自动化 HTTPS¶

`Let's Encrypt` 使用 `ACME` 协议来校验域名是否真的属于你，校验成功后就可以自动颁发免费证书，证书有效期只有 90 天，在到期前需要再校验一次来实现续期，而 cert-manager 是可以自动续期的，所以事实上并不用担心证书过期的问题。目前主要有 HTTP 和 DNS 两种校验方式。

HTTP-01 校验

`HTTP-01` 的校验是通过给你域名指向的 HTTP 服务增加一个临时 location，在校验的时候 `Let's Encrypt` 会发送 http 请求到 `http://<YOUR_DOMAIN>/.well-known/acme-challenge/<TOKEN>`，其中 `YOUR_DOMAIN` 就是被校验的域名，`TOKEN` 是 cert-manager 生成的一个路径，它通过修改 Ingress 规则来增加这个临时校验路径并指向提供 TOKEN 的服务。`Let's Encrypt` 会对比 TOKEN 是否符合预期，校验成功后就会颁发证书了，不过这种方法不支持泛域名证书。

使用 HTTP 校验这种方式，首先需要将域名解析配置好，也就是需要保证 ACME 服务端可以正常访问到你的 HTTP 服务。这里我们以上面的 TODO 应用为例，我们已经将 `todo.qikqiak.com` 域名做好了正确的解析。

由于 Let's Encrypt 的生产环境有着严格的接口调用限制，所以一般我们需要先在 staging 环境测试通过后，再切换到生产环境。首先我们创建一个全局范围 staging 环境使用的 HTTP-01 校验方式的证书颁发机构：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging-http01
spec:
  acme:
    # ACME 服务端地址
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    # 用于 ACME 注册的邮箱
    email: icnych@gmail.com
    # 用于存放 ACME 帐号 private key 的 secret
    privateKeySecretRef:
      name: letsencrypt-staging-http01
    solvers:
    - http01: # ACME HTTP-01 类型
        ingress:
          class: nginx  # 指定ingress的名称
EOF
```

同样再创建一个用于生产环境使用的 ClusterIssuer 对象：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-http01
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
```

```
    email: icnych@gmail.com
    privateKeySecretRef:
      name: letsencrypt-http01
    solvers:
    - http01:
        ingress:
          class: nginx
EOF
```

创建完成后可以看到两个 ClusterIssuer 对象：

```
➜ kubectl get clusterissuer
NAME                          READY   AGE
letsencrypt-http01            True    17s
letsencrypt-staging-http01    True    3m12s
```

有了 `Issuer/ClusterIssuer` 证书颁发机构，接下来我们就可以生成免费证书了，cert-manager 给我们提供了 `Certificate` 这个用于生成证书的自定义资源对象，不过这个对象需要在一个具体的命名空间下使用，证书最终会在这个命名空间下以 Secret 的资源对象存储。我们这里是要结合 ingress-nginx 一起使用，实际上我们只需要修改 Ingress 对象，添加上 cert-manager 的相关注解即可，不需要手动创建 Certificate 对象了，修改上面的 todo 应用的 Ingress 资源对象，如下所示：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: todo
  annotations:
    kubernetes.io/ingress.class: "nginx"
    cert-manager.io/cluster-issuer: "letsencrypt-staging-http01"  # 使用哪个
issuer
spec:
  tls:
  - hosts:
    - todo.qikqiak.com     # TLS 域名
    secretName: todo-tls   # 用于存储证书的 Secret 对象名字
  rules:
  - host: todo.qikqiak.com
    http:
      paths:
      - path: /
        backend:
          serviceName: todo
          servicePort: 3000
EOF
```

更新了上面的 Ingress 对象后，正常就会自动为我们创建一个 Certificate 对象：

```
➜ kubectl get certificate
NAME       READY   SECRET     AGE
todo-tls   False   todo-tls   34s
```

在校验过程中会自动创建一个 Ingress 对象用于 ACME 服务端访问：

```
➜ kubectl get ingress
NAME                         CLASS    HOSTS                ADDRESS         PORTS
  AGE
cm-acme-http-solver-tgwlb    <none>   todo.qikqiak.com     10.151.30.11    80
   25s
my-nginx                     <none>   ngdemo.qikqiak.com   10.151.30.11    80
   23h
todo                         <none>   todo.qikqiak.com     10.151.30.11    80, 443
  33s
```

校验成功后会将证书保存到 todo-tls 的 Secret 对象中：

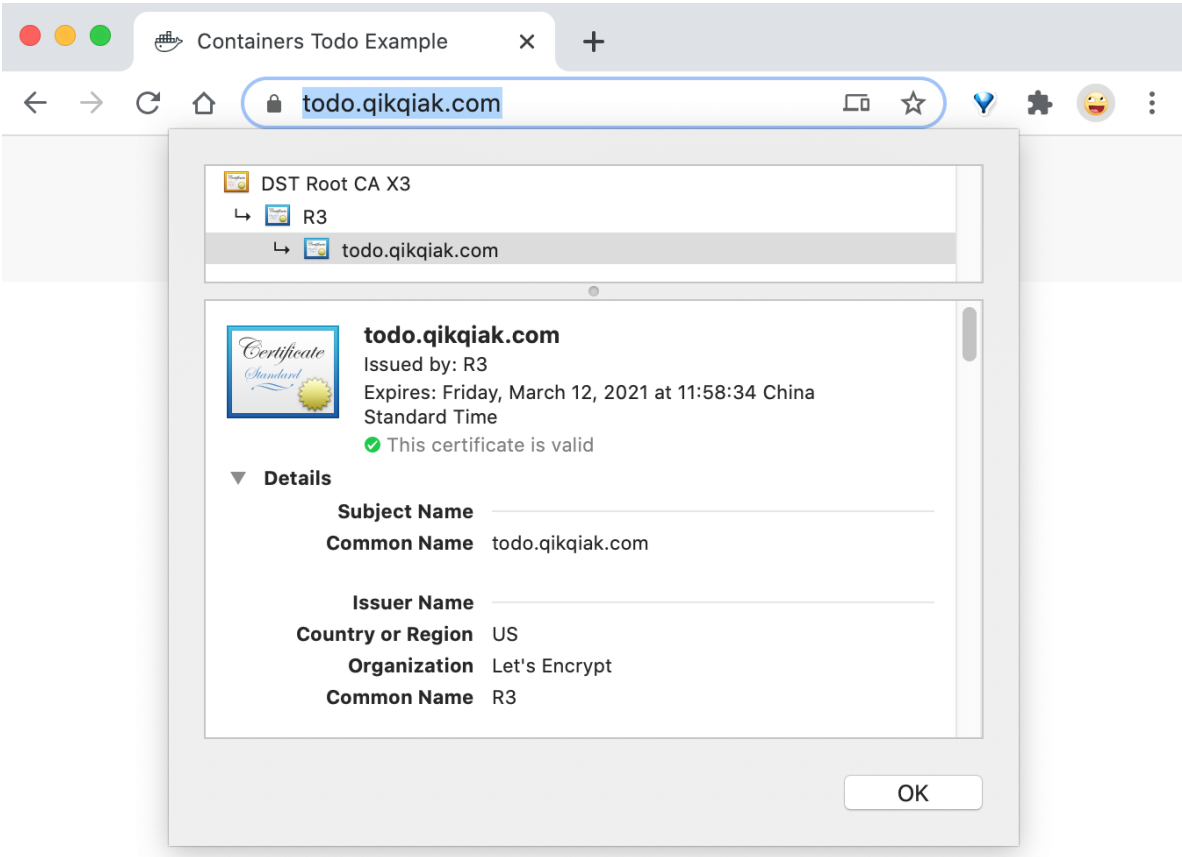
```
➜ kubectl get certificate
NAME       READY   SECRET     AGE
todo-tls   True    todo-tls   21m
➜ kubectl get secret
NAME                          TYPE                                  DATA   AGE
default-token-hpd7s           kubernetes.io/service-account-token   3      55d
todo-tls                      kubernetes.io/tls                     2      20m
➜  kubectl describe certificate todo-tls
Name:         todo-tls
Namespace:    default
......
Events:
  Type    Reason     Age   From          Message
  ----    ------     ----  ----          -------
  Normal  Issuing    22m   cert-manager  Issuing certificate as Secret does not
exist
  Normal  Generated  22m   cert-manager  Stored new private key in temporary
Secret resource "todo-tls-tr4pq"
  Normal  Requested  22m   cert-manager  Created new CertificateRequest resource
"todo-tls-2gchg"
  Normal  Issuing    21m   cert-manager  The certificate has been successfully
issued
```

证书自动获取成功后，现在就可以讲 ClusterIssuer 替换成生产环境的了：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: todo
  annotations:
    kubernetes.io/ingress.class: "nginx"
```

```
    cert-manager.io/cluster-issuer: "letsencrypt-http01"  # 使用生产环境的issuer
spec:
  tls:
  - hosts:
    - todo.qikqiak.com      # TLS 域名
    secretName: todo-tls    # 用于存储证书的 Secret 对象名字
  rules:
  - host: todo.qikqiak.com
    http:
      paths:
      - path: /
        backend:
          serviceName: todo
          servicePort: 3000
EOF
➜ kubectl get certificate
NAME       READY   SECRET     AGE
todo-tls   True    todo-tls   25m
```

校验成功后就可以自动获取真正的 HTTPS 证书了，现在在浏览器中访问 https://todo.qikqiak.com 就可以看到证书是有效的了。



DNS-01 校验

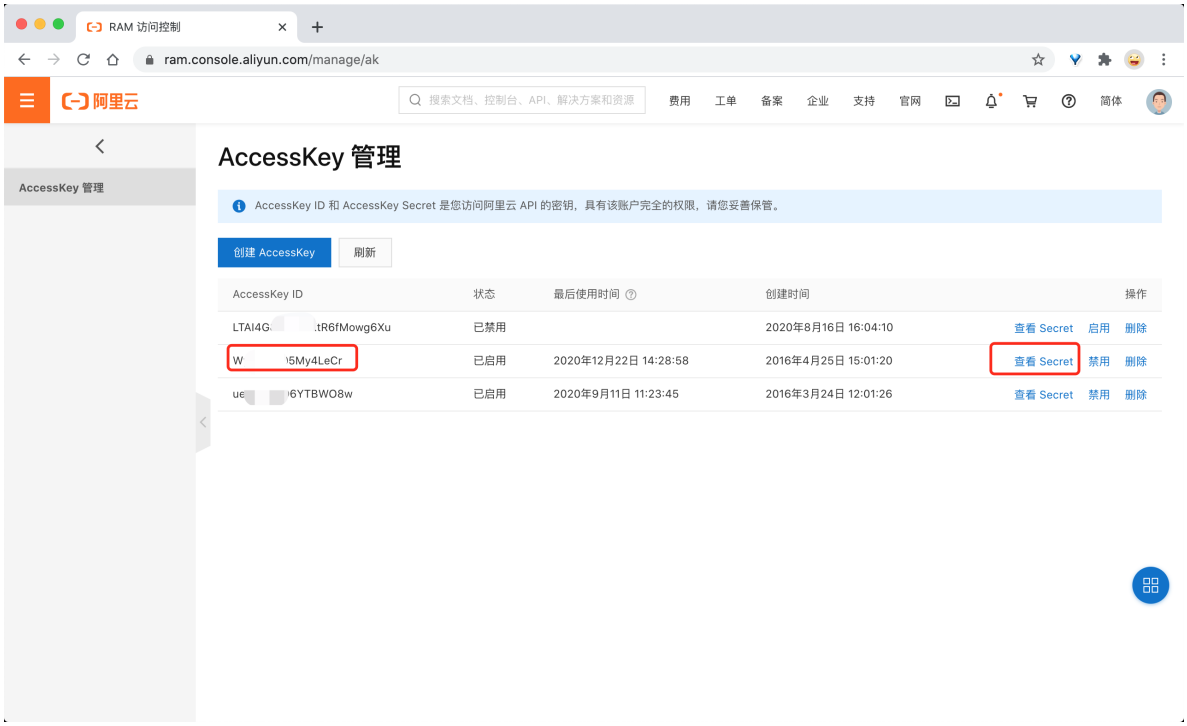
`DNS-01` 的校验是通过 DNS 提供商的 API 拿到你的 DNS 控制权限， 在 `Let's Encrypt` 为 cert-manager 提供 TOKEN 后，cert-manager 将创建从该 TOKEN 和你的帐户密钥派生的 `TXT` 记录，并将该记录放在 `_acme-challenge.<YOUR_DOMAIN>`。然后 `Let's Encrypt` 将向 DNS 系统查询该记录，如果找到匹配项，就可以颁发证书，这种方法是支持泛域名证书的。

DNS-01 支持多种不同的服务提供商，直接在 Issuer 或者 ClusterIssuer 中可以直接配置，对于一些不支持的 DNS 服务提供商可以使用外部 webhook 来提供支持，比如阿里云的 DNS 解析默认情况下是不支持的，我们可以使用 https://github.com/pragkent/alidns-webhook 这个 webhook 来提供支持。

首先使用如下命令安装 alidns-webhook：

```
# Install alidns-webhook to cert-manager namespace.
➜ kubectl apply -f https://raw.githubusercontent.com/pragkent/alidns-webhook/master/deploy/bundle.yaml
```

接着创建一个包含访问阿里云 DNS 认证密钥信息的 Secret 对象，对应的 `accessk-key` 和 `secret-key` 可以前往阿里云网站 https://ram.console.aliyun.com/manage/ak 获取：



默认可以查看到 AccessKey ID 的值，点击后面的`查看 Secret`即可看到对应的 AccessSecret，创建如下所示的 Secret 资源对象：

```
➜ kubectl create secret generic alidns-secret --from-literal=access-key=YOUR_ACCESS_KEY --from-literal=secret-key=YOUR_SECRET_KEY -n cert-manager
```

接下来同样首先创建一个 staging 环境的 DNS 类型的证书机构资源对象：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging-dns01
spec:
  acme:
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    email: icnych@gmail.com
    privateKeySecretRef:
      name: letsencrypt-staging-dns01
```

```
    solvers:
    - dns01:   # ACME DNS-01 类型
        webhook:
          groupName: acme.yourcompany.com
          solverName: alidns
          config:
            region: ""
            accessKeySecretRef:  # 引用 ak
              name: alidns-secret
              key: access-key
            secretKeySecretRef:  # 引用 sk
              name: alidns-secret
              key: secret-key
EOF
```

再创建一个用于线上环境正式使用的 DNS 类型的 ClusterIssuer 对象：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-dns01
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: icnych@gmail.com
    privateKeySecretRef:
      name: letsencrypt-dns01
    solvers:
    - dns01:   # ACME DNS-01 类型
        webhook:
          groupName: acme.yourcompany.com
          solverName: alidns
          config:
            region: ""
            accessKeySecretRef:  # 引用 ak
              name: alidns-secret
              key: access-key
            secretKeySecretRef:  # 引用 sk
              name: alidns-secret
              key: secret-key
EOF
➜ kubectl get clusterissuer
NAME                        READY   AGE
letsencrypt-dns01           True    7s
letsencrypt-staging-dns01   True    90s
```

接下来我们就可以使用上面的 ClusterIssuer 对象来或者证书数据了，创建如下所示的 Certificate 资源对象：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
```

```
metadata:
  name: qikqiak-com-cert
spec:
  secretName: qikqiak-com-tls
  commonName: "*.qikqiak.com"
  dnsNames:
  - qikqiak.com
  - "*.qikqiak.com"
  issuerRef:
    name: letsencrypt-staging-dns01
    kind: ClusterIssuer
EOF
```

这里我们为 `*.qikqiak.com` 这个泛域名来获取证书，创建完成后正常就可以获取到证书了：

```
➜ kubectl get certificate
NAME               READY   SECRET            AGE
qikqiak-com-cert   True    qikqiak-com-tls   3m21s
```

生成的证书数据也被存储到了名为 qikqiak-com-tls 的 Secret 对象中，接下来将上面的 Certficate 资源对象更换成正式环境的 ClusterIssuer 证书颁发机构：

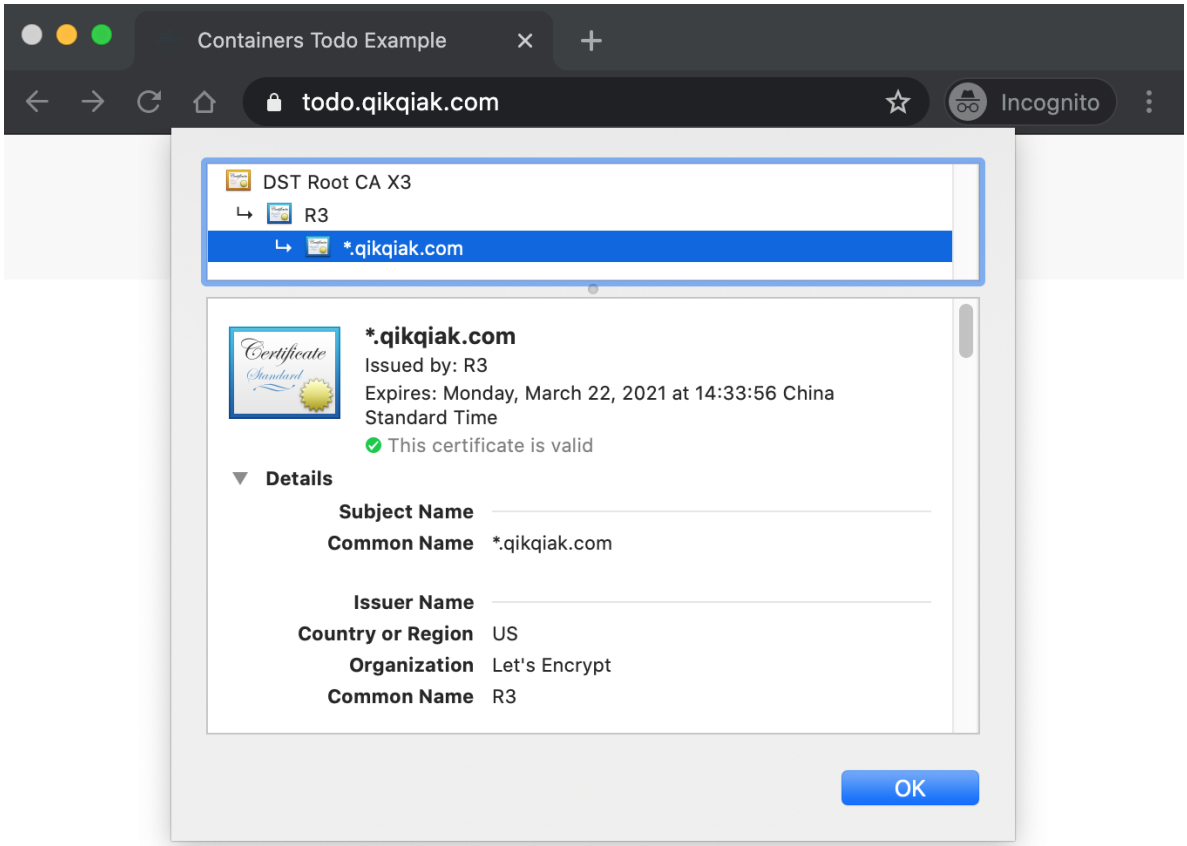
```
➜ cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: qikqiak-com-cert
spec:
  secretName: qikqiak-com-tls
  commonName: "*.qikqiak.com"
  dnsNames:
  - qikqiak.com
  - "*.qikqiak.com"
  issuerRef:
    name: letsencrypt-dns01  # 替换成正式的 clusterissuer 对象
    kind: ClusterIssuer
EOF
➜ kubectl get certificate
NAME               READY   SECRET            AGE
qikqiak-com-cert   True    qikqiak-com-tls   10m
➜ kubectl get certificaterequest
NAME                     READY   AGE
qikqiak-com-cert-stct8   True    2m54s
➜ kubectl get order
NAME                                  STATE   AGE
qikqiak-com-cert-stct8-3109450950   valid   3m38s
➜ kubectl describe order qikqiak-com-cert-stct8-3109450950

Name:         qikqiak-com-cert-stct8-3109450950
Namespace:    default
Labels:       <none>
......
Spec:
```

```
  Common Name:  *.qikqiak.com
  Dns Names:
    qikqiak.com
    *.qikqiak.com
  Issuer Ref:
    Kind:   ClusterIssuer
    Name:   letsencrypt-dns01
  ......
  Finalize URL:     https://acme-
v02.api.letsencrypt.org/acme/finalize/107093987/6878258208
  State:            valid
  URL:              https://acme-
v02.api.letsencrypt.org/acme/order/107093987/6878258208
Events:
  Type    Reason    Age    From          Message
  ----    ------    ----   ----          -------
  Normal  Created   3m45s  cert-manager  Created Challenge resource "qikqiak-
com-cert-stct8-3109450950-1046480550" for domain "qikqiak.com"
  Normal  Created   3m45s  cert-manager  Created Challenge resource "qikqiak-
com-cert-stct8-3109450950-4189768354" for domain "qikqiak.com"
  Normal  Complete  80s    cert-manager  Order completed successfully
```

更新后生成正式环境的通配符证书，然后我们就可以直接在 Ingress 资源对象中使用上面的 Secret 对象了：

```
➜ cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: todo
  annotations:
    kubernetes.io/ingress.class: "nginx"
spec:
  tls:
  - hosts:
    - "*.qikqiak.com"     # 泛域名
    secretName: qikqiak-com-tls   # 用于存储通配符证书的 Secret 对象名字
  rules:
  - host: todo.qikqiak.com
    http:
      paths:
      - path: /
        backend:
          serviceName: todo
          servicePort: 3000
EOF
```

现在当我们去访问 `https://todo.qikqiak.com` 的时候就有对应的受浏览器信任的证书了，而且这个证书是支持泛域名的，对 `qikqiak.com` 的二级域名都适用的。


Containers Todo Example
todo.qikqiak.com
Incognito
DST Root CA X3
R3
*.qikqiak.com
*.qikqiak.com
Issued by: R3
Expires: Monday, March 22, 2021 at 14:33:56 China Standard Time
This certificate is valid
Details
Subject Name
Common Name *.qikqiak.com
Issuer Name
Country or Region US
Organization Let's Encrypt
Common Name R3
OK