Red Hat OpenShift Service on AWS 4

Application development

Configuring Red Hat OpenShift Service on AWS for your applications
Configuring Red Hat OpenShift Service on AWS for your applications
Abstract

This document provides information about configuring Red Hat OpenShift Service on AWS (ROSA) for your application deployments. This includes setting up custom wildcard domains.
Table of Contents

CHAPTER 1. DEPLOYMENTS .............................................................. 3

1.1. CUSTOM DOMAINS FOR APPLICATIONS ................................. 3
   1.1.1. Configuring custom domains for applications ................ 3
   1.1.2. Renewing a certificate for custom domains .................... 5
1.1. CUSTOM DOMAINS FOR APPLICATIONS

NOTE

Starting with Red Hat OpenShift Service on AWS 4.14, the Custom Domain Operator is deprecated. To manage Ingress in Red Hat OpenShift Service on AWS 4.14, use the Ingress Operator. The functionality is unchanged for Red Hat OpenShift Service on AWS 4.13 and earlier versions.

You can configure a custom domain for your applications. Custom domains are specific wildcard domains that can be used with Red Hat OpenShift Service on AWS applications.

1.1.1. Configuring custom domains for applications

The top-level domains (TLDs) are owned by the customer that is operating the Red Hat OpenShift Service on AWS cluster. The Custom Domains Operator sets up a new ingress controller with a custom certificate as a second day operation. The public DNS record for this ingress controller can then be used by an external DNS to create a wildcard CNAME record for use with a custom domain.

NOTE

Custom API domains are not supported because Red Hat controls the API domain. However, customers can change their application domains. For private custom domains with a private IngressController, set .spec.scope to Internal in the CustomDomain CR.

Prerequisites

- A user account with dedicated-admin privileges
- A unique domain or wildcard domain, such as *.apps.<company_name>.io
- A custom certificate or wildcard custom certificate, such as CN=*.apps.<company_name>.io
- Access to a cluster with the latest version of the oc CLI installed

IMPORTANT

Do not use the reserved names default or apps*, such as apps or apps2, in the metadata/name: section of the CustomDomain CR.

Procedure

1. Create a new TLS secret from a private key and a public certificate, where fullchain.pem and privkey.pem are your public or private wildcard certificates.

   Example

   $ oc create secret tls <name>-tls --cert=fullchain.pem --key=privkey.pem -n <my_project>

2. Create a new CustomDomain custom resource (CR):
Example `<company_name>-custom-domain.yaml`

```yaml
apiVersion: managed.openshift.io/v1alpha1
kind: CustomDomain
metadata:
  name: `<company_name>`
spec:
  domain: apps.<company_name>.io  
  scope: External
  loadBalancerType: Classic
  certificate:
    name: `<name>-tls`
    namespace: `<my_project>`
  routeSelector:
    matchLabels:
      route: acme
  namespaceSelector:
    matchLabels:
      type: sharded
```

1. The custom domain.
2. The type of load balancer for your custom domain. This type can be the default `classic` or `NLB` if you use a network load balancer.
3. The secret created in the previous step.
4. Optional: Filters the set of routes serviced by the CustomDomain ingress. If no value is provided, the default is no filtering.
5. Optional: Filters the set of namespaces serviced by the CustomDomain ingress. If no value is provided, the default is no filtering.

3. Apply the CR:

   ```bash
   $ oc apply -f `<company_name>-custom-domain.yaml`
   ```

4. Get the status of your newly created CR:

   ```bash
   $ oc get customdomains
   ```

   **Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ENDPOINT</th>
<th>DOMAIN</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;company_name&gt;</code></td>
<td>xxrywp.&lt;company_name&gt;.cluster-01.opln.s1.openshiftapps.com</td>
<td>*.apps.&lt;company_name&gt;.io</td>
<td>Ready</td>
</tr>
</tbody>
</table>

5. Using the endpoint value, add a new wildcard CNAME recordset to your managed DNS provider, such as Route53.

   **Example**
6. Create a new application and expose it:

**Example**

```
$ oc new-app --docker-image=docker.io/openshift/hello-openshift -n my-project

$ oc create route <route_name> --service=hello-openshift hello-openshift-tls --hostname hello-openshift-tls-my-project.apps.<company_name>.io -n my-project

$ oc get route -n my-project

$ oc create secret tls <secret-new> --cert=fullchain.pem --key=privkey.pem -n <my_project>

$ oc patch customdomain <company_name> --type='merge' -p '{"spec":{"certificate": {"name":"<secret-new>"}}}'

$ oc delete secret <secret-old> -n <my_project>
```

**Troubleshooting**

- Error creating TLS secret
- Troubleshooting: CustomDomain in NotReady state

### 1.1.2. Renewing a certificate for custom domains

You can renew certificates with the Custom Domains Operator (CDO) by using the `oc` CLI tool.

**Prerequisites**

- You have the latest version `oc` CLI tool installed.

**Procedure**

1. Create new secret

```
$ oc create secret tls <secret-new> --cert=fullchain.pem --key=privkey.pem -n <my_project>
```

2. Patch CustomDomain CR

```
$ oc patch customdomain <company_name> --type='merge' -p '{"spec":{"certificate": {"name":"<secret-new>"}}}'
```

3. Delete old secret

```
$ oc delete secret <secret-old> -n <my_project>
```

**Troubleshooting**

- Error creating TLS secret