Advanced Configuration Options For OpenShift

NetApp Solutions

NetApp
August 18, 2021

Table of Contents

Advanced Configuration Options For OpenShift

Exploring load balancer options: Red Hat OpenShift with NetApp

In most of the cases, Red Hat OpenShift makes applications available to the outside world through routes. A service is exposed by giving it an externally reachable hostname. The defined route and the endpoints identified by its service can be consumed by an OpenShift router to provide this named connectivity to external clients.

However in some cases, applications require the deployment and configuration of customized load balancers to expose the appropriate services. One example of this is NetApp Astra Control Center. To meet this need, we have evaluated a number of custom load balancer options. Their installation and configuration are described in this section.

The following pages have additional information about load balancer options validated in the Red Hat OpenShift with NetApp solution:

- MetalLB

Next: Solution validation/use cases: Red Hat OpenShift with NetApp.

Installing MetalLB load balancers: Red Hat OpenShift with NetApp

This page lists the installation and configuration instructions for the MetalLB load balancer.

MetalLB is a self-hosted network load-balancer installed on your OpenShift cluster that allows the creation of OpenShift services of type load balancer in clusters that don't run on a cloud provider. The two main features of MetalLB that work together to support LoadBalancer services are address allocation and external announcement.

MetalLB configuration options

Based on how MetalLB announces the IP address assigned to LoadBalancer services outside of the OpenShift cluster, it operates in two modes:

- **Layer 2 mode.** In this mode, one node in the OpenShift cluster takes ownership of the service and responds to ARP requests for that IP to make it reachable outside of the OpenShift cluster. Because only the node advertises the IP, it has a bandwidth bottleneck and slow failover limitations. For more information, see the documentation here.
- **BGP mode.** In this mode, all nodes in the OpenShift cluster establish BGP peering sessions with a router and advertise the routes to forward traffic to the service IPs. The pre-requisite for this is to integrate MetalLB with a router in that network. Owing to the hashing mechanism in BGP, it has certain limitation when IP-to-Node mapping for a service changes. For more information, refer to the documentation here.

For the purpose of this document, we are configuring MetalLB in layer 2 mode.

Installing The MetalLB Load Balancer

1. Download the MetalLB resources.

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. Edit file `metallb.yaml` and remove `spec.template.spec.securityContext` from controller Deployment and the speaker DaemonSet.

   Lines to be deleted:

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. Create the `metallb-system` namespace.

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. Create the MetalLB CR.

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. Before configuring the MetalLB speaker, grant the speaker DaemonSet elevated privileges so that it can perform the networking configuration required to make the load balancers work.

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. Configure MetalLB by creating a `ConfigMap` in the `metallb-system` namespace.

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: metallb-config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/metallb-config created
```

7. Now when loadbalancer services are created, MetalLB assigns an externalIP to the services and advertises the IP address by responding to ARP requests.

> If you wish to configure MetalLB in BGP mode, skip step 6 above and follow the procedure in the MetalLB documentation here.

Next: Solution validation/use cases: Red Hat OpenShift with NetApp.

Creating Private Image Registries: Red Hat OpenShift with NetApp

For most deployments of Red Hat OpenShift, using a public registry like Quay.io or DockerHub meets most customer's needs. However there are times when a customer may want to host their own private or customized images.

This procedure documents creating a private image registry which is backed by a persistent volume provided by Astra Trident and NetApp ONTAP.

> Astra Control Center requires a registry to host the images the Astra containers require. The following section describes the steps to setup a private registry on Red Hat OpenShift cluster and pushing the images required to support the installation of Astra Control Center.

Creating A Private Image Registry

1. Edit the imageregistry operator, enter the below storage parameters to `spec` section

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

2. Then enter the following parameters to `spec` section for creating a OpenShift route with a custom hostname, save and exit

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

> The above route config is used when you want a custom hostname for your route. If you want OpenShift to create a route with default hostname, you can just add the following parameters to `spec` section – `defaultRoute: true`

Custom TLS Certificates

When you are using custom hostname for the route, by default, it uses the default TLS configuration of OpenShift Ingress operator. However, you can add a custom TLS configuration to the route. To do so, following the below steps –

a. Create a secret with the route's TLS certificates and key –

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. Edit the imageregistry operator and add the following parameters to the `spec` section –

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

3. Next step is to edit the imageregistry operator again and change the management state of the operator to `Managed` state, save and exit –

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

4. If all the pre-requisites are satisfied, it should start creating PVCs, pods and services for the private image registry. In a few minutes, the registry should be up.

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                   READY   STATUS
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                      0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                      0/1     Completed
0          33h
```

```
pod/image-pruner-1627430400-rv5nt                          0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                         1/1     Running
0          76m
pod/node-ca-bwb5r                                          1/1     Running
0          90d
pod/node-ca-f8w54                                          1/1     Running
0          90d
pod/node-ca-gjx7h                                          1/1     Running
0          90d
pod/node-ca-lcx4k                                          1/1     Running
0          33d
pod/node-ca-v7zmx                                          1/1     Running
0          7d21h
pod/node-ca-xpppp                                          1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                     DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                             READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h

NAME                                                          DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
1       90d
replicaset.apps/image-registry-6758b547f                     1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                    0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                    0         0
0       80m
```

```
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                 COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                          SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                      HOST/PORT
PATH   SERVICES         PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

5. If you are using the default TLS certificates of Ingress operator OpenShift registry route, you can fetch the TLS certificates using the below command.

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

6. To allow OpenShift nodes to access and pull the images from the registry, you need to add the certificates to the docker client on the OpenShift nodes. Create a configmap in `openshift-config` namespace using the TLS certificates and patch it to the cluster image config to make the certificate trusted.

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

7. OpenShift internal registry is controlled by authentication. All the OpenShift users can access the OpenShift registry, but the operations that the logged in user can perform depends on the user permissions.

    a. To allow a user/group of users to pull images from the registry, the user/s must have registry-viewer role assigned.

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

b. To allow a user/group of users to write or push images, the user/s must have registry-editor role assigned.

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

8. For OpenShift nodes to access the registry and push/pull the images, you will need to configure a pull secret.

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server= astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

9. This pull secret can then be patched to serviceaccounts or be referenced in the corresponding pod definition.

a. To patch it to service accounts

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. To reference the pull secret in Pod definition, add the following parameter to the ‘spec’ section.

```
imagePullSecrets:
  - name: astra-registry-credentials
```

10. To push/pull an image from workstations apart from OpenShift node.

a. Add the TLS certificates to the docker client.

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

b. Log into OpenShift using oc login command.

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

c. Log into the registry using OpenShift user credentials via podman/docker command.

podman

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

docker

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

d. Push/pull the images.

podman

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

docker

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

Next: Solution Validation/Use Cases: Red Hat OpenShift with NetApp.