⚠ A translation for your language preference does not exist.

When using Persistent Volumes with high file counts in OpenShift, why do pods fail to start or take an excessive amount of time to achieve "Ready" state?

SOLUTION 已验证 - 已更新 星期二 在 8:27 PM - English

环境

- Red Hat OpenShift Container Platform 3
- Red Hat OpenShift Container Platform 4.7+
- Docker Container Engine
- CRI-O Container Engine

问题

- When attaching volumes to pods in Red Hat OpenShift Container Platform, why do pods sometimes not start, or otherwise take an excessive amount of time to start?
- The volumes themselves have very high file counts, measured often in tens of thousands of files and directories (or higher).
- Starting the pods without the high file count volumes allows the pod to become "Ready" quickly (but without access to the data the volume provides).
- It is possible that entire nodes sometimes are marked as "NotReady" due to this issue as the container runtime (`docker` or `cri-o`) is unresponsive (as seen with hung `docker ps` or `crictl ps` commands).

- Pods not able to start falling into `CreateContainerError` status:

```
# oc get pod
NAME                    READY   STATUS                 RESTARTS   AGE
mypod-5-1111a           0/1     CreateContainerError   0          7m29s
```

- Pod deployments are failing with the following message: `Error: Failed to create pod sandbox: rpc error: code = Unknown desc = Kubelet may be retrying requests that are timing out in CRI-O due to system load: context deadline exceeded`

决议

- In upstream Kubernetes, and therefore within OpenShift, there exists two issues when Persistent Volumes were mounted to pods upon container creation that can significantly delay a container's ability to start:
  - File ownership update causes a significant delay in container startup.
  - SELinux file context relabeling causes a significant delay in container startup.
- The below section describes the current state of resolution for each issue.
- For a deeper understanding of the technical aspect of the issue, proceed to the below "Root Cause" section.

File Ownership Update

- This issue can be mitigated by applying `fsGroupChangePolicy` to the security context of a pod with the value of `OnRootMismatch`, which will prevent the entire volume from having file permissions re-applied (although it does not prevent it completely).
- As an example, the `securityContext` of a pod may have the following line to avoid this problem:

```
securityContext:
  fsGroupChangePolicy: "OnRootMismatch"
```

- The volume being mounted into the pod *must* support `fsGroup` permissions functionality, otherwise the above parameter will have no effect.

SELinux File Content Relabeling

- There currently exist two workarounds for skipping the SELinux relabeling for a volume.
- Note, both of these workarounds are implemented in the CRI-O level. There is work

being done tracking a proper fix upstream.
- This issue is being tracked upstream as well as within Red Hat.
- Please contact Red Hat Technical Support for direct assistance with this issue.
- For appropriate links and technical explanations, please refer to the "Root Cause" section below.

Skip SELinux Relabeling with `spc_t`

- This approach is the simplest, but requires the user to have SecurityContextConstraints (SCC) permission to update the SELinux type of the pod. Furthermore in case of a container runtime vulnerability and the container is not running with the `restricted` SCC, the container could possibly access any file on the host. (see below for details).
- This approach is the only one available in 4.7, first appearing in 4.7.37. It also is present in 4.8.16 and 4.9.2.
- To implement this workaround, the user must specify `type: "spc_t"` either on a pod or container `securityContext` :

```
securityContext:
  seLinuxOptions:
    type: "spc_t"
```

If a pod `securityContext` has the type `spc_t` set, then this type will be inherited by containers having no type specified at all. When this option is configured, CRI-O will skip the relabel, leaving it as it previously was.

- `spc_t` is a special SELinux type, standing for *super privileged container type*. A container having this type will not be constrained by SELinux policies .
- If the pod is running with a SCC having `runAsUser` set to `MustRunAsRange` like the `restricted` scc, this is safe because file access is already completely constrained. In the case of a container escape, the container process is running as a random UID and has no rights to modify anything on the host, although it could read world-readable files.
- If the pod is running with a SCC having `runAsUser` set to `runAsAny` , this is less safe because in case of a container runtime escape like https://access.redhat.com/security/vulnerabilities/RHSB-2021-004 an unconstrained container process would be able to overwrite files on the host.
- The default for OpenShift is to run containers with the `restricted` SCC.

Steps for skipping SELinux Relabeling with `spc_t`

1. Create a custom SCC :

```
# cat custom_scc.yaml

allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: false
allowedCapabilities: null
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: MustRunAs
groups:
- system:authenticated
kind: SecurityContextConstraints
metadata:
  name: custom
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities:
- KILL
- MKNOD
- SETUID
- SETGID
runAsUser:
  type: MustRunAsRange
seLinuxContext:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
users: []
volumes:
- configMap
- downwardAPI
- emptyDir
- persistentVolumeClaim
- projected
- secret

$ oc create -f custom_scc.yaml

$ oc get scc

custom   false   <no value>   RunAsAny    MustRunAsRange     MustRunAs
RunAsAny    <no value>   false
```

```
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secre
t"]

restricted false   <no value>   MustRunAs   MustRunAsRange     MustRunAs
RunAsAny    <no value>   false
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secre
t"]

2. Assign it to the default service account in a project where deployment is
stuck:

$ oc adm policy add-scc-to-user custom -z default -n <namespace>

Note: If you are not using default service account then change `default` with
the service account used by the pods.

3. Then added the following changes to the deployment in securityContext :

      securityContext:
        fsGroupChangePolicy: OnRootMismatch
        seLinuxOptions:
          type: spc_t
```

Skip SELinux Relabeling if already done with an annotation

- This option is a bit more complex, but is also more secure. It involves adding a custom `RuntimeClass`, which, when configured correctly in CRI-O, can interpret an annotation to skip the relabel if the top-level of the volume is found to have the correct label.
- It requires 4.8.16, 4.9.2 or any release >4.10
- A drawback of this approach is that the volume will have to be labeled at least once.
  - This will be done automatically by CRI-O, but could incur a container creation timeout.
- A consequence of this is that the container processes may fail to access sub-paths of the volume *if they're relabeled*.
  - An improvement in the SELinux relabeling code causes the top-level of the directory to be labeled last. Thus, assuming another process doesn't attempt to relabel a file in the volume, and assuming CRI-O doesn't crash during the intial relabel, the volume should be accessible to the container after the initial relabel.

1. First, a MachineConfig will need to be created to configure CRI-O to have a customized runtime class. This runtime class will be the same as the default one, but configure an `allowed_annotation`. The resulting CRI-O configuration file and subsequent MachineConfig could look like:

```
[crio.runtime.runtimes.selinux]
runtime_path = "/usr/bin/runc"
runtime_root = "/run/runc"
runtime_type = "oci"
allowed_annotations = ["io.kubernetes.cri-o.TrySkipVolumeSELinuxLabel"]
```

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-selinux-configuration
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,W2NyaW8ucnVudGltZS5ydW50aW1lcy5zZWxpbnV4XQpydW50aW1lX3BhdGggPSAiL3Vzci9
iaW4vcnVuYyIKcnVudGltZV9yb290ID0gIi9ydW4vcnVuYyIKcnVudGltZV90eXBlID0gIm9jaSIKYWx
sb3dlZF9hbm5vdGF0aW9ucyA9IFsiaW8ua3ViZXJuZXRlcy5jcmktby5UcnlTa2lwVm9sdW1lU0VMaW5
1eExhYmVsIl0K
        mode: 0640
        overwrite: true
        path: /etc/crio/crio.conf.d/01-selinux.conf
  osImageURL: ""
```

2. Next, a RuntimeClass must be created in the API server. The name should match that described in the CRI-O config above:

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: selinux
handler: selinux
```

3. Finally, the pod should be configured to have the annotation configured in the metadata, as well as the runtime class configured

```
apiVersion: v1
kind: Pod
metadata:
  name: sandbox
  annotations:
    io.kubernetes.cri-o.TrySkipVolumeSELinuxLabel: "true"
...
spec:
  runtimeClassName: selinux
...
```

- Now, when the pod is created, CRI-O will run it with the RuntimeClass `selinux`, which is configured to be allowed to process the annotation `io.kubernetes.cri-o.TrySkipVolumeSELinuxLabel`. This pod has the value "true" for `io.kubernetes.cri-o.TrySkipVolumeSELinuxLabel`, which means the SELinux relabel will be skipped *if the volume is already correctly labeled*.

根源

There are two distinct issues described in this article. The cause for each is discussed below.

Recursive File Ownership Delays

- When a pod is created and requires a volume to be mounted, the node's `kubelet` process performed a recursive file ownership change across the entire volume causing significant delays in pod creation.
- This issue, while not technically "solved", can be worked around by instructing the `kubelet` to perform the bare minimum of file system permissions changes by implementing the workaround mentioned in the above "Resolution" section.
- This Kubernetes Enhancement Proposal discusses a more long-term fix to completely avoid permissions checking by the `kubelet`.
- Work is ongoing upstream to include this into Kubernetes, and provided the solution is accepted into the community should be rebased into the Red Hat OpenShift Container Platform product.
- Please contact Red Hat Support if you believe you are experiencing this issue.

Recursive SELinux File Context Delays

- When a pod is created and requires a volume to be mounted, the container runtime

(either Docker or CRI-O on OpenShift nodes depending on version) is instructed by the node's `kubelet` process upon pod creation to relabel the entire volume with proper SELinux contexts.

- There exist some ways to work around this. By default, volume SElinux context relabeling happens for every volume on container startup and can cause significant delays when the volume has many files and directories as the procedure has to occur asynchronously, recursively through the entire volume. However, if a pod is configured to have `spc_t` type, or is correctly configured to have `io.kubernetes.cri-o.TrySkipVolumeSELinuxLabel` and the volume is already correctly labeled, then the relabel will be skipped.
- A Kubernetes Enhancement Program Issue exists, with work from Red Hat within it, attempting to provide a method in future version of Kubernetes (and therefore OpenShift) to avoid recursive relabeling that won't require as invasive/insecure changes to the pod spec.
- The official technical solution will require an API change, and is likely to be very complex, and therefore has no targeted release upstream at this time.
- Discussion of relying on the upcoming feature `fsGroupChangePolicy` has occurred within this GitHub issue and can be read about within this KEP update but is subject to change.
- Please contact Red Hat Support if you believe you are experiencing this issue.

产品（第） Red Hat OpenShift Container Platform 元件 cri-o docker openshift-node

类别 Troubleshoot 标记 crio docker openshift selinux

This solution is part of Red Hat's fast-track publication program, providing a huge library of solutions that Red Hat engineers have created while supporting our customers. To give you the knowledge you need the instant it becomes available, these articles may be presented in a raw and unedited form.

People who viewed this solution also viewed

SELinux relabel failed after upgrade to RHOCP 4.7

Solution - Sep 23, 2021

Containers are unable to start due to "no space left on device"

Solution - Mar 6, 2020

Docker: SELinux relabeling issue for a local nfs volume

Solution - Sep 11, 2017

4 评论

1 June 2022 2:46 PM

Gilles HAMEL

COMMUNITY MEMBER

25 Points

Skip SELinux Relabeling with spc_t

This approach is the simplest, but requires the user to have SecurityContextConstraints (SCC) permission to update the SELinux type of the pod.

to allow that, the pod must be running with a scc having :

```
seLinuxContext:
  type: RunAsAny
```

it was not clear for me.

回复

20 June 2022 1:21 PM

MYCOM OSI

NEWBIE

18 Points

Have this issue on ocp 3.11 with millions of files on many volumes. Tried to skip relabeling which made the Node went to NotReady. Waiting for v4 upgrade.

回复

COMMUNITY MEMBER

25 Points

22 June 2022 8:40 AM

Gilles HAMEL

Here, the workaround "Skip SELinux Relabeling if already done with an annotation" with OCP v4.8.35 / OCS 4.8.8 doesn't work. We have to use the workaround "Skip SELinux Relabeling with spc_t".

回复

RED HAT

COMMUNITY MEMBER

22 Points

24 June 2022 4:14 PM

Giovan Battista Salinetti

I'm afraid the file ownership update workaround won't work in OCP 3.11 since pod.spec.securityContext.fsGroupChangePolicy field was not yet available in this version.

回复