

Rancher 2 and Letsencrypt

🕒 13 minute read

I decided to write this post to help with the [discussion on the Rancher Forum](https://forums.rancher.com/t/rancher-2-and-letsencrypt/10492) (<https://forums.rancher.com/t/rancher-2-and-letsencrypt/10492>), regarding the difficulties many were having trying to setup Letsencrypt certificates with cert-manager. I've borrowed and owe credit to work that's already been documented [here](https://www.idealcoders.com/posts/rancher/2018/06/rancher-2-x-and-lets-encrypt-with-cert-manager-and-nginx-ingress/) (<https://www.idealcoders.com/posts/rancher/2018/06/rancher-2-x-and-lets-encrypt-with-cert-manager-and-nginx-ingress/>), and I'll try to stick to the steps I took to enable the full automation of the certificate process.

NOTE: This post has been updated to demonstrate usage of a newer version of cert-manager provided by JetPack. Notices from Letsencrypt have been sent regarding the blocking of versions older than v0.8.0.

Prerequisites

I'm going to assume a few things for brevity.

- You have a functioning Rancher 2.0 Cluster
- You have kubectl set up with your [Rancher Kubeconfig File](https://rancher.com/docs/rancher/v2.x/en/cluster-admin/kubectl/#accessing-clusters-with-kubectl-and-a-kubeconfig-file) (<https://rancher.com/docs/rancher/v2.x/en/cluster-admin/kubectl/#accessing-clusters-with-kubectl-and-a-kubeconfig-file>)
- You have a publicly reachable dns service that points the domain, for which you want to issue certificates, to your cluster nodes, loadbalancer or port forwarder if using NAT.
- If your cluster is behind NAT you have set up split DNS. (See section on DNS)

Installation

Enable JetPack Helm Repository

The default library repository in Rancher only includes cert-manager versions up to v0.5.2. The first thing we need to do is add the JetPack repository to Rancher.

In the Rancher UI navigation go to **Tools** and select **Catalogs**.



Global ▾

Clusters

Apps

Users

Settings

Security ▾

Tools ▾

Catalogs

Drivers

RKE Templates

Catalogs

Delete

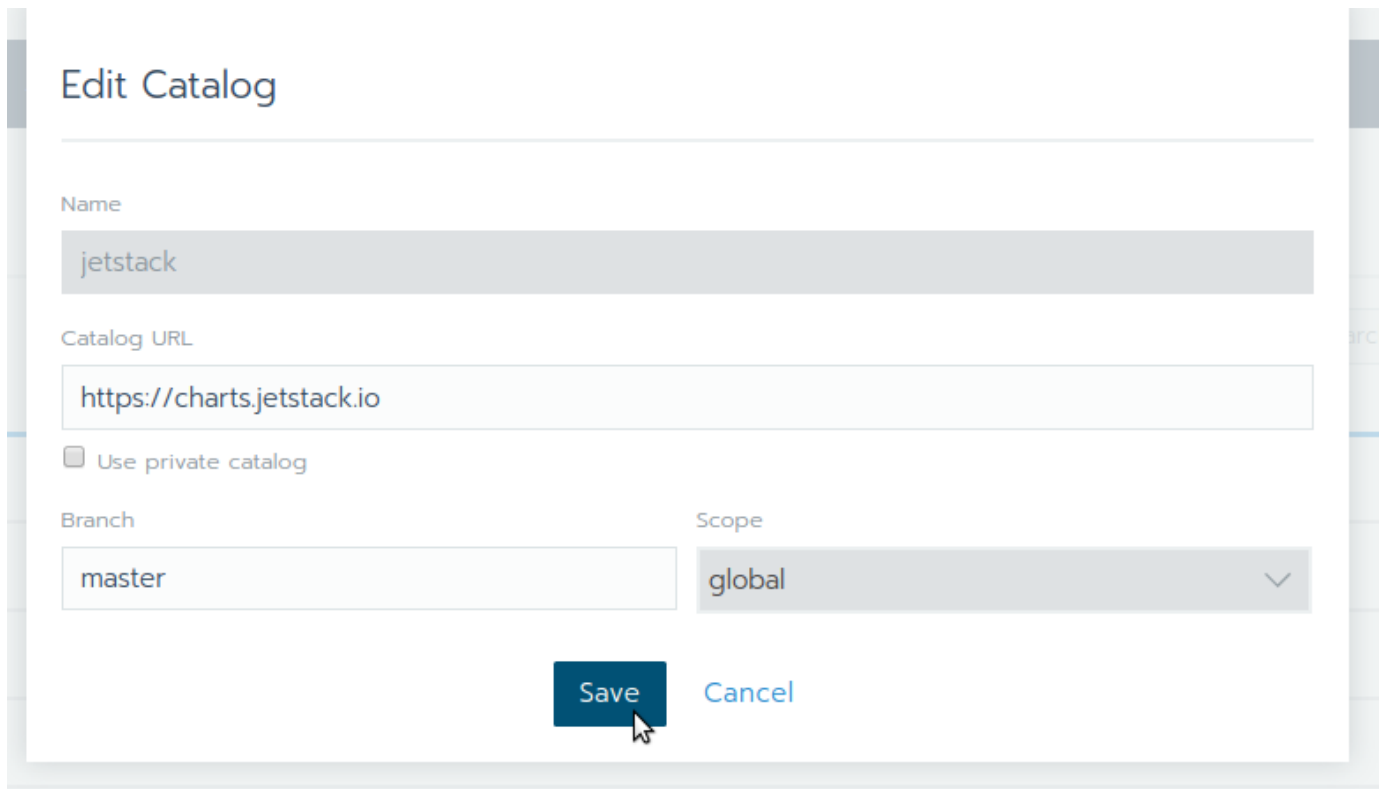
<input type="checkbox"/> State ▾	Scope ▾	Name ▾	Catalog URL ▾
<input type="checkbox"/> Disabled	Global	alibaba-app-hub	https://apphub.aliyuncs.com
<input type="checkbox"/> Active	Global	bitnami	https://charts.bitnami.com
<input type="checkbox"/> Active	Global	helm	https://kubernetes-charts.st
<input type="checkbox"/> Disabled	Global	helm-incubator	https://kubernetes-charts-ir

Next click the **Add Catalog** button.

Branch ▾

master	
master	

Give the new catalog a name like `jetstack` and configure `https://charts.jetstack.io` as the catalog URL.



Install cert-manager

Lets start by ensuring we are working from a clean slate. If you have attempted to install cert-manager before remove any existing resources and verify that the required name space isn't in use.

```
~$ kubectl get all -n cert-manager  
No resources found.
```

```
~$ kubectl describe clusterissuers letsencrypt-staging  
error: the server doesnt have a resource type "clusterissuers"
```

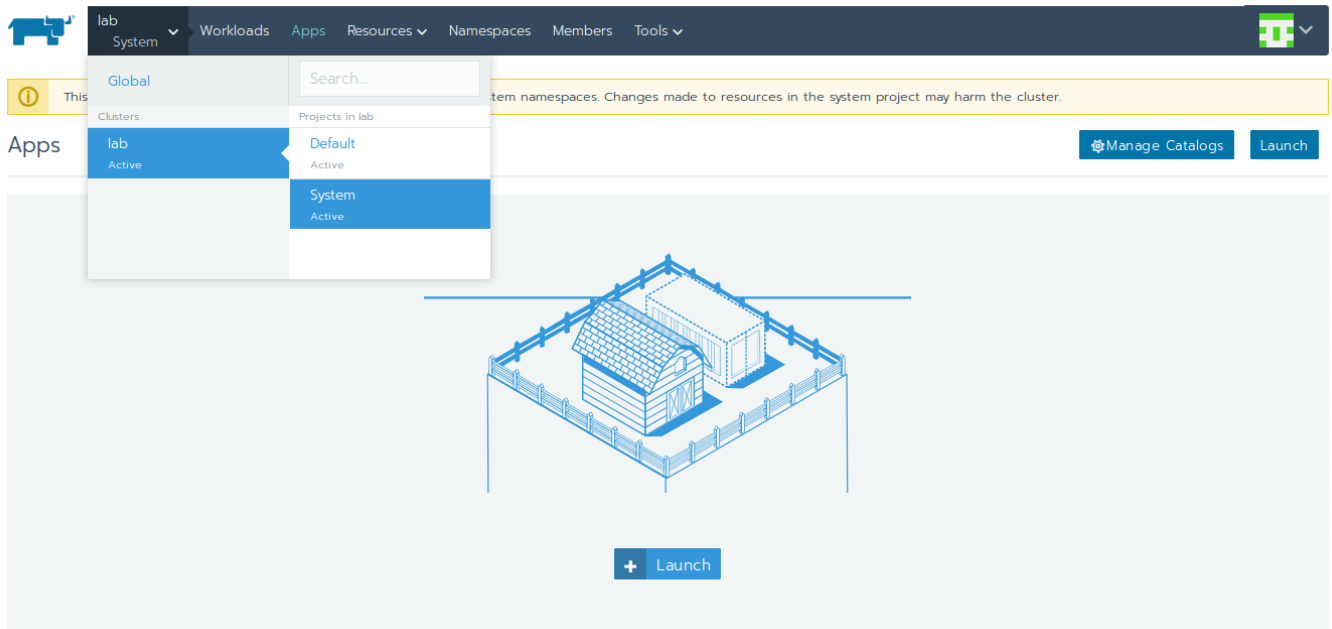
Before installing the cert-manager application in Rancher we need to first add the Customer Resource Definition (<https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/#customresourcedefinitions>). From your workstation execute the following.

```
kubectl apply -f https://raw.githubusercontent.com/jetstack/cert-manager/release-0.9/deploy/manifests/00-crds.yaml
```

Now label the kube-system namespace to disable resource validation.

```
kubectl label namespace kube-system certmanager.k8s.io/disable-validation=true
```

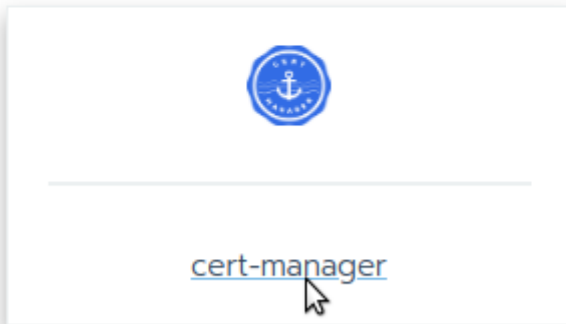
Next navigate to the **Apps** section of the Rancher **System** Project.



Next click the **Launch** button and type **cert** in the search menu. Click on the cert-manager provided by the JetStack catalog.

Catalog

jetstack



Change the namespace to **kube-system** and set the template version to **v0.9.1**.

Previous versions of cert-manager used the namespace 'cert-manager'. Make sure you deploy to kube-system or the installation will fail.

Detailed Descriptions
Application information and user guide

Configuration Options
Helm templates accept a comma separated list of strings

Name

cert-manager

Add a Description

Template Version

v0.11.0

Select a version of the template t

NAMESPACE

Namespace *

kube-system

Choose a Namespace...

cattle-system

cert-manager

external-dns

ingress-nginx

kube-node-lease

kube-public

kube-system

longhorn-system

metallb

nfs-provisioner

PREVIEW

Verify Installation

Now verify your app deployment with kubectl.

```
~$ kubectl get all -n kube-system | grep cert-manager
pod/cert-manager-5b9ff77b7-lhsb4          1/1      Running    0           165m
pod/cert-manager-cainjector-59d69b9b-9f5ng 1/1      Running    0           165m
pod/cert-manager-webhook-cfd6587ff-fz2cv   1/1      Running    0           125m
service/cert-manager-webhook      ClusterIP  10.43.104.116  <none>      443/TCP
165m
deployment.apps/cert-manager          1/1      1          1           165m
deployment.apps/cert-manager-cainjector 1/1      1          1           165m
deployment.apps/cert-manager-webhook   1/1      1          1           165m
replicaset.apps/cert-manager-5b9ff77b7 1         1          1           165m
replicaset.apps/cert-manager-cainjector-59d69b9b 1         1          1           165m
replicaset.apps/cert-manager-webhook-cfd6587ff 1         1          1           165m
```

Unlike previous versions of the Rancher cert-manager application, you'll need to create your own **Cluster Issuer** (<https://docs.cert-manager.io/en/release-0.9/reference/clusterissuers.html>). First create a file similar to the following.

```
~$ cat cluster-issuer.yaml
apiVersion: certmanager.k8s.io/v1alpha1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging
spec:
  acme:
    # The ACME server URL
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    # Email address used for ACME registration
    email: 2stacks@2stacks.net
    # Name of a secret used to store the ACME account private key
    privateKeySecretRef:
      name: letsencrypt-staging-account-key
    # Enable HTTP01 validations
    http01: {}
```

Now apply the configuration with;

```
kubectl create -f cluster-issuer.yaml
```

Verify the creation of the Cluster Issuer with;

```
kubectl describe clusterissuers letsencrypt-
```

The important thing to note is that the cert-manager pod is running and that your email account was successfully registered with the ACME server API. If your output isn't similar to above check the logs of the cert-manager pod for any issues.

The screenshot displays the Rancher management interface. At the top, a yellow warning banner states: "This is the system project which has all Kubernetes and Rancher system namespaces. Changes made to resources in the system project may harm the cluster." Below this, navigation tabs include Workloads, Load Balancing, Service Discovery, Volumes, and Pipelines. Action buttons for Redeploy, Pause Orchestration, Download YAML, and Delete are visible. The main table lists pods across two namespaces: 'cattle-system' and 'cert-manager'. In the 'cert-manager' namespace, the pod 'cert-manager-58c7cf9fb4-xlwkl' is highlighted with a blue 'Running' status badge. A context menu is open for this pod, showing options like 'Execute Shell', 'View Logs' (indicated by a red arrow), 'View/Edit YAML', 'View in API', and 'Delete'.

Notice in the logs below the message that says `Not syncing ingress default/nginx` as it does not contain necessary annotations. I precreated a test nginx deployment which we are going to use to test the `ingress-shim` functionality of cert-manager.

Logs: cert-manager

Connected

ProTip: Hold the Control key when opening logs to launch a new window.

```
5/30/2019 4:41:39 PM 10530 20:41:39.089978 1 start.go:79] starting cert-manager v0.5.2 (revision 9e8c3ad899c5aafaa360ca947eac7f5ba6301035)
5/30/2019 4:41:39 PM 10530 20:41:39.090728 1 server.go:84] Listening on http://0.0.0.0:9402
5/30/2019 4:41:39 PM 10530 20:41:39.094081 1 controller.go:126] Using the following nameservers for DNS01 checks: [10.43.0.10:53]
5/30/2019 4:41:39 PM 10530 20:41:39.097348 1 leaderelection.go:175] attempting to acquire leader lease cert-manager/cert-manager-controller...
5/30/2019 4:41:39 PM 10530 20:41:39.163397 1 leaderelection.go:184] successfully acquired lease cert-manager/cert-manager-controller
5/30/2019 4:41:39 PM 10530 20:41:39.167787 1 controller.go:68] Starting clusterissuers controller
5/30/2019 4:41:39 PM 10530 20:41:39.167823 1 controller.go:68] Starting ingress-shim controller
5/30/2019 4:41:39 PM 10530 20:41:39.167902 1 controller.go:68] Starting issuers controller
5/30/2019 4:41:39 PM 10530 20:41:39.167976 1 controller.go:68] Starting certificates controller
5/30/2019 4:41:44 PM 10530 20:41:44.244660 1 controller.go:168] ingress-shim controller: syncing item 'default/nginx'
5/30/2019 4:41:44 PM 10530 20:41:44.245087 1 sync.go:65] Not syncing ingress default/nginx as it does not contain necessary annotations
5/30/2019 4:41:44 PM 10530 20:41:44.245187 1 controller.go:182] ingress-shim controller: Finished processing work item "default/nginx"
5/30/2019 4:41:44 PM 10530 20:41:44.394052 1 controller.go:140] clusterissuers controller: syncing item 'letsencrypt-staging'
5/30/2019 4:41:44 PM 10530 20:41:44.394618 1 setup.go:73] letsencrypt-staging: generating acme account private key 'letsencrypt-staging-account-key'
5/30/2019 4:41:45 PM 10530 20:41:45.121561 1 logger.go:88] Calling GetAccount
5/30/2019 4:41:46 PM 10530 20:41:46.108560 1 logger.go:83] Calling CreateAccount
5/30/2019 4:41:46 PM 10530 20:41:46.224767 1 setup.go:181] letsencrypt-staging: verified existing registration with ACME server
5/30/2019 4:41:46 PM 10530 20:41:46.224810 1 helpers.go:147] Setting lastTransitionTime for ClusterIssuer "letsencrypt-staging" condition "Ready" to 2019-05-30 20:41:46.224800837 +0000
5/30/2019 4:41:46 PM 10530 20:41:46.253559 1 controller.go:154] clusterissuers controller: Finished processing work item "letsencrypt-staging"
5/30/2019 4:41:51 PM 10530 20:41:51.255739 1 controller.go:140] clusterissuers controller: syncing item 'letsencrypt-staging'
5/30/2019 4:41:51 PM 10530 20:41:51.256983 1 setup.go:144] Skipping re-verifying ACME account as cached registration details look sufficient.
5/30/2019 4:41:51 PM 10530 20:41:51.257127 1 controller.go:154] clusterissuers controller: Finished processing work item "letsencrypt-staging"
```

☐ Wrap lines

[Scroll to Top](#)

[Scroll to Bottom](#)

[Download Logs](#)

[Clear Screen](#)


[Close](#)

Configuring Ingress


Deploy a Test Workload

I chose to deploy an nginx container as a test since it provides the default server and nginx welcome page without any configuration.

From the `Workloads` section of your chosen Rancher Project click the `Deploy` button. Give the workload a name, choose the `nginx` `Docker Image` of your choice, leave the `Namespace` set to default. Add a `Port Mapping` for port 80 and publish the service as a `Cluster IP (Internal only)`. Click `Launch` to create the new workload.



lab
Default
Workloads
Apps
Resources
Namespaces
Members
Tools



Deploy Workload

Name
Add a Description

nginx

Workload Type
More options

Scalable deployment of 1 pod

Docker Image
Add to a new namespace

nginx:alpine

Namespace

default

Port Mapping

Publish the container port
Protocol
As a
On listening port

80
TCP
Cluster IP (Internal only)
Same as container port

+ Add Port

Create an Ingress

Next from the **Load Balancing** menu, click the **Add Ingress** button.

In the **Add Ingress** configuration page, give the Ingress a name and leave the Namespace set to default. Under the **Rules** section select the option for **Specify a hostname to use**. My test lab is setup to use the domain **bsptn.xyz** so I have configured the **Request Host** name as “**nginx.bsptn.xyz**”

By default Rancher chooses a **Workload** as the default for the **Target Backend**. We want to use the service that was automatically created when we deployed our nginx workload. Click the minus sign button to the right of the **Port** field to remove the existing **Target Backend**.

The screenshot shows the Rancher 'Add Ingress' configuration page. At the top, there's a navigation bar with 'lab Default' and various menu items. The main section is titled 'Add Ingress'. It has two input fields: 'Name' (containing 'nginx') and 'Namespace' (a dropdown menu showing 'default'). Below these is the 'Rules' section. It has three radio buttons: 'Automatically generate a `nginx-100` hostname', 'Specify a hostname to use' (which is selected), and 'Use as the default backend' (with a note 'Ingress controller does not support default backend'). The 'Specify a hostname to use' option has a 'Request Host' input field containing 'nginx.bsptn.xyz'. Below the rules is the 'Target Backend' section, which has two buttons: '+ Service' and '+ Workload'. The '+ Service' button is highlighted. Below this is a table with three columns: 'Path', 'Target', and 'Port'. The first row has 'Path' as '/foo', 'Target' as 'Choose a Workload...', and 'Port' as '80'. A red arrow points to a minus sign button next to the 'Port' field. At the bottom right, there is a '+ Add Rule' button.

Now click the **Service** button next to **Target Backend**. Set the **Path** to “/” and in the **Target** drop down select the nginx service.

Rules

☐ Automatically generate a `nginx` hostname
 ☒ Specify a hostname to use
 ☐ Use as the default backend
Ingress controller does not support default backend

Request Host

ⓘ If the target is a service, only the port exposed by the service can be selected. You can go to Service Discovery tab and edit the service to add ports by editing the YAML.

Target Backend + Service + Workload

Path
 Target
 Port
-

+ Add Rule

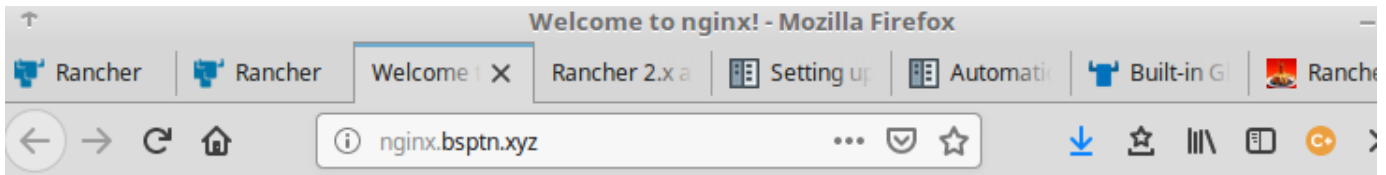
[Expand All](#)

▶ **SSL/TLS Certificates**
 Configure the certificates that will be presented for requests to encrypted ports.

▶ **Labels & Annotations**
 Key/Value pairs that can be used to label/annotate containers and make scheduling decisions.
 None

Save Cancel

At this point you should save the ingress without configuring any SSL Certificates or Annotations. You should verify that your nginx deployment is reachable via the ingress from both the Internet and your internal network. If you can not then you have more work to do with DNS, Load Balancing, NAT etc. before you can proceed to the next step.



Welcome to nginx!

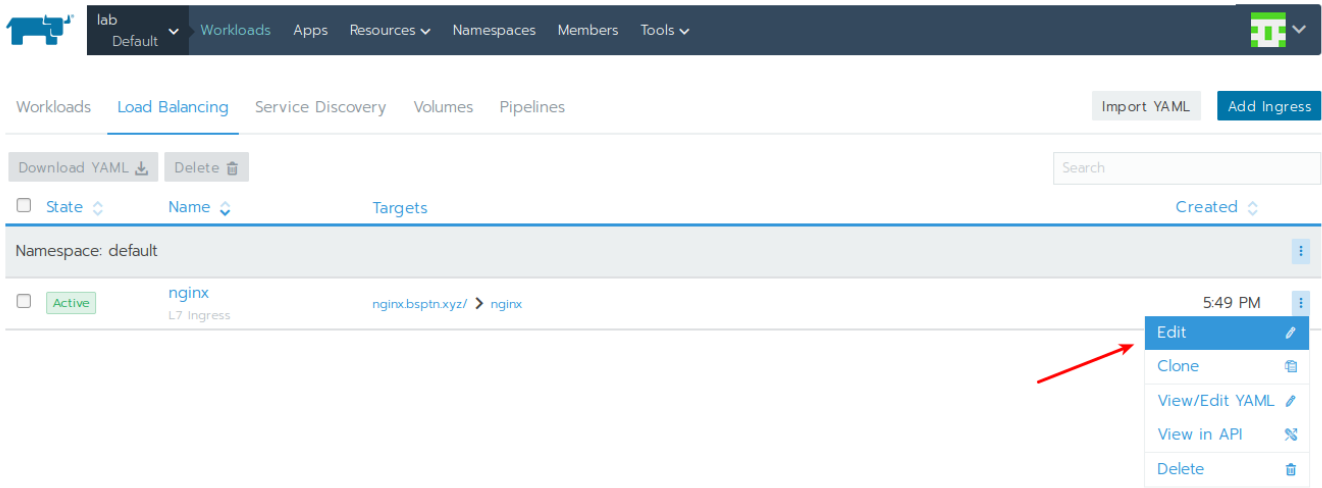
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

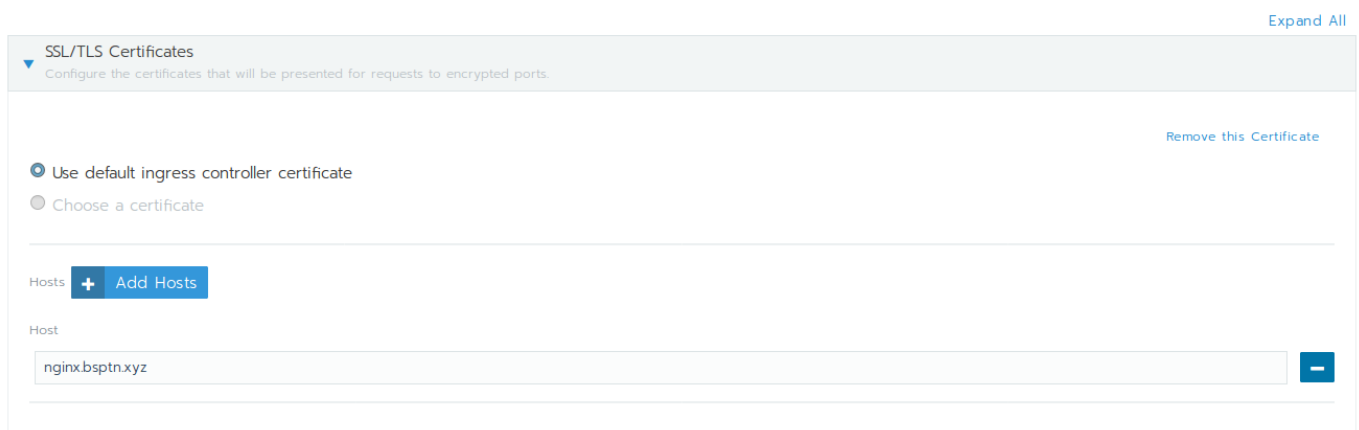
Thank you for using nginx.

Edit the Ingress

At this point if you have verified that your ingress service is reachable you can proceed to adding the annotations required to automatically request and deploy a certificate. From the **Load Balancing** menu click the drop down to the far right of the nginx ingress and then select **edit**



Scroll to the bottom of the page and expand the **SSL/TLS Certificates** and **Labels & Annotations** sections. First click the **Add Certificate** button under the SSL/TLS section. Leave the option set for **Use default ingress controller certificate**. Don't worry we will manually edit this in the Yaml later. Set the **Host** section to the FQDN of your service.



Now you need to add the annotations as per the [ingress-shim](https://cert-manager.readthedocs.io/en/latest/tasks/issuing-certificates/ingress-shim.html#supported-annotations) (<https://cert-manager.readthedocs.io/en/latest/tasks/issuing-certificates/ingress-shim.html#supported-annotations>), documentation. If you forget the required annotations you can view the docs. They are also provided in the **Notes** section when you first launched the cert-manager app.



cert-manager

cert-manager is a Kubernetes addon to automate the management and issuance of TLS certificates from various issuing sources. It will ensure certificates are valid and up to date periodically, and attempt to renew certificates at an appropriate time before expiry.

How to Use It

Ingress-shim

Cert-manager will create Certificate resources that reference the `ClusterIssuer` for all Ingresses that have following annotations.

```
kubernetes.io/tls-acme: "true"
certmanager.k8s.io/cluster-issuer: letsencrypt-staging # your cluster issuer name
nginx.ingress.kubernetes.io/secure-backends: "true" # optional
```

For cert-manager to work properly, the following information has to be added on your ingress definition.

```
spec:
  tls:
  - hosts:
    - host.example.com
    secretName: host-example-crt
```

Under the **Labels & Annotations** section click the **Add Annotation** button twice and add the following annotations.

- `kubernetes.io/tls-acme: "true"`
- `certmanager.k8s.io/cluster-issuer: letsencrypt-staging`

Now click **Save** to update the Ingress.

SSL/TLS Certificates
Configure the certificates that will be presented for requests to encrypted ports.

Remove this Certificate

☒ Use default ingress controller certificate
 ☐ Choose a certificate

Hosts + Add Hosts

Host

—

Labels & Annotations
Key/Value pairs that can be used to label/annotate containers and make scheduling decisions.

4 Configured

Labels

Key *

Value

—

Annotations

Key *

Value

—

ProTip: Paste one or more lines of key-value pairs into any key field for easy bulk entry.

+ Add Label

ProTip: Paste lines of key-value pairs into any key field for easy bulk entry.

+ Add Annotation

Save

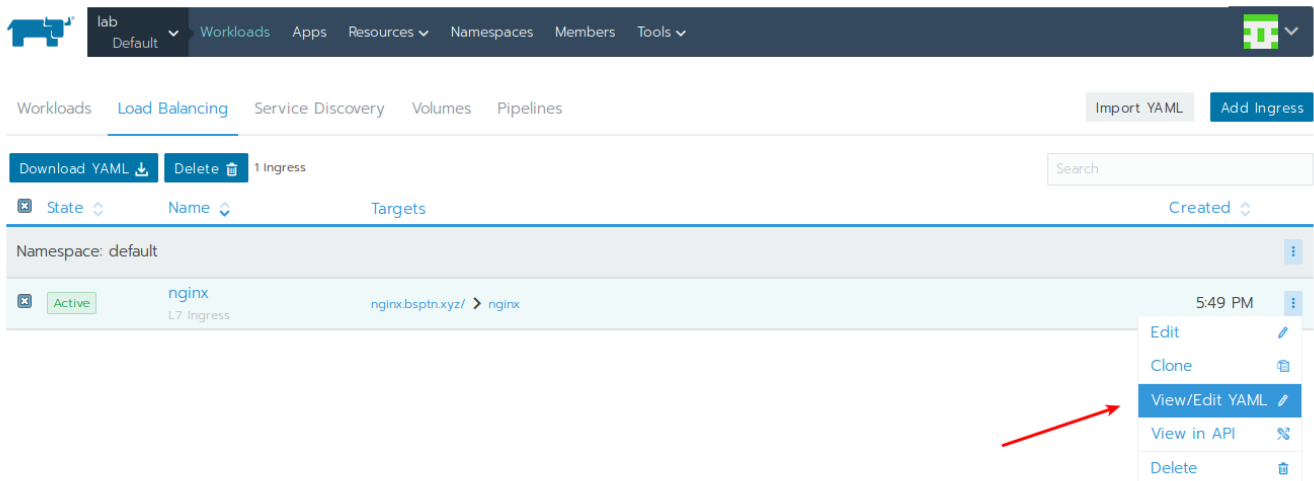
Cancel

The final step can not be performed through the Rancher Gui at this time. We'll need to manually edit the Yaml of the Ingress we just created.

<https://www.2stacks.net/blog/rancher-2-and-letsencrypt/>

11/24

From the **Load Balancing** menu click the drop down to the far right of the nginx ingress and then select **View/Edit YAML**.



Scroll the bottom of the Yaml config and under **spec -> tls -> hosts** add the **secretName** definition with the resource name you want the certificate to be saved with. I've chosen **nginx-bsptn-xyz-crt** for my implementation. Now click the save button.

Edit YAML: nginx

[Read from a file](#) [Download](#)

```

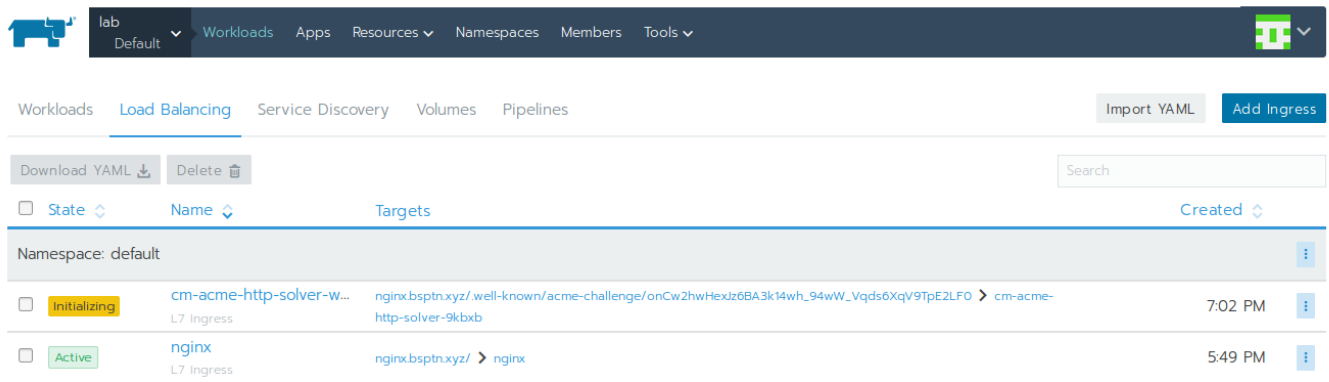
0  ["10.1.0.146"], "port": 443, "protocol": "HTTPS", "serviceName": "default:nginx", "ingressName": "default:nginx", "hostname": "nginx.bsptn.xyz", "path": "/", "allNodes": true]]'
9  kubernetes.io/tls-acme: "true"
10 creationTimestamp: "2019-05-30T21:49:03Z"
11 generation: 2
12 labels:
13   cattle.io/creator: norman
14 name: nginx
15 namespace: default
16 resourceVersion: "1019183"
17 selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/nginx
18 uid: bdb8b6c1-8324-11e9-8c27-52540042c274
19 spec:
20   rules:
21   - host: nginx.bsptn.xyz
22     http:
23       paths:
24       - backend:
25           serviceName: nginx
26           servicePort: 80
27         path: /
28   tls:
29   - hosts:
30     - nginx.bsptn.xyz
31     secretName: nginx-bsptn-xyz-crt
32 status:
33   loadBalancer:
34     ingress:
35     - ip: 10.1.0.146
36     - ip: 10.1.0.147
37     - ip: 10.1.0.148
38

```

[Copy to Clipboard](#)

[Save](#) [Cancel](#)

If everything to this point has gone well and if you watch carefully, cert-manager will temporarily create a new Ingress for the purposes of performing HTTP01 challenge verification.



State	Name	Targets	Created
Initializing	cm-acme-http-solver-w...	nginx.bsptn.xyz/.well-known/acme-challenge/onCw2hwHexiz6BA3k14wh_94wW_Vqds6XqV9TpE2LF0 > cm-acme-http-solver-9kxbxb	7:02 PM
Active	nginx	nginx.bsptn.xyz/ > nginx	5:49 PM

If you missed it or if something went wrong now is a good time to review the cert-manager logs. I've included my logs from the last time the ingress-shim failed due to missing configurations up until the certificate is pulled and the temporary HTTP01 challenge Ingress is removed.

```
E0530 23:01:37.571902 1 controller.go:177] ingress-shim controller: Re-queuing item
"default/nginx" due to error processing: TLS entry 0 for ingress "nginx" must specify a
secretName
I0530 23:02:37.572406 1 controller.go:168] ingress-shim controller: syncing item
'default/nginx'
I0530 23:02:37.598364 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/nginx"
I0530 23:02:37.598406 1 controller.go:168] ingress-shim controller: syncing item
'default/nginx'
I0530 23:02:37.598430 1 sync.go:140] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
already exists
I0530 23:02:37.598462 1 sync.go:143] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
is up to date
I0530 23:02:37.598480 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/nginx"
I0530 23:02:39.598243 1 controller.go:171] certificates controller: syncing item
'default/nginx-bsptn-xyz-crt'
I0530 23:02:39.598704 1 sync.go:274] Preparing certificate default/nginx-bsptn-xyz-crt with
issuer
I0530 23:02:39.599901 1 prepare.go:263] Cleaning up previous order for certificate
default/nginx-bsptn-xyz-crt
I0530 23:02:39.599939 1 prepare.go:279] Cleaning up old/expired challenges for Certificate
default/nginx-bsptn-xyz-crt
I0530 23:02:39.599998 1 logger.go:38] Calling CreateOrder
I0530 23:02:40.148955 1 acme.go:126] Created order for domains: [{dns nginx.bsptn.xyz}]
I0530 23:02:40.149074 1 logger.go:73] Calling GetAuthorization
I0530 23:02:40.212749 1 logger.go:93] Calling HTTP01ChallengeResponse
I0530 23:02:40.212917 1 prepare.go:279] Cleaning up old/expired challenges for Certificate
default/nginx-bsptn-xyz-crt
I0530 23:02:40.212949 1 logger.go:68] Calling GetChallenge
I0530 23:02:40.340693 1 pod.go:65] No existing HTTP01 challenge solver pod found for
Certificate "default/nginx-bsptn-xyz-crt". One will be created.
I0530 23:02:40.394210 1 service.go:51] No existing HTTP01 challenge solver service found
for Certificate "default/nginx-bsptn-xyz-crt". One will be created.
I0530 23:02:40.506689 1 ingress.go:49] Looking up Ingresses for selector
certmanager.k8s.io/acme-http-domain=806880787,certmanager.k8s.io/acme-http-token=1172442703
I0530 23:02:40.506761 1 ingress.go:102] No existing HTTP01 challenge solver ingress found
for Certificate "default/nginx-bsptn-xyz-crt". One will be created.
I0530 23:02:40.595694 1 helpers.go:194] Setting lastTransitionTime for Certificate "nginx-
bsptn-xyz-crt" condition "Ready" to 2019-05-30 23:02:40.595666151 +0000 UTC
m=+8461.551329830
I0530 23:02:40.595767 1 sync.go:276] Error preparing issuer for certificate default/nginx-
bsptn-xyz-crt: http-01 self check failed for domain "nginx.bsptn.xyz"
E0530 23:02:40.595863 1 sync.go:197] [default/nginx-bsptn-xyz-crt] Error getting
certificate 'nginx-bsptn-xyz-crt': secret "nginx-bsptn-xyz-crt" not found
E0530 23:02:40.711924 1 controller.go:180] certificates controller: Re-queuing item
"default/nginx-bsptn-xyz-crt" due to error processing: http-01 self check failed for domain
"nginx.bsptn.xyz"
I0530 23:02:40.721555 1 controller.go:168] ingress-shim controller: syncing item
```

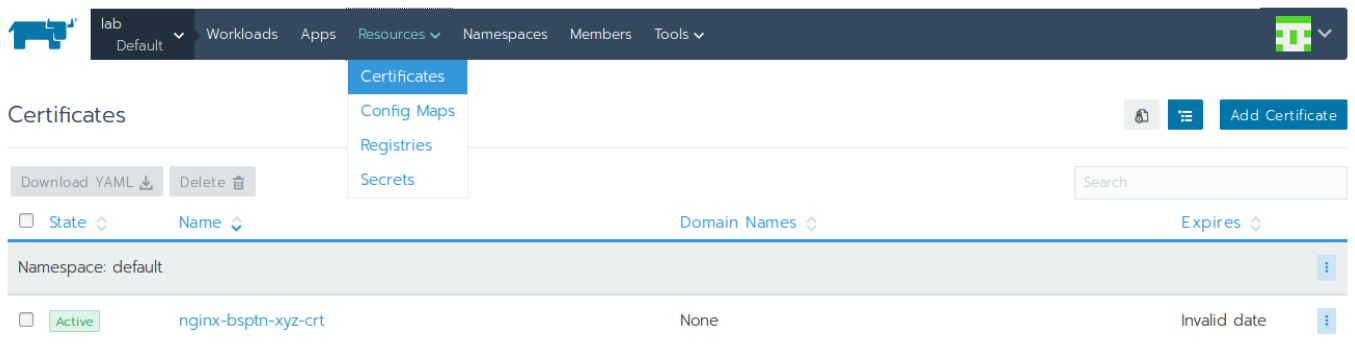
```
'default/nginx'
I0530 23:02:40.723070 1 sync.go:140] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
already exists
I0530 23:02:40.723508 1 sync.go:143] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
is up to date
I0530 23:02:40.723762 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/nginx"
I0530 23:02:44.713550 1 controller.go:171] certificates controller: syncing item
'default/nginx-bsptn-xyz-crt'
I0530 23:02:44.713701 1 sync.go:274] Preparing certificate default/nginx-bsptn-xyz-crt with
issuer
I0530 23:02:44.714024 1 logger.go:43] Calling GetOrder
I0530 23:02:44.808517 1 logger.go:73] Calling GetAuthorization
I0530 23:02:44.897577 1 logger.go:93] Calling HTTP01ChallengeResponse
I0530 23:02:44.897678 1 prepare.go:279] Cleaning up old/expired challenges for Certificate
default/nginx-bsptn-xyz-crt
I0530 23:02:44.897711 1 logger.go:68] Calling GetChallenge
I0530 23:02:44.973655 1 http.go:134] wrong status code '503'
I0530 23:02:44.974074 1 ingress.go:49] Looking up Ingresses for selector
certmanager.k8s.io/acme-http-domain=806880787,certmanager.k8s.io/acme-http-token=1172442703
I0530 23:02:44.974202 1 helpers.go:201] Found status change for Certificate "nginx-bsptn-
xyz-crt" condition "Ready": "False" -> "False"; setting lastTransitionTime to 2019-05-30
23:02:44.974192204 +0000 UTC m=+8465.929855813
I0530 23:02:44.974303 1 sync.go:276] Error preparing issuer for certificate default/nginx-
bsptn-xyz-crt: http-01 self check failed for domain "nginx.bsptn.xyz"
E0530 23:02:44.974380 1 sync.go:197] [default/nginx-bsptn-xyz-crt] Error getting
certificate 'nginx-bsptn-xyz-crt': secret "nginx-bsptn-xyz-crt" not found
I0530 23:02:45.004974 1 controller.go:168] ingress-shim controller: syncing item
'default/nginx'
I0530 23:02:45.005147 1 sync.go:140] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
already exists
I0530 23:02:45.005183 1 sync.go:143] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
is up to date
I0530 23:02:45.005248 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/nginx"
E0530 23:02:45.008794 1 controller.go:180] certificates controller: Re-queuing item
"default/nginx-bsptn-xyz-crt" due to error processing: http-01 self check failed for domain
"nginx.bsptn.xyz"
I0530 23:02:45.599202 1 controller.go:168] ingress-shim controller: syncing item
'default/cm-acme-http-solver-wtcbm'
I0530 23:02:45.599414 1 sync.go:65] Not syncing ingress default/cm-acme-http-solver-wtcbm
as it does not contain necessary annotations
I0530 23:02:45.599638 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/cm-acme-http-solver-wtcbm"
I0530 23:03:01.005455 1 controller.go:171] certificates controller: syncing item
'default/nginx-bsptn-xyz-crt'
I0530 23:03:01.006291 1 sync.go:274] Preparing certificate default/nginx-bsptn-xyz-crt with
issuer
I0530 23:03:01.007303 1 logger.go:43] Calling GetOrder
```

```
I0530 23:03:01.187043 1 logger.go:73] Calling GetAuthorization
I0530 23:03:01.271914 1 logger.go:93] Calling HTTP01ChallengeResponse
I0530 23:03:01.272196 1 prepare.go:279] Cleaning up old/expired challenges for Certificate
default/nginx-bsptn-xyz-crt
I0530 23:03:01.272583 1 logger.go:68] Calling GetChallenge
I0530 23:03:11.760008 1 prepare.go:488] Accepting challenge for domain "nginx.bsptn.xyz"
I0530 23:03:11.760125 1 logger.go:63] Calling AcceptChallenge
I0530 23:03:12.179202 1 prepare.go:500] Waiting for authorization for domain
"nginx.bsptn.xyz"
I0530 23:03:12.179361 1 logger.go:78] Calling WaitAuthorization
I0530 23:03:14.383630 1 prepare.go:510] Successfully authorized domain "nginx.bsptn.xyz"
I0530 23:03:14.383916 1 prepare.go:303] Cleaning up challenge for domain "nginx.bsptn.xyz"
as part of Certificate default/nginx-bsptn-xyz-crt
I0530 23:03:14.642912 1 ingress.go:49] Looking up Ingresses for selector
certmanager.k8s.io/acme-http-domain=806880787,certmanager.k8s.io/acme-http-token=1172442703
I0530 23:03:14.674932 1 sync.go:281] Issuing certificate...
I0530 23:03:14.675267 1 logger.go:43] Calling GetOrder
I0530 23:03:15.898507 1 logger.go:58] Calling FinalizeOrder
I0530 23:03:16.872750 1 issue.go:196] successfully obtained certificate:
cn="nginx.bsptn.xyz" altNames=[nginx.bsptn.xyz] url="https://acme-staging-
v02.api.letsencrypt.org/acme/order/9448784/35891936"
I0530 23:03:16.931122 1 sync.go:300] Certificate issued successfully
I0530 23:03:16.931385 1 helpers.go:201] Found status change for Certificate "nginx-bsptn-
xyz-crt" condition "Ready": "False" -> "True"; setting lastTransitionTime to 2019-05-30
23:03:16.931293187 +0000 UTC m=+8497.886956711
I0530 23:03:16.932560 1 sync.go:206] Certificate default/nginx-bsptn-xyz-crt scheduled for
renewal in 1438 hours
I0530 23:03:16.951754 1 controller.go:185] certificates controller: Finished processing
work item "default/nginx-bsptn-xyz-crt"
I0530 23:03:16.952862 1 controller.go:168] ingress-shim controller: syncing item
'default/nginx'
I0530 23:03:16.953088 1 sync.go:140] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
already exists
I0530 23:03:16.953906 1 sync.go:143] Certificate "nginx-bsptn-xyz-crt" for ingress "nginx"
is up to date
I0530 23:03:16.954138 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/nginx"
I0530 23:03:18.953126 1 controller.go:171] certificates controller: syncing item
'default/nginx-bsptn-xyz-crt'
I0530 23:03:18.954881 1 sync.go:206] Certificate default/nginx-bsptn-xyz-crt scheduled for
renewal in 1438 hours
I0530 23:03:18.955045 1 controller.go:185] certificates controller: Finished processing
work item "default/nginx-bsptn-xyz-crt"
I0530 23:03:19.674777 1 controller.go:168] ingress-shim controller: syncing item
'default/cm-acme-http-solver-wtcbm'
E0530 23:03:19.674894 1 controller.go:198] ingress 'default/cm-acme-http-solver-wtcbm' in
work queue no longer exists
I0530 23:03:19.674942 1 controller.go:182] ingress-shim controller: Finished processing
work item "default/cm-acme-http-solver-wtcbm"
```


Verification

If you get a log message similar to `issue.go:196] successfully obtained certificate: cn="nginx.bsptn.xyz"` chances are you are in good shape. I'll run through just a couple of things to verify everything is working.

Within the project in which you created your Ingress navigate to `Resources -> Certificates` and verify that your certificate resource has been created in the cluster.



The Rancher UI doesn't give a lot of information about the certificate so to view its details we'll need to use `Kubectl`.

```
~$ kubectl get certificates
NAME                AGE
nginx-bsptn-xyz-crt 26m
```

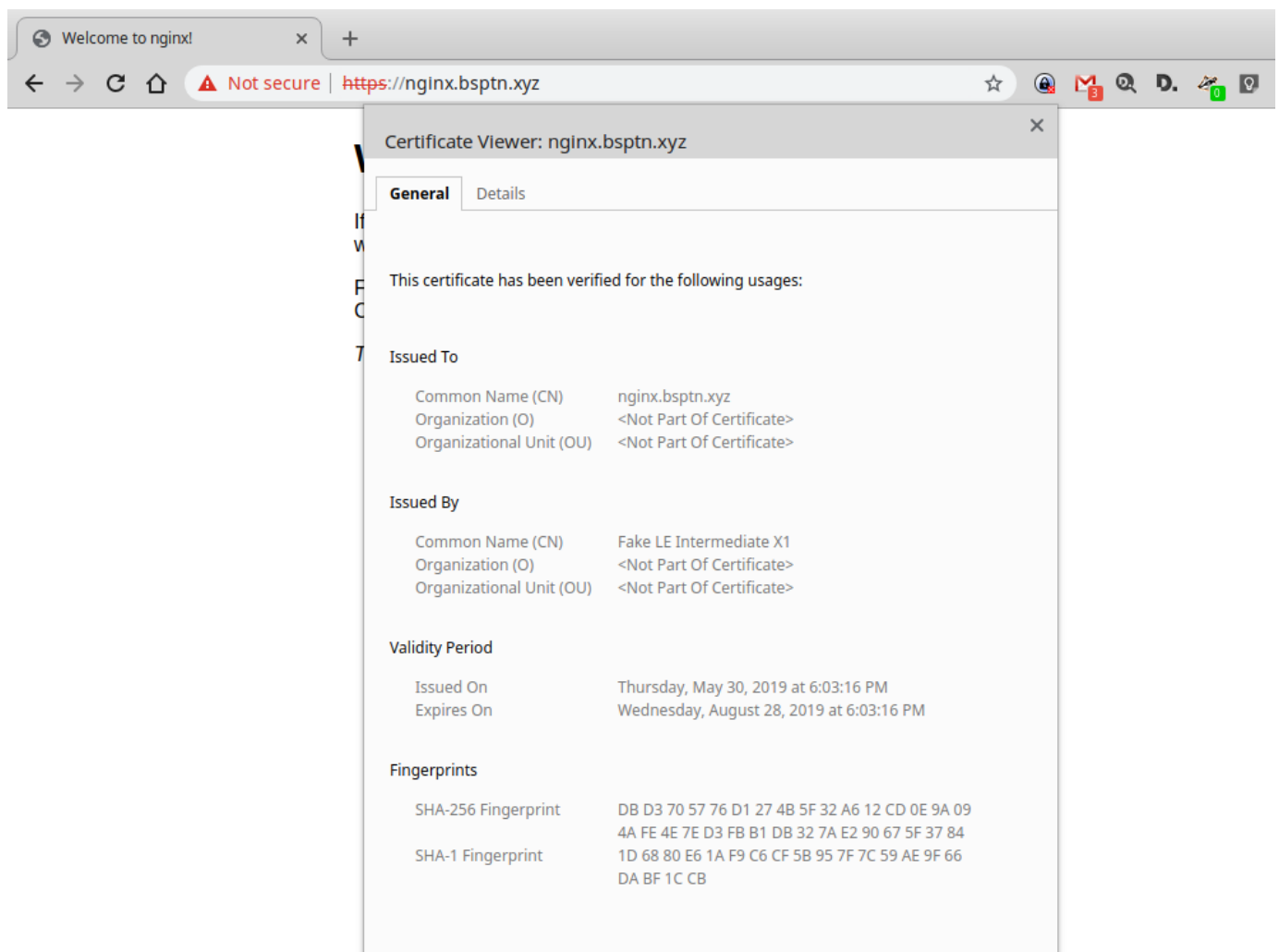
```
~$ kubectl describe certificates nginx-bsptn-xyz-crt
Name:          nginx-bsptn-xyz-crt
Namespace:     default
Labels:        <none>
Annotations:   <none>
API Version:   certmanager.k8s.io/v1alpha1
Kind:          Certificate
Metadata:
  Creation Timestamp:  2019-05-30T23:02:37Z
  Generation:         4
  Owner References:
    API Version:      extensions/v1beta1
    Block Owner Deletion:  true
    Controller:       true
    Kind:              Ingress
    Name:              nginx
    UID:               <some_uid>
  Resource Version:   1023500
  Self Link:
/apis/certmanager.k8s.io/v1alpha1/namespaces/default/certificates/nginx-bsptn-xyz-crt
  UID:               <some_uid>
Spec:
  Acme:
    Config:
      Domains:
        nginx.bsptn.xyz
      Http 01:
        Ingress:
  Dns Names:
    nginx.bsptn.xyz
  Issuer Ref:
    Kind:      ClusterIssuer
    Name:      letsencrypt-staging
  Secret Name: nginx-bsptn-xyz-crt
Status:
  Acme:
    Order:
      URL:  https://acme-staging-v02.api.letsencrypt.org/acme/order/<number>/<number>
  Conditions:
    Last Transition Time:  2019-05-30T23:03:16Z
    Message:              Certificate issued successfully
    Reason:               CertIssued
    Status:               True
    Type:                 Ready
    Last Transition Time:  <nil>
    Message:              Order validated
    Reason:               OrderValidated
    Status:               False
    Type:                 ValidateFailed
```

Events:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	CreateOrder	26m	cert-manager	Created new ACME order, attempting validation...
Normal	DomainVerified	25m	cert-manager	Domain "nginx.bsptn.xyz" verified with "http-01" validation
Normal	IssueCert	25m	cert-manager	Issuing certificate...
Normal	CertObtained	25m	cert-manager	Obtained certificate from ACME server
Normal	CertIssued	25m	cert-manager	Certificate issued successfully

Again, if everything worked under the `Events` section you should be able to see that the certificate was issued successfully.

The last obvious verification is to load the nginx web page and verify with the a browser that a LetsEncrypt certificate has been issued from the staging API. Since I chose to issue certs from the staging API the browser will still generate a certificate error however, you can see from the certificate details that it has been Issued By `Fake LE Intermediate X1`.



Additional Notes

I'm sure if this process works for you you'll want to proceed to issuing certs from LetsEncrypt's production API. I have test this exact same procedure using the `letsencrypt-prod` cluster issuer on a clean cluster. I have not attempted to run more than one cluster issue on the same Rancher cluster. If you're ready to issue valid certificates I recommend you delete the cert-manager app you deployed and start over. You should be able to follow all of the steps in this post replacing all instances have `letsencrypt-staging` with `letsencrypt-prod`.

DNS

You may or may not have noticed that may cluster nodes have private IP addresses. I'll share a little about my setup in case some of you are attempting to use cert-manager on a private lab network. I assume that clusters deployed in public clouds won't have as many http01 verification issues but I could be wrong.

- First, I have a wildcard dns record in AWS Route53 that points `*.bsptn.xyz` to a device performing NAT for my lab environment.
- That NAT boundary forwards all port 80 and 443 to an L4-7 loadbalancer that services my Kubernetes clusters.
- I have a private DNS server built with [PowerDNS](https://www.powerdns.com/) for internal name resolution of private IPs. I chose PowerDNS because it provides an API that integrates with the Kubernetes add on service [external-dns](https://github.com/kubernetes-incubator/external-dns).
- Inside my Kubernetes cluster's I deploy the Bitnami version of the external-dns application.

Any Ingress I create in my clusters is automatically registered in PowerDNS via the external-dns application. This make the process of performing HTTP01 verification much easier in my environment.

If you're interested in more details of how I set up my lab environment feel free to contact me. I have posted a lot of the work I've done to GitHub [@2stacks](https://github.com/2stacks) and I mostly use Terraform so that my deployments are repeatable.

 Tags: Kubernetes Letsencrypt Rancher

 Categories: Blog

 Updated: May 24, 2019

COMMENTS

2stacks Comment Policy

Keep It Classy, San Diego.

[19 Comments](#)[2stacks](#)[1 Login](#) ▼[♥ Recommend](#)[🐦 Tweet](#)[f Share](#)[Sort by Best](#) ▼

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)



Klaus Kobald • 23 days ago • edited

very good writing!

I got errors in the cert-manager that I solved by using this annotation:

```
cert-manager.io/cluster-issuer: letsencrypt-staging
```

^ | ▼ • Reply • Share ›



2stacks Mod ➔ Klaus Kobald • 23 days ago

I'm glad that worked. The new cert-manager api dropped 'k8s' among other changes.

^ | ▼ • Reply • Share ›



Klaus Kobald ➔ 2stacks • 22 days ago

what I do not understand: I have deleted letsencrypt-prod issuer but I did not delete the annotations in the ingress. Why is cert-manager issuing a self signed certificate and not simply producing an error, like "hey, I cannot find the issuer" ? so thinking this further it seems, that even if the issuer is installed it is not used.

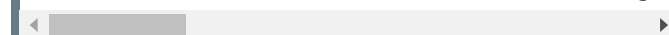
^ | ▼ • Reply • Share ›



Klaus Kobald ➔ 2stacks • 23 days ago • edited

something must be wrong completely:

```
I0510 21:34:02.521582 1 controller.go:
```



why does it say "selfsigned"?

the phrase letsencrypt does not appear anywhere in the log. the generated certificate for the project is created:

```
apiVersion: v1
data:
  ca.crt: ""
  tls.crt: ""
  tls.key: [....]
kind: Secret
metadata:
  annotations:
```

[see more](#)

^ | v • Reply • Share ›



Klaus Kobald ➔ 2stacks • 23 days ago

... what about the .well-known url ? I came accross that somewhere. Do I have to take care of this? or is cert-manager doing it all?

^ | v • Reply • Share ›



Klaus Kobald ➔ 2stacks • 23 days ago • edited

well almost. I now did the same thing with production from scratch. but I still get security warning in the browser and it says "fake certificate"

the log of the cert-manager looks good:

```
"msg"="certificate resource is already up to
```

a few minutes ago it was

```
"msg"="CertificateRequest is not in a final
```

am I missing something? how does letsencrypt know about my company? I only provided the email in the cluster-issuer.yml. Is there any more steps to fulfill?

^ | v • Reply • Share ›



Klaus Kobald ➔ Klaus Kobald
• 22 days ago

Frustrating - started over 5 times. I give up and try again somewhen. I wonder why all of a sudden it should be the cert-manager namespace and not kube-system. I think the problem is, that everybody has a different setun and you might have

different setup and you might have

something installed, that I don't have.

Fun fact: not even the helm 3 commands are in the right syntax - hihi

^ | v • Reply • Share ›



2stacks Mod ➔ Klaus Kobald

• 21 days ago

Yeah, I went through similar frustrations. You definitely have to purge everything cert-manager related before starting over with helm3. The name space, crds, api end points etc. have all changed.

^ | v • Reply • Share ›



2stacks Mod ➔ Klaus Kobald

• 23 days ago • edited

The e-mail and the http-01 verification is all Letsencrypt needs. Cert-manager automates the generation of the .well-known url for LetsEncrypt to validate against. Once its validated and a certificate is issued .well-known is removed.

So much has changed you might have better luck purging you existing cert-manager

<https://cert-manager.io/doc...>

And then re-installing with helm3

<https://hub.helm.sh/charts/...>

\$ kubectl apply --validate=false -f

<https://github.com/jetstack...>

^ | v • Reply • Share ›



Klaus Kobald ➔ 2stacks • 22 days ago

after installation, the first lines in the log are:

client_config.go:543] Neither --kubeconfig nor --master was specified. Using the inClusterConfig. This might not work.

maybe it's nothing, but is this right? Does your log look the same?

^ | v • Reply • Share ›



2stacks Mod ➔ Klaus Kobald

• 22 days ago

Doesn't look familiar. Which container produced that log?

^ | v • Reply • Share ›



Klaus Kobald → 2stacks • 22 days ago

cert-manager

^ | v • Reply • Share ›



Leo Critchley • a month ago

Thanks for writing this up, it was really useful!

^ | v • Reply • Share ›



2stacks Mod → Leo Critchley • a month ago

Thanks, Im using helm V3 to deploy now and the process is much simpler. I hope to update this info soon

1 ^ | v • Reply • Share ›



Klaus Kobald • 2 months ago

hm - it ends at the point: "select namespace". I can only select a project. And there is no kube-system. I am using the latest rancher / k8s any ideas?

^ | v • Reply • Share ›



2stacks Mod → Klaus Kobald • 2 months ago