kubernetes dashboard在ssl的各种场景下的手动部署

2019年4月17日 | 作者 张馆长 | 4900字 | 阅读大约需要10分钟
查看原文 | 归档于 kubernetes | 标签 #Kubernetes #Dashboard

> 本文转载自zhangguanzhang的博客。

旨在面向新手讲解手动部署过程，本文dashboard的暴露不会用nodePort（不喜欢使用它）和apiserver的web proxy代理也就是 `/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/` 这种。

主要讲下四种场景方式： - 纯dashboard http和https不惨合外部证书 - openssl 证书给dashboard当https - 个人向域名使用https小绿锁 - ingress tls 代理http[s]的dashboard 以及最后讲解的如何定义带权限的token去利用token登陆dashboard。

先要理解的一点就是JWT（JSON Web Tokens）思想，k8s的很多addon都是pod形式跑的，addon的pod都是要连接kube-apiserver来操作集群来减少运维的工作量和提供方便。addon都是pod，pod操作和查看集群信息需要鉴权。

为此k8s使用了RBAC的思想（RBAC思想不是k8s独有的），资源对象和对声明的资源对象的操作权限组合最终落实到 `ServiceAccount` 上，而每个名为 `name` 的sa会关联着一个名为 `name-token-xxxxx` 的secret。

可以通过kubectl describe命令或者api看这个secret实际上就是个token和一个ca.crt。下列命令列出缺省sa default的token和整个集群的ca.crt（jsonpath打印的时候敏感信息是base64编码需要自己解码）：

```
kubectl get secret -o jsonpath='{range .items[?(@.metadata.annotations.kubernetes\.io/service-account
token: ZXlKaGJHY2lPaUpGVXpVeE1pSXNJbXR.............

ca.crt: LS0tLS1CRUdJTiBDRVJUS.......................
```

每个pod都会被kubelet挂载pod声明的ServiceAccount关联的secret的里的ca.crt和token到容器里路径 `/var/run/secrets/kubernetes.io/serviceaccount` 。

部署的yaml的话不推荐直接使用官方的yaml，我们需要看场景修改或者删减一些东西，这里先放下官方的yaml链接，后面以文件讲解。同时也推荐把本文看完了后再开始部署。

https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml

全文我基本都是用的 hostNetwork 和 nodeName 固定在一台上，如果对k8s的几种svc和 hostNetwork 以及 hostPort 以及 Ingress 熟悉的话可以自己决定暴露方式。 这里我是使用的下面这种方式暴露出去，也就意味着我们不需要svc可以删掉官方yaml里最后那段 Dashboard Service ，访问的话用node的ip带上端口访问即可，改成大概下面这样：

```
...
    spec:
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet
      containers:
      - name: kubernetes-dashboard
        image: registry.cn-hangzhou.aliyuncs.com/google_containers/kubernetes-dashboard-amd64:v1.10
        ...
        ...
      nodeName: k8s-m1
      volumes:
```

纯dashboard

分为两种：http和开https，其中开https又分为使用自带的cert和openssl生成的。

默认自带的https

首先说说自带的https，镜像默认的 entrypoint 是这样：

```
...
            "Entrypoint": [
                "/dashboard",
                "--insecure-bind-address=0.0.0.0",
                "--bind-address=0.0.0.0"
            ],
...
```

默认定义的容器启动参数为下面这样：

```
...
        args:
          - --auto-generate-certificates
...
```

这里我宿主机的8443被占用了，我修改了下dashboard的端口，后面同理：

```
...
        ports:
        - containerPort: 5443
          protocol: TCP
        command:
          - /dashboard
          - --bind-address=0.0.0.0
        args:
          - --auto-generate-certificates
          - --port=5443
...
...
        livenessProbe:
          httpGet:
            scheme: HTTPS
            path: /
            port: 5443
...
```

`--auto-generate-certificates` 从字面意思看是dashboard自己生成https的证书，但是实际上如下面的图这个证书chrome浏览器是不认的其他浏览器不清楚，chrome打开后在网页上是没有无视警告继续的选项，可以自行去试试看，网上也没找到添加例外只找到了全局关闭非权威SSL警告。不推荐这种（或者说这种完全行不通？）

使用http

使用http我们要小心有几个坑！ 使用http我们只要使用选项 `--insecure-port` 修改成下面即可端口不一定需要和我一样，pod的健康检查记得把' `HTTPS` 改成 `HTTP` ：

```
        ports:
        - containerPort: 5443
          protocol: TCP
        command:
          - /dashboard
          - --insecure-bind-address=0.0.0.0
        args:
          - --insecure-port=5443
...
        livenessProbe:
          httpGet:
            scheme: HTTP
            path: /
            port: 5443
```

默认http是不需要登陆的，所有人进去都是可以的，我们可以注意到dashboard默认带了一个sa以及一个Role：

```
# ------------------- Dashboard Role & Role Binding ------------------- #
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: kubernetes-dashboard-minimal
  namespace: kube-system
rules:
  # Allow Dashboard to create 'kubernetes-dashboard-key-holder' secret.
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create"]
  # Allow Dashboard to create 'kubernetes-dashboard-settings' config map.
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["create"]
  # Allow Dashboard to get, update and delete Dashboard exclusive secrets.
- apiGroups: [""]
  resources: ["secrets"]
  resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs"]
  verbs: ["get", "update", "delete"]
  # Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["kubernetes-dashboard-settings"]
  verbs: ["get", "update"]
  # Allow Dashboard to get metrics from heapster.
- apiGroups: [""]
  resources: ["services"]
  resourceNames: ["heapster"]
  verbs: ["proxy"]
- apiGroups: [""]
  resources: ["services/proxy"]
  resourceNames: ["heapster", "http:heapster:", "https:heapster:"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: kubernetes-dashboard-minimal
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kubernetes-dashboard-minimal
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system
```

这样绕过登陆的话所有人都是上面的权限了，所以我们得使用选项 `--enable-insecure-login` 开启登陆界面，最终的args为下面：

```
...
        args:
          - --insecure-port=5443
          - --enable-insecure-login=true
...
```

这样下进去是强制让登录了，但是token的话（后面说这个token如何创建和获取）是无法登陆的，找到issue说通过kubectl proxy出去的http和直接暴露的http将无法登陆（但是实际上我测了下百度浏览器可以登录）。

- https://github.com/kubernetes/dashboard/issues/3216
- https://github.com/kubernetes/dashboard/issues/2735

但是也不是意味着完全不能用这种方法，可以sa kubernetes-dashboard绑定到集群角色cluster-admin然后外面套层nginx的auth到它，然后配置iptables或者网络设备ACL让dashboard只收到来源ip是nginx。

openssl生成证书给dashboard当https证书

如果我们使用openssl生成证书给dashboard使用的话，浏览器会有跳过继续前往页面的选项，能够在内网没域名下使用，我们内网给研发搭建dashboard目前就是这样使用的。具体就是openssl命令生成证书并根据证书生成tls类型的secret：

```
mkdir certs
openssl req -nodes -newkey rsa:2048 -keyout certs/dashboard.key -out certs/dashboard.csr -subj "/C=/S
openssl x509 -req -sha256 -days 10000 -in certs/dashboard.csr -signkey certs/dashboard.key -out certs
kubectl create secret generic kubernetes-dashboard-certs --from-file=certs -n kube-system
```

这里生成secret后我们先分析下官方的yaml，我们可以用命令帮助 `--help` 查看到dashboard的默认cert-dir是 `/certs` ：

```
$ docker run --rm -ti --entrypoint /dashboard registry.cn-hangzhou.aliyuncs.com/google_containers/kub
      --default-cert-dir string          Directory path containing '--tls-cert-file' and '--tls-key-f
```

而且可以注意到他有个挂载：

```
...
        volumeMounts:
        - name: kubernetes-dashboard-certs
          mountPath: /certs
...
...
      volumes:
      - name: kubernetes-dashboard-certs
        secret:
          secretName: kubernetes-dashboard-certs
...
```

上面挂载的secret也是来源于官方yaml里的secret，也就说默认情况下这个secret是给选项 `--auto-generate-certificates` 使用的：

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-certs
  namespace: kube-system
type: Opaque
```

所以我们使用openssl部署dashboard的步骤是先上面openssl生成证书然后导入生成secret，然后yaml里删掉 `Dashboard Secret` 然后修改dashboard的运行选项：

```
...
        command:
          - /dashboard
          - --bind-address=0.0.0.0
        args:
          - --auto-generate-certificates
          - --port=5443
...
```

上面为啥我开了自动生成证书的选项呢，这个选项开启时不会覆盖我们的挂载的certs文件的同时它还是全局https的开关- -坑了我好久。

个人向域名使用https小绿锁(有单点故障风险)

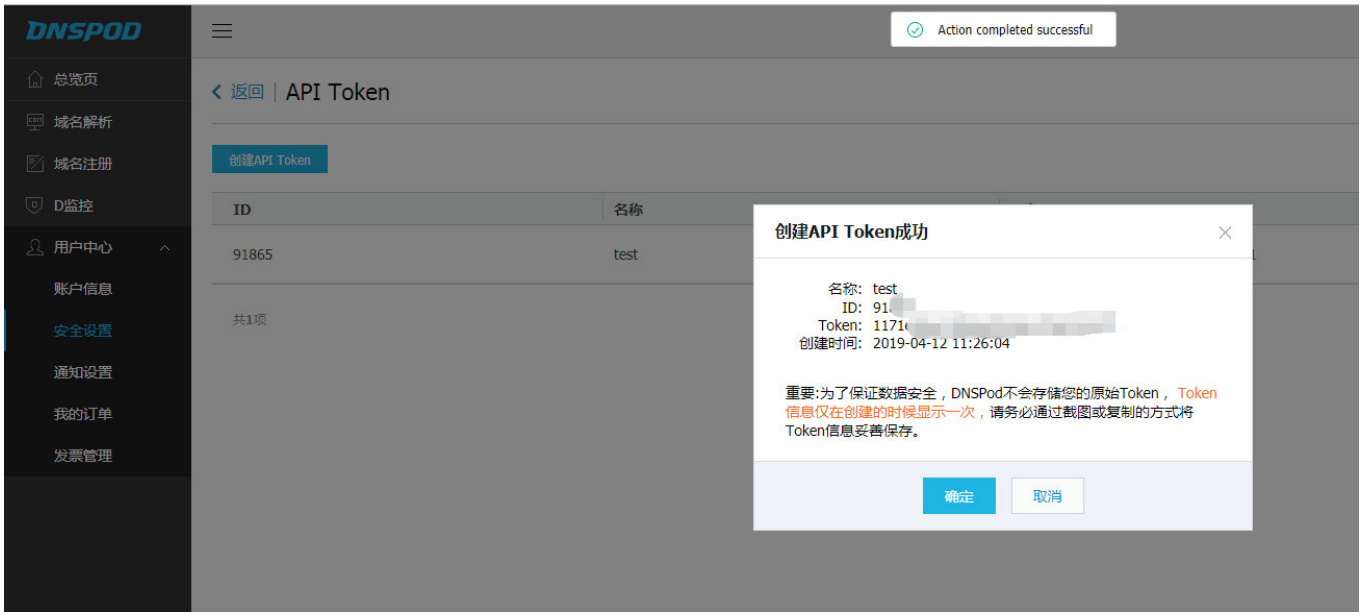
这里是通过域名+https访问，证书的话可以买的也可以免费签署的SSL证书都行。 我使用的是 `acme.sh + token` 使用 `Let’s Encrypt` 签署免费的SSL证书，也是我个人用的。另外acme申请证书的时候不一定非得通配符域名 安装acme.sh脚本：

```
curl  -s https://get.acme.sh | sh
# 设置别名方便使用命令
alias acme.sh=~/.acme.sh/acme.sh
```

DNS API ，阿里云需要设置 RAM 策略对应为 AliyunDNSFullAccess，然后在控制台获取API的token，腾讯的话确保域名解析是dnspod，其他的域名提供商请查看 https://github.com/Neilpang/acme.sh/wiki/dnsapi 。 例如阿里的话去阿里云上生成token然后下面执行：

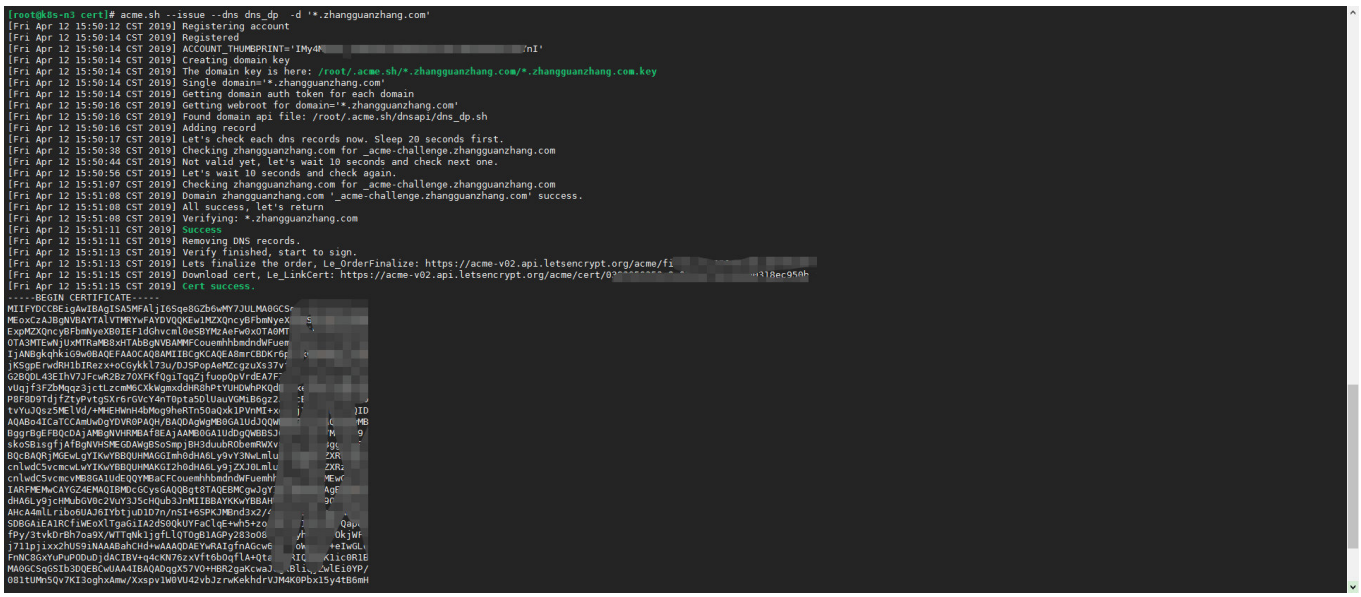
```
export Ali_Key="yourkey"
export Ali_Secret="yoursecret"
# 申请证书
acme.sh --issue --dns dns_ali -d *.k8s.youdomain.com
```

域名在腾讯云的话确保nameserver设置的是dnspod（好像默认就是这个），我们去dnspod的官网上使用登陆腾讯云的账号（例如我是qq登陆）后在开发者api里开启dnspod的api token。 注意token在创建的时候只显示一次，记得截图发给自己的时候别点错地方关了，不然得再创建个。




```
export DP_Id="1234"
export DP_Key="sADDsdasdgdsfsdfsdfasfdsfasfasfds"
acme.sh --issue --dns dns_dp -d *.zhangguanzhang.com
```

运行后会看到文件路径：



前面证书生成以后， 接下来需要把证书 copy 到真正需要用它的地方。 注意， 默认生成的证书都放在安装目录下： ~/.acme.sh/ ， 请不要直接使用此目录下的文件， 例如：不要直接让 nginx/apache 的配置文件使用这下面的文件．这里面的文件都是内部使用， 而且目录结构可能会变化。 正确的使用方法是使用 --installcert 命令，并指定目标位置， 然后证书文件会被copy到相应的位置。 COPY 证书，安装到 ~/cert 目录中，cert 证书使用的是 fullchain cert， keyfile和fullchain的证书名字随意，自己记住就行了。

```
mkdir -p ~/cert
acme.sh  --installcert  -d  *.zhangguanzhang.com   \
        --key-file           ~/cert/zhangguanzhang.com.key \
        --fullchain-file     ~/cert/zhangguanzhang.com.crt
```

然后从证书创建tls类型的secret：

```
kubectl -n kube-system create secret tls kubernetes-dashboard-certs \
  --key ~/cert/zhangguanzhang.com.key \
  --cert ~/cert/zhangguanzhang.com.crt
```

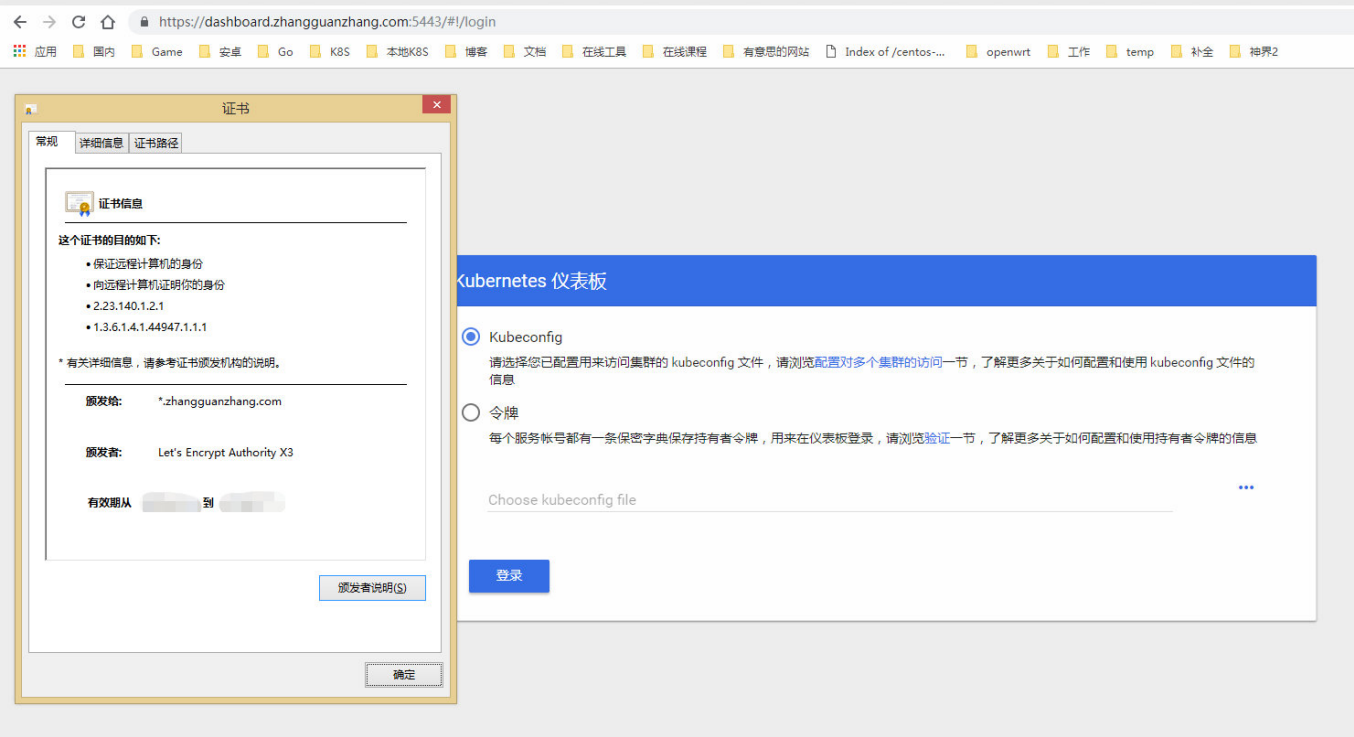
删掉官方yaml文件里的 Dashboard Secret ，我们发现tls的secret是下面俩文件名：

```
$ kubectl -n kube-system get secrets kubernetes-dashboard-certs -o yaml
apiVersion: v1
data:
  tls.crt: ....
  tls.key: ....
kind: Secret
metadata:
  name: kubernetes-dashboard-certs
  namespace: kube-system
type: kubernetes.io/tls
```

我们修改运行参数关闭insecure和指定使用证书文件，这里用5443是因为我公网ip没备案，如果ip备案了可以5443改成443（记得yaml其他地方端口也修改下）：

```
...
        command:
          - /dashboard
        args:
          - --auto-generate-certificates
          - --bind-address=0.0.0.0
          - --port=5443
          - --tls-cert-file=tls.crt
          - --tls-key-file=tls.key
...
```

创建dashboard后在云上的域名控制台设置解析过来，通过https的域名访问。



Ingress Controller使用域名证书代理dashboard

上面直接dashboard使用域名证书会有单点故障，所以实际应用我们可以用高可用的ingress nginx（详见之前的文章）来代理集群内部的dashboard 这里我使用的是 Ingress nginx 。 如果没域名的话可以使用openssl生成证书：

```
openssl req -x509 -nodes -days 10000 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=dashboa
```

从证书创建tls类型的secret：

```
kubectl -n kube-system create secret tls dashboard-tls \
  --key ~/cert/zhangguanzhang.com.key \
  --cert ~/cert/zhangguanzhang.com.crt
```

官方yaml的deploy数量改多个后直接使用创建即可，然后创建下面ingress：

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: dashboard-ingress
  namespace: kube-system
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /
    #nginx.ingress.kubernetes.io/secure-backends: "true"  该注释在0.18.0中被弃用，并在0.20.0发布后被删
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  tls:
  - hosts:
    - dashboard.example.com
    secretName: dashboard-tls
  rules:
  - host: dashboard.example.com
    http:
      paths:
      - backend:
          serviceName: kubernetes-dashboard
```

如果ingress是tls的，dashboard是http的可以去掉上面三个annotations（这种情况我未测试，有兴趣可以自己试试）。

token

dashboard登陆的话可以选择kubeconfig和token，kubeconfig一般是集群外使用的，例如管理组件之间想和apiserver tls下通信都得使用kubeconfig，里面实际上就是ca签署的客户端证书+各自CN和O签署的证书。而token里的ca.crt也是客户端证书和kubeconfig里 `client-certificate-data` 的是一样的，RBAC落实在它那个token字段。 很多addon可以看到他们的 `--help` 选项看到也支持kubeconfig的，他们的默认逻辑是没有用kubeconfig选项下起来的时候会去查看secret路径 `/var/run/secrets/kubernetes.io/serviceaccount` 获取token（也就是在pod里运行的请求逻辑），当然也并不意味着token一定在集群内用可以把secret的token获取到后制作成kubeconfig。 dashboard登陆的token如果使用管理员的话可以用rbac绑定集群管理员角色，这也是最常见的使用方法，像kubectl拥有集群管理员那样。如果对RBAC熟悉可以单独给不用部门生成不同权限的RBAC取token给人员登陆。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dashboard     # sa名字随意
  namespace: kube-system
  labels:
    k8s-app: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dashboard
  namespace: ""
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole     # 权限来源于集群角色
  name: cluster-admin   # 这个是集群管理员角色名
subjects:
  - kind: ServiceAccount
    name: dashboard    # 和上面名字一样
    namespace: kube-system
```

取它token用于登陆dashboard，你在dashboard web上操作集群的时候实际上是拿着你登陆的token的去以api调用kube-apiserver。

使用下面命令取上面创建的sa的token：

```
kubectl -n kube-system get secret -o jsonpath='{range .items[?(@.metadata.annotations.kubernetes\.io/
xxxxxxxxxxxxxx
```

最后 Let’s Encrypt 的证书是一次3个月，可以看脚本官方文档去定时获取新的证书然后导入 https://github.com/Neilpang/acme.sh/wiki/%E8%AF%B4%E6%98%8E certmanager和acme.sh一样的原理去调用api签署域名证书，有兴趣可以去试试