Red Hat OpenShift Service on AWS 4

Storage and registry

Configuring storage for Red Hat OpenShift Service on AWS clusters
Configuring storage for Red Hat OpenShift Service on AWS clusters
Abstract

This document provides information about setting up storage for Red Hat OpenShift Service on AWS (ROSA) clusters.
# Table of Contents

CHAPTER 1. PERSISTENT STORAGE ................................................................. 3

1.1. PERSISTENT STORAGE USING AWS ELASTIC BLOCK STORE (EBS) 3
   1.1.1. Format of persistent volumes 3
   1.1.2. Capacity of EBS volumes on a node 3
   1.1.3. Creating a persistent volume claim 4

1.2. SETTING UP AWS EFS FOR RED HAT OPENSIGHT SERVICE ON AWS 4
   1.2.1. Prerequisites 5
   1.2.2. Configuring the AWS account 5
   1.2.3. Installing the EFS Operator 6
   1.2.4. Creating SharedVolume resources using the console 6
   1.2.5. Creating SharedVolume resources using the CLI 7
   1.2.6. Connecting pods 7
   1.2.7. Uninstalling the EFS Operator 8
CHAPTER 1. PERSISTENT STORAGE

1.1. PERSISTENT STORAGE USING AWS ELASTIC BLOCK STORE (EBS)

Red Hat OpenShift Service on AWS (ROSA) clusters are prebuilt with two storage classes that use AWS Elastic Block Store (EBS) volumes. These storage classes are ready to use and some familiarity with Kubernetes and AWS is assumed.

Following are the two prebuilt storage classes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Provisioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>gp2</td>
<td>kubernetes.io/aws-ebs</td>
</tr>
<tr>
<td>gp2-csi</td>
<td>ebs.csi.aws.com</td>
</tr>
</tbody>
</table>

The gp2 storage class is set as default; however, you can select either one as the default storage class.

The Kubernetes persistent volume framework enables administrators to provision a cluster with persistent storage and gives users a way to request those resources without having any knowledge of the underlying infrastructure. You can dynamically provision AWS EBS volumes. Persistent volumes are not bound to a single project or namespace; therefore, the volumes can be shared across ROSA clusters. Persistent volume claims are specific to a project or namespace and can be requested by users.

**IMPORTANT**

- ROSA defaults to using an in-tree, or non-Container Storage Interface (CSI), plug-in to provision AWS EBS storage. In future ROSA versions, volumes provisioned using existing in-tree plug-ins are planned for migration to their equivalent CSI driver. After full migration, the in-tree plug-ins are planned to be removed from the future versions of ROSA.

- High-availability of storage in the infrastructure is left to the underlying storage provider.

1.1.1. Format of persistent volumes

Before a ROSA cluster mounts the volume and passes it to a container, the cluster checks that the volume contains a file system as specified by the `fsType` parameter in the persistent volume definition. If the device is not formatted with a file system, all data from the device is erased and the device is automatically formatted with the given file system. This verification enables you to use unformatted AWS volumes as persistent volumes, as the ROSA cluster formats the AWS volumes before the first use.

1.1.2. Capacity of EBS volumes on a node

By default, a ROSA cluster supports a maximum of 39 EBS volumes attached to one node. This limit is consistent with the AWS volume limits. The volume limit depends on the instance type.
IMPORTANT

You must use either in-tree or CSI volumes and their respective storage classes, but never both volume types at the same time. The maximum attached EBS volume number is counted separately for in-tree and CSI volumes, so you could have up to 39 EBS volumes of each type.

For information about accessing additional storage options, such as volume snapshots, that are not possible with in-tree volume plug-ins, see Elastic Block Store CSI Driver Operator.

1.1.3. Creating a persistent volume claim

Prerequisites

Storage must exist in the underlying infrastructure before it can be mounted as a volume in the ROSA cluster.

Procedure

1. In the OpenShift Cluster console, click Storage → Persistent Volume Claims.
2. In the persistent volume claims overview, click Create Persistent Volume Claim.
3. Define the desired options on the page that appears.
   a. Select the previously created storage class from the drop-down menu.
   b. Enter a unique name for the storage claim.
   c. Select the access mode. This selection determines the read and write access for the storage claim.
   d. Define the size of the storage claim.
4. Click Create to create the persistent volume claim and generate a persistent volume.

1.2. SETTING UP AWS EFS FOR RED HAT OPENSIGHT SERVICE ON AWS

The Amazon Web Services Elastic File System (AWS EFS) is a Network File System (NFS) that can be provisioned on Red Hat OpenShift Service on AWS clusters. AWS also provides and supports a CSI EFS Driver to be used with Kubernetes that allows Kubernetes workloads to leverage this shared file storage.

This document describes the basic steps needed to set up your AWS account to prepare EFS to be used by Red Hat OpenShift Service on AWS. For more information about AWS EFS, see the AWS EFS documentation.

IMPORTANT

Red Hat does not provide official support for this feature, including backup and recovery. The customer is responsible for backing up the EFS data and recovering it in the event of an outage or data loss.

The high-level process to enable EFS on a cluster is:
1. Create an AWS EFS in the AWS account used by the cluster.
2. Install the AWS EFS Operator from OperatorHub.
3. Create `SharedVolume` custom resources.
4. Use the generated persistent volume claims in pod `spec.volumes`.

### 1.2.1. Prerequisites
- A Red Hat OpenShift Service on AWS cluster
- Administrator access to the AWS account of that cluster

### 1.2.2. Configuring the AWS account
Set up your AWS account to prepare AWS EFS for use by Red Hat OpenShift Service on AWS.

**Procedure**

1. Log in to the [AWS EC2 Console](https://aws.amazon.com/ec2/).
2. Select the region that matches the cluster region.
3. Filter only worker EC2 instances, and select an instance. Note the VPC ID and security group ID. These values are required later in the process.
4. Click the `Security` tab, and click the Security Group Name.
5. From the `Actions` dropdown menu, click **Edit Inbound Rules**. Scroll to the bottom, and click **Add Rule**.
6. Add an NFS rule that allows NFS traffic from the VPC private CIDR.
7. Open the [Amazon EFS page](https://aws.amazon.com/efs/). To create the EFS, click **Create file system**.
8. Click **Customize** and proceed through the wizard.
   - In **Step 2:** configure the network access:
     - i. Click the VPC of the cluster that you noted previously.
     - ii. Ensure that the private subnets are selected.
     - iii. Select the Security Group Name that you noted previously for the EC2 worker instances.
     - iv. Click **Next**.
   - In **Step 3:** configure the client access:
     - i. Click **Add access point**
     - ii. Enter a unique Path such as `/access_point_1`. 
iii. Configure the Owner fields with ownership or permissions that allow write access for your worker pods. For example, if your worker pods run with group ID 100, you can set that ID as your Owner Group ID and ensure the permissions include g+rwx.

9. Continue through the wizard steps, and click Create File System.

10. After the file system is created:
   a. Note the file system ID for later use.
   b. Click Manage client access and note the access point ID.

You can add more NFS rules, using steps 5-10, to create separate shared data stores. In each case, make note of the corresponding file system ID and access point ID.

1.2.3. Installing the EFS Operator

Procedure

1. Log in to the OpenShift Web UI for your cluster.

2. Click Operators → OperatorHub.

3. Search for and select the AWS EFS Operator. Click Install.

4. Accept the default settings, and click Subscribe.

1.2.4. Creating SharedVolume resources using the console

You must create one SharedVolume resource per file system:access point pair in each project from which you want pods to access it.

Procedure

1. In the OpenShift web console, create and navigate to a project.

2. Click Operators → Installed Operators. Find the entry for AWS EFS Operator, and click SharedVolume under Provided APIs.

3. Click Create SharedVolume.

4. Edit the sample YAML:
   a. Type a suitable value for name.
   b. Replace the values of accessPointID and fileSystemID with the values from the EFS resources you created earlier.

```yaml
apiVersion: aws-efs.managed.openshift.io/v1alpha1
kind: SharedVolume
metadata:
  name: sv1
  namespace: efsop2
spec:
  accessPointID: fsap-0123456789abcdef
  fileSystemID: fs-0123cdef
```
5. Click **Create**.
The **SharedVolume** resource is created, and triggers the AWS EFS Operator to generate and associate a PersistentVolume:PersistentVolumeClaim pair with the specified EFS access point.

6. To verify that the persistent volume claim (PVC) exists and is bound, click **Storage → Persistent Volume Claims**
The PVC name is **pvc-{shared_volume_name}**. The associated PV name is **pv-{project_name}-{shared_volume_name}**.

### 1.2.5. Creating SharedVolume resources using the CLI

You must create one **SharedVolume** resource per file system:access point pair in each project from which you want pods to access it. You can create a SharedVolume manually by entering YAML or JSON definitions, or by dragging and dropping a file into an editor.

**Procedure**

1. Using the **oc** CLI, create the YAML file using the **accessPointID** and **fileSystemID** values from the EFS resources you created earlier.

   ```yaml
   apiVersion: aws-efs.managed.openshift.io/v1alpha1
   kind: SharedVolume
   metadata:
     name: sv1
     namespace: efsop2
   spec:
     accessPointID: fsap-0123456789abcdef
     fileSystemID: fs-0123cdef
   ```

2. Apply the file to the cluster using the following command:

   ```bash
   $ oc apply -f <filename>.yaml
   ```

   The **SharedVolume** resource is created, and triggers the AWS EFS Operator to generate and associate a PersistentVolume:PersistentVolumeClaim pair with the specified EFS access point.

3. To verify that the PVC exists and is bound, navigate to **Storage > Persistent Volume Claims**
The PVC name is **pvc-{shared_volume_name}**. The associated PV name is **pv-{project_name}-{shared_volume_name}**.

### 1.2.6. Connecting pods

The persistent volume claim (PVC) that was created in your project is ready for use. You can create a sample pod to test this PVC.

**Procedure**

1. Create and navigate to a project.

2. Click **Workloads → Pods → Create Pod**.

3. Enter the YAML information. Use the name of your **PersistentVolumeClaim** object under **.spec.volumes[].persistentVolumeClaim.claimName**.
Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-efs
spec:
  volumes:
    - name: efs-storage-vol
      persistentVolumeClaim:
        claimName: pvc-sv1
  containers:
    - name: test-efs
      image: centos:latest
      command: ["/bin/bash", "-c", "--"]
      args: ["while true; do touch /mnt/efs-data/verify-efs && echo 'hello efs' && sleep 30; done;"
      volumeMounts:
        - mountPath: "/mnt/efs-data"
          name: efs-storage-vol
```

4. After the pods are created, click Workloads → Pods → Logs to verify the pod logs.

1.2.7. Uninstalling the EFS Operator

Procedure

To remove the Operator from your cluster:

1. Delete all of the workloads using the persistent volume claims that were generated by the Operator.

2. Delete all of the shared volumes from all of the namespaces. The Operator automatically removes the associated persistent volumes and persistent volume claims.

3. Uninstall the Operator:
   
   a. Click Operators → Installed Operators.

   b. Find the entry for AWS EFS Operator, and click the menu button on the right-hand side of the Operator.

   c. Click Uninstall and confirm the deletion.

4. Delete the shared volume CRD. This action triggers the deletion of the remaining Operator-owned resources.