is reachable, and creates the `network-ingress` OpenShift project. The command also gives the `developer` user edit access on the project.

```
[student@workstation ~]$ lab start network-ingress
```

Instructions

As an application developer, you are ready to deploy your application in OpenShift. In this activity, you deploy two versions of the application: one that is exposed over unencrypted traffic (HTTP), and one that is exposed over secure traffic (HTTPS).

The container image, which is accessible at `https://registry.ocp4.example.com:8443/redhattraining/todo-angular`, has two tags: `v1.1`, which is the insecure version of the application, and `v1.2`, which is the secure version. Your organization uses its own certificate authority (CA) that can sign certificates for the following domains:

- `*.apps.ocp4.example.com`
- `*.ocp4.example.com`

The CA certificate is accessible at `~/DO280/labs/network-ingress/certs/training-CA.pem`.
The `passphrase.txt` file contains a unique password that protects the CA key.
The `certs` directory also contains the CA key.

1. Log in to the OpenShift cluster and create the `network-ingress` project.

   Log in to the cluster as the `developer` user.

   ```
   [student@workstation ~]$ oc login -u developer -p developer \
       https://api.ocp4.example.com:6443
   Login successful.

   ...output omitted...
   ```

Create the `network-ingress` project.

```
[student@workstation ~]$ oc new-project network-ingress
Now using project "network-ingress" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

2. The OpenShift deployment file for the application is accessible at `~/DO280/labs/network-ingress/todo-app-v1.yaml`. The deployment points to `registry.ocp4.example.com:8443/redhattraining/todo-angular:v1.1`, which is the initial and unencrypted version of the application. The file defines the `todo-http` service that points to the application pod.

   Create the application and expose the service.

   Use the `oc create` command to deploy the application in the `network-ingress` OpenShift project.

```
[student@workstation ~]$ oc create -f \
    ~/DO280/labs/network-ingress/todo-app-v1.yaml
Warning: would violate PodSecurity "restricted:v1.24": ...output omitted...
deployment.apps/todo-http created
service/todo-http created
```

> **NOTE**
>
> It is safe to ignore pod security warnings for exercises in this course. OpenShift uses the Security Context Constraints controller to provide safe defaults for pod security.

   Wait a couple of minutes, so that the application can start, and then review the resources in the project.

```
[student@workstation ~]$ oc status
In project network-ingress on server https://api.ocp4.example.com:6443

svc/todo-http - 172.30.247.75:80 -> 8080
  deployment/todo-http deploys registry.ocp4.example.com:8443/redhattraining/todo-angula
r:v1.1
    deployment #1 running for 16 seconds - 1 pod
...output omitted...
```

   Run the `oc expose` command to create a route for accessing the application. Give the route a hostname of `todo-http.apps.ocp4.example.com`.

```
[student@workstation ~]$ oc expose svc todo-http \
    --hostname todo-http.apps.ocp4.example.com
route.route.openshift.io/todo-http exposed
```

   Retrieve the name of the route and copy it to the clipboard.

```
[student@workstation ~]$ oc get routes
NAME        HOST/PORT                          PATH   SERVICES    PORT   ...
todo-http   todo-http.apps.ocp4.example.com           todo-http   8080   ...
```

On the `workstation` machine, open Firefox and access the application URL. Confirm that you can see the application.

- `http://todo-http.apps.ocp4.example.com`

Open a new terminal tab and run the `tcpdump` command with the following options to intercept the traffic on port 80:

- `-i eth0` intercepts traffic on the main interface.
- `-A` strips the headers and prints the packets in ASCII format.
- `-n` disables DNS resolution.
- `port 80` is the port of the application.

  Optionally, use the `grep` command to filter on JavaScript resources.

  Start by retrieving the name of the main interface, whose IP is `172.25.250.9`.

```
[student@workstation ~]$ ip addr | grep 172.25.250.9
inet 172.25.250.9/24 brd 172.25.250.255 scope global noprefixroute eth0

[student@workstation ~]$ sudo tcpdump -i eth0 -A -n port 80 | grep "angular"
```

> **NOTE**
>
> The full command is available at `~/DO280/labs/network-ingress/tcpdump-command.txt`.

On Firefox, refresh the page and notice the activity in the terminal. Press **Ctrl**+**C** to stop the capture.

```
...output omitted...
<script type="text/javascript" src="assets/js/libs/angular/angular.min.js">
<script type="text/javascript" src="assets/js/libs/angular/angular-route.min.js">
<script type="text/javascript" src="assets/js/libs/angular/angular-animate.min.js">
...output omitted...
```

3. Create a secure edge route. Edge certificates encrypt the traffic between the client and the router, but leave the traffic between the router and the service unencrypted. OpenShift generates its own certificate that it signs with its CA.

   In later steps, you extract the CA to ensure that the route certificate is signed.

   Go to `~/DO280/labs/network-ingress` and run the `oc create route` command to define the new route.

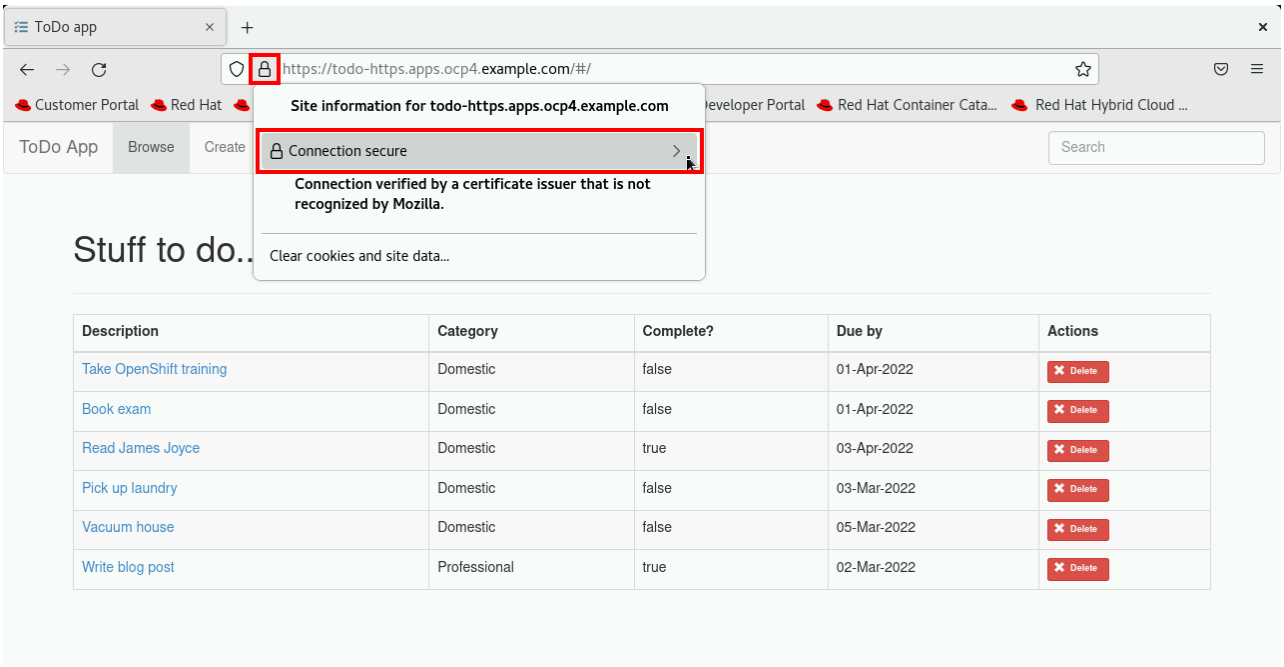
   Give the route a hostname of `todo-https.apps.ocp4.example.com`.

```
[student@workstation ~]$ cd ~/DO280/labs/network-ingress
[student@workstation network-ingress]$ oc create route edge todo-https \
    --service todo-http \
    --hostname todo-https.apps.ocp4.example.com
route.route.openshift.io/todo-https created
```

To test the route and read the certificate, open Firefox and access the application URL.

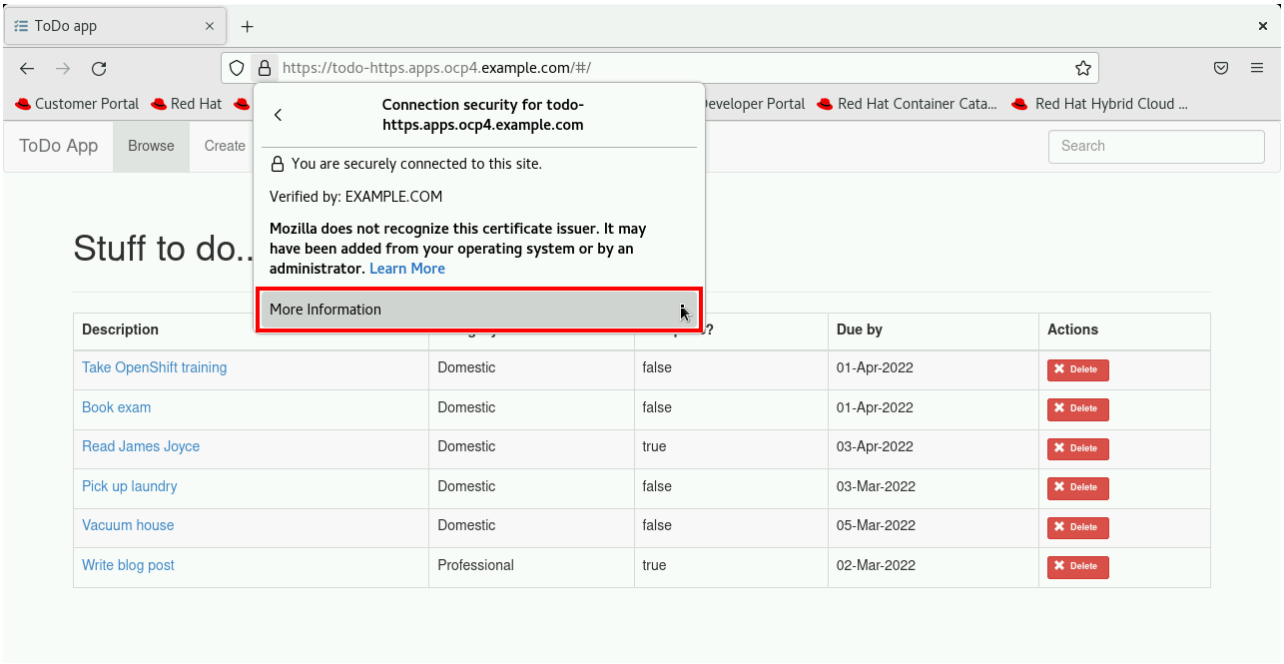
- `https://todo-https.apps.ocp4.example.com`

  Click the **padlock**, and then click the arrow next to **Connection secure**.

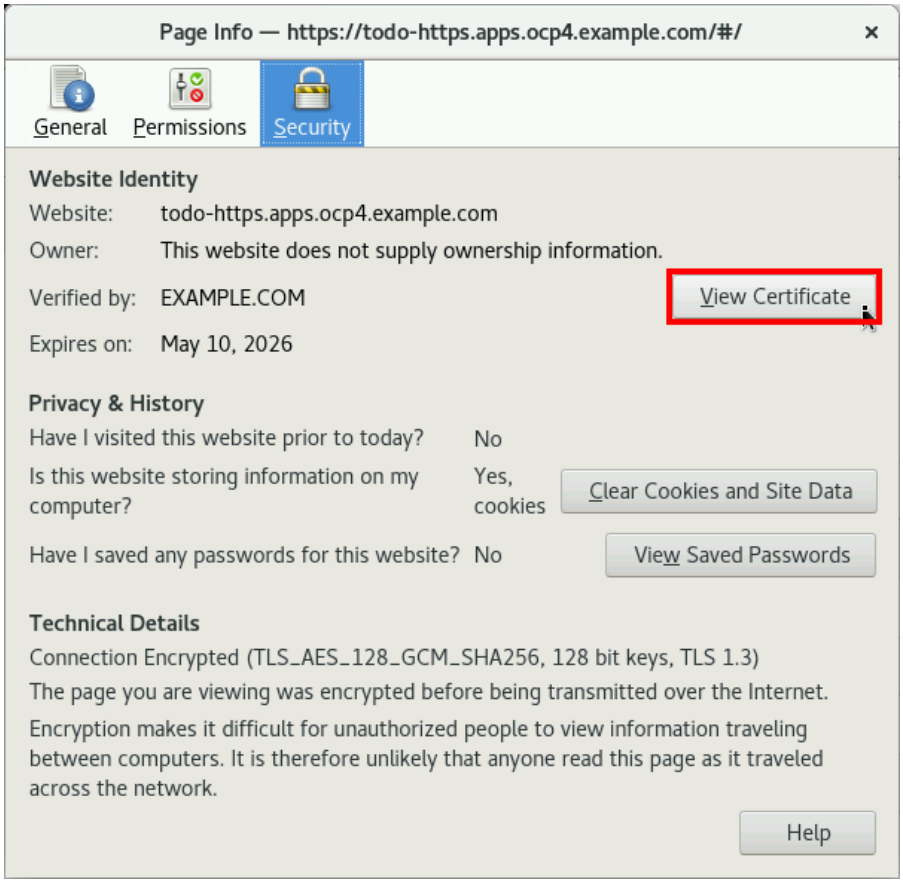


  Firefox displays a message that the connection is verified by a certificate issuer that Mozilla does not recognize. This message is displayed because the route signed certificate comes from an internal CA that is installed on the classroom OS. This CA, although not recognized by Mozilla, is valid for the lab environment purposes. If your organization uses a custom public key infrastructure (PKI), then you might see the same message.

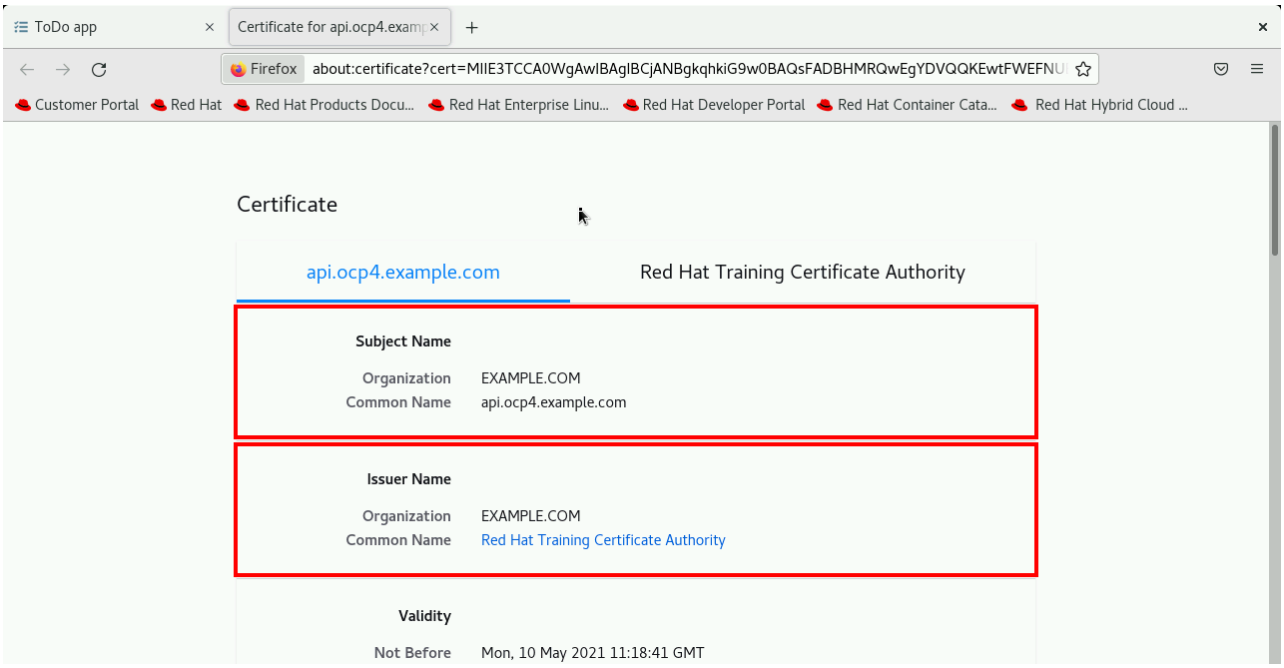
  Click **More Information** to display the page information window.



Click **View Certificate** to display the certificate information.



Locate the CN entry to see that the OpenShift ingress operator created the certificate with its own CA.



From the terminal, use the `curl` command with the `-I` and `-v` options to retrieve the connection headers.

The `Server certificate` section shows some information about the certificate. The alternative name matches the name of the route. The output indicates that the remote certificate is trusted because it matches the CA.

```
[student@workstation network-ingress]$ curl -I -v \
    https://todo-https.apps.ocp4.example.com
...output omitted...
* Server certificate:
*  subject: O=EXAMPLE.COM; CN=.api.ocp4.example.com
*  start date: May 10 11:18:41 2021 GMT
*  expire date: May 10 11:18:41 2026 GMT
*  subjectAltName: host "todo-https.apps.ocp4.example.com" matched cert's "*.apps.ocp4.ex
ample.com"
*  issuer: O=EXAMPLE.COM; CN=Red Hat Training Certificate Authority
*  SSL certificate verify ok.
...output omitted...
```

Although the traffic is encrypted at the edge with a certificate, you can still access the insecure traffic at the service level, because the pod behind the service does not offer an encrypted route.

Retrieve the IP address of the `todo-http` service.

```
[student@workstation network-ingress]$ oc get svc todo-http \
    -o jsonpath="{.spec.clusterIP}{'\n'}"
172.30.102.29
```

Create a debug pod in the `todo-http` deployment. Use the Red Hat Universal Base Image (UBI), which contains tools to interact with containers.

```
[student@workstation network-ingress]$ oc debug -t deployment/todo-http \
    --image registry.ocp4.example.com:8443/ubi8/ubi:8.4
Starting pod/todo-http-debug ...
Pod IP: 10.131.0.255
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

From the debug pod, use the `curl` command to access the service over HTTP. Replace the IP address with the one that you obtained in a previous step.

The output indicates that the application is available over HTTP.

```
sh-4.4$ curl -v 172.30.102.29
* Rebuilt URL to: 172.30.102.29/
*   Trying 172.30.102.29...
* TCP_NODELAY set
* Connected to 172.30.102.29 (172.30.102.29) port 80 (#0)
> GET / HTTP/1.1
> Host: 172.30.102.29
> User-Agent: curl/7.61.1
> Accept: */*
>
< HTTP/1.1 200 OK
...output omitted...
```

Exit the debug pod.

```
sh-4.4$ exit
Removing debug pod ...
```

Delete the edge route. In the following steps, you define the passthrough route.

```
[student@workstation network-ingress]$ oc delete route todo-https
route.route.openshift.io "todo-https" deleted
```

4. Generate TLS certificates for the application.

   In the following steps, you generate a CA-signed certificate that you attach as a secret to the pod. You then configure a secure route in passthrough mode and let the application expose that certificate.

   Go to the `~/DO280/labs/network-ingress/certs` directory and list the files.

```
[student@workstation network-ingress]$ cd certs
[student@workstation certs]$ ls -l
total 20
-rw-rw-r--. 1 student student  604 Nov 29 17:35 openssl-commands.txt
-rw-r--r--. 1 student student   33 Nov 29 17:35 passphrase.txt
-rw-r--r--. 1 student student 1743 Nov 29 17:35 training-CA.key
-rw-r--r--. 1 student student 1363 Nov 29 17:35 training-CA.pem
-rw-r--r--. 1 student student  406 Nov 29 17:35 training.ext
```

Generate the private key for your CA-signed certificate.

> **NOTE**
>
> The following commands for generating a signed certificate are all available in the `~/DO280/labs/network-ingress/certs/openssl-commands.txt` file.

```
[student@workstation certs]$ openssl genrsa -out training.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
...output omitted...
e is 65537 (0x010001)
```

Generate the certificate signing request (CSR) for the `todo-https.apps.ocp4.example.com` hostname.

```
[student@workstation certs]$ openssl req -new \
    -key training.key -out training.csr \
    -subj "/C=US/ST=North Carolina/L=Raleigh/O=Red Hat/\
    CN=todo-https.apps.ocp4.example.com"
```

> **WARNING**
>
> Type the request **subject** on one line. Alternatively, remove the `-subj` option and its content. Without the `-subj` option, the `openssl` command prompts you for the values; indicate a common name (`CN`) of `todo-https.apps.ocp4.example.com`.

Finally, generate the signed certificate. Notice the use of the `-CA` and `-CAkey` options for signing the certificate against the CA. Use the `-passin` option to reuse the password of the CA. Use the `extfile` option to define a *Subject Alternative Name (SAN)*.

```
[student@workstation certs]$ openssl x509 -req -in training.csr \
    -passin file:passphrase.txt \
    -CA training-CA.pem -CAkey training-CA.key -CAcreateserial \
    -out training.crt -days 1825 -sha256 -extfile training.ext
Signature ok
subject=C = US, ST = North Carolina, L = Raleigh, O = Red Hat, CN = todo-https.apps.ocp4.
example.com
Getting CA Private Key
```

Ensure that the newly created certificate and key are present in the current directory.

```
[student@workstation certs]$ ls -lrt
total 36
-rw-r--r--. 1 student student  599 Jul 31 09:35 openssl-commands.txt
-rw-r--r--. 1 student student   33 Aug  3 12:38 passphrase.txt
-rw-r--r--. 1 student student  352 Aug  3 12:38 training.ext
-rw-------. 1 student student 1743 Aug  3 12:38 training-CA.key
-rw-r--r--. 1 student student 1334 Aug  3 12:38 training-CA.pem
-rw-------. 1 student student 1675 Aug  3 13:38 training.key
-rw-rw-r--. 1 student student 1017 Aug  3 13:39 training.csr
-rw-rw-r--. 1 student student   41 Aug  3 13:40 training-CA.srl
-rw-rw-r--. 1 student student 1399 Aug  3 13:40 training.crt
```

Return to the `network-ingress` directory. This step is important, because the next step involves creating a route that uses the self-signed certificate.

```
[student@workstation certs]$ cd ~/DO280/labs/network-ingress
```

5. Deploy a new version of your application.

   The new version of the application expects a certificate and a key inside the container at `/usr/local/etc/ssl/certs`. The web server in that version is configured with SSL support. Create a secret to import the certificate from the `workstation` machine. In a later step, the application deployment requests that secret and exposes its content to the container at `/usr/local/etc/ssl/certs`.

   Create a `tls` OpenShift secret named `todo-certs`. Use the `--cert` and `--key` options to embed the TLS certificates. Use `training.crt` as the certificate, and `training.key` as the key.

```
[student@workstation network-ingress]$ oc create secret tls todo-certs \
    --cert certs/training.crt --key certs/training.key
secret/todo-certs created
```

   The deployment file at `~/DO280/labs/network-ingress/todo-app-v2.yaml` points to version 2 of the container image. The new version of the application is configured to support SSL certificates. Run `oc create` to create a deployment that uses that image.

```
[student@workstation network-ingress]$ oc create -f todo-app-v2.yaml
Warning: would violate PodSecurity "restricted:v1.24": ...output omitted...
deployment.apps/todo-https created
service/todo-https created
```

   Wait a couple of minutes to ensure that the application pod is running. Copy the pod name to your clipboard.

```
[student@workstation network-ingress]$ oc get pods
NAME                          READY   STATUS    RESTARTS   AGE
...output omitted...
todo-https-59d8fc9d47-265ds   1/1     Running   0          62s
```

   Review the volumes that are mounted inside the pod. The output indicates that the certificates are mounted to `/usr/local/etc/ssl/certs`.

```
[student@workstation network-ingress]$ oc describe pod \
    todo-https-59d8fc9d47-265ds | grep -A2 Mounts
  Mounts:
    /usr/local/etc/ssl/certs from tls-certs (ro)
    /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qrtkj (ro)
```

6. Create the secure route.

   Run the `oc create route` command to define the new route.

   Give the route a hostname of `todo-https.apps.ocp4.example.com`.

```
[student@workstation network-ingress]$ oc create route passthrough todo-https \
    --service todo-https --port 8443 \
    --hostname todo-https.apps.ocp4.example.com
route.route.openshift.io/todo-https created
```

   Use the `curl` command in verbose mode to test the route and to read the certificate. Use the `--cacert` option to pass the CA certificate to the `curl` command.

   The output indicates a match between the certificate chain and the application certificate. This match indicates that the OpenShift router forwards only packets that are encrypted by the application web server certificate.

```
[student@workstation network-ingress]$ curl -vv -I \
    --cacert certs/training-CA.pem \
    https://todo-https.apps.ocp4.example.com
...output omitted...
* Server certificate:
*  subject: C=US; ST=North Carolina; L=Raleigh; O=Red Hat; CN=todo-https.apps.ocp4.exampl
e.com
*  start date: Jun 15 01:53:30 2021 GMT
*  expire date: Jun 14 01:53:30 2026 GMT
*  subjectAltName: host "todo-https.apps.ocp4.example.com" matched cert's "*.apps.ocp4.ex
ample.com"
*  issuer: C=US; ST=North Carolina; L=Raleigh; O=Red Hat; CN=ocp4.example.com
*  SSL certificate verify ok.
...output omitted...
```

7. Create a debug pod to further confirm proper encryption at the service level.

   Retrieve the IP address of the `todo-https` service.

```
[student@workstation network-ingress]$ oc get svc todo-https \
    -o jsonpath="{.spec.clusterIP}{'\n'}"
172.30.121.154
```

   Create a debug pod in the `todo-https` deployment with the Red Hat UBI container image.

```
[student@workstation network-ingress]$ oc debug -t deployment/todo-https \
    --image registry.ocp4.example.com:8443/ubi8/ubi:8.4
Starting pod/todo-https-debug ...
Pod IP: 10.128.2.129
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

From the debug pod, use the `curl` command to access the service over HTTP. Replace the IP address with the one that you obtained in a previous step.

The output indicates that the application is not available over HTTP, and the web server redirects you to the secure version.

```
sh-4.4$ curl -I http://172.30.121.154
HTTP/1.1 301 Moved Permanently
Server: nginx/1.14.1
Date: Tue, 15 Jun 2021 02:01:19 GMT
Content-Type: text/html
Connection: keep-alive
Location: https://172.30.121.154:8443/
```

Finally, access the application over HTTPS. Use the `-k` option, because the container does not have access to the CA certificate.

```
sh-4.4$ curl -s -k https://172.30.121.154:8443 | head -n5
<!DOCTYPE html>
<html lang="en" ng-app="todoItemsApp" ng-controller="appCtl">
<head>
    <meta charset="utf-8">
    <title>ToDo app</title>
```

Exit the debug pod.

```
sh-4.4$ exit
Removing debug pod ...
```

8. Clean up the exercise directory and project.

   Change to the home directory.

   ```
   [student@workstation network-ingress]$ cd
   ```

   Delete the `network-ingress` project.

   ```
   [student@workstation ~]$ oc delete project network-ingress
   project.project.openshift.io "network-ingress" deleted
   ```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-ingress
```

This concludes the section.