Troubleshoot Containers and Pods

Objectives

- Troubleshoot a pod by starting additional processes on its containers, changing their ephemeral file systems, and opening short-lived network tunnels.

Container Troubleshooting Overview

Containers are designed to be immutable and ephemeral. When changes are needed or a new container image is available, you can apply them to running containers without redeployment.

Updating a running container is best reserved for troubleshooting problematic containers. Red Hat does not generally recommend editing a running container to fix errors in a deployment. Changes to a running container are not captured in source control, but help to identify the needed corrections to the source code for the container functions. Capture these container updates in version control after you identify the necessary changes. Then, build a new container image and redeploy the application.

Custom alterations to a running container are incompatible with elegant architecture, reliability, and resilience for the environment.

CLI Troubleshooting Tools

Administrators use various tools to interact with, inspect, and alter running containers. Administrators can use commands such as `oc get` to gather initial details for a specified resource type. Other commands are available for detailed inspection of a resource, or to update a resource in real time.

> **NOTE**
>
> When interacting with the cluster containers, take suitable precautions with actively running components, services, and applications.

Use these tools to validate the functions and environment for a running container:

- `oc describe`: Display the details of a resource.
- `oc edit`: Edit a resource configuration by using the system editor.
- `oc patch`: Update a specific attribute or field for a resource.
- `oc cp`: Copy files and directories to and from containers.
- `oc exec`: Execute a command within a specified container.
- `oc port-forward`: Configure a port forwarder for a specified container.

- `oc logs`: Retrieve the logs for a specified container.
- `oc rsync`: Synchronize files and directories to and from containers.
- `oc rsh`: Start a remote shell within a specified container.

Editing Resources

Troubleshooting and remediation often begin with a phase of inspection and data gathering. When solving issues, the `describe` command can provide helpful details about the running resource, such as the definition of a container and its purpose.

The following example demonstrates use of the `oc describe` *`RESOURCE NAME`* command to retrieve information about a pod in the `openshift-dns` namespace:

```
[user@host ~]$ oc describe pod dns-default-lt13h
Name:                 dns-default-lt13h
Namespace:            openshift-dns
Priority:             2000001000
Priority Class Name: system-node-critical
...output omitted...
```

Various CLI tools can apply a change that you determine is needed to a running container. The `edit` command opens the specified resource in the default editor for your environment. This editor is specified by setting either the `KUBE_EDITOR` or the `EDITOR` environment variable, or otherwise with the editor application in your operating system.

The following example demonstrates use of the `oc edit` *`RESOURCE NAME`* command to edit a running container:

```
[user@host ~]$ oc edit pod mongo-app-sw88b

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will
be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  annotations:
...output omitted...
```

You can also use the `patch` command to update fields of a resource.

The following example uses the `patch` command to update the container image that a pod uses:

```
[user@host ~]$ oc patch pod valid-pod --type='json' \
-p='[{"op": "replace", "path": "/spec/containers/0/image", \
"value":"http://registry.access.redhat.com/ubi8/httpd-24"}]'
```

Copy Files to and from Containers

Administrators can copy files and directories to or from a container to inspect, update, or correct functionality. Adding a configuration file or retrieving an application log are common use cases.

> **NOTE**
>
> To use the `cp` command with the `oc` CLI, the `tar` binary must be present in the container. If the binary is absent, then an error message appears and the operation fails.

The following example demonstrates copying a file from a running container to a local directory by using the `oc cp` *`SOURCE DEST`* command:

```
[user@host ~]$ oc cp apache-app-kc82c:/var/www/html/index.html /tmp/index.bak

[user@host ~]$ ls /tmp
index.bak
```

The following example demonstrates use of the `oc cp` *`SOURCE DEST`* command to copy a file from a local directory to a directory in a running container:

```
[user@host ~]$ oc cp /tmp/index.html apache-app-kc82c:/var/www/html/

[user@host ~]$ oc exec -it apache-app-kc82c -- ls /var/www/html
index.html
```

> **NOTE**
>
> Targeting a file path within a pod for either the *`SOURCE`* or *`DEST`* arguments uses the *`pod_name:path`* format, and can include the `-c` *`container_name`* option to specify a container within the pod. If you omit the `-c` *`container_name`* option, then the command targets the first container in the pod.

Additionally, when using the `oc` CLI, file and directory synchronization is available by using the `oc rsync` command.

The following example demonstrates use of the `oc rsync` *`SOURCE_NAME DEST`* command to synchronize files from a running container to a local directory.

```
[user@host ~]$ oc rsync apache-app-kc82c:/var/www/ /tmp/web_files

[user@host ~]$ ls /tmp/web_files
cgi-bin
html
```

The `oc rsync` command uses the `rsync` client on your local system to copy changed files to and from a pod container. The `rsync` binary must be available locally and within the container for this approach. If the `rsync` binary is not found, then a `tar` archive is created on the local system and is sent to the container. The container then uses the `tar` utility to extract files from the archive. Without the `rsync` and `tar` binaries, an error message occurs and the `oc rsync` command fails.

> **NOTE**
>
> For Linux-based systems, you can install the `rsync` client and the `tar` utility on a local system by using a package manager, such as DNF. For Windows-based systems, you can install the `cwRsync` client. For more information about the `cwRysnc` client, refer to https://www.itefix.net/cwrsync.

Remote Container Access

Exposing a network port for a container is routine, especially for containers that provide a service. In a cluster, port forwarding connections are made through the kubelet, which maps a local port on your system to a port on a pod. Configuring port forwarding creates a request through the Kubernetes API, and creates a multiplexed stream, such as HTTP/2, with a `port` header that specifies the target port in the pod. The kubelet delivers the stream data to the target pod and port, and vice versa for egress data from the pod.

When troubleshooting an application that typically runs without a need to connect locally, you can use the port-forwarding function to expose connectivity to the pod for investigation. With this function, an administrator can connect on the new port and inspect the problematic application. After you remediate the issue, the application can be redeployed without the port-forward connection.

To forward a local port to a pod's port, use the following command syntax:

```
[user@host ~]$ oc port-forward RESOURCE EXTERNAL_PORT:CONTAINER_PORT
```

The following example demonstrates the use of the `oc port-forward` command to listen locally on port `8080` and to forward connections to port `80` on the pod:

```
[user@host ~]$ oc port-forward nginx-app-cc78k 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Connect to Running Containers

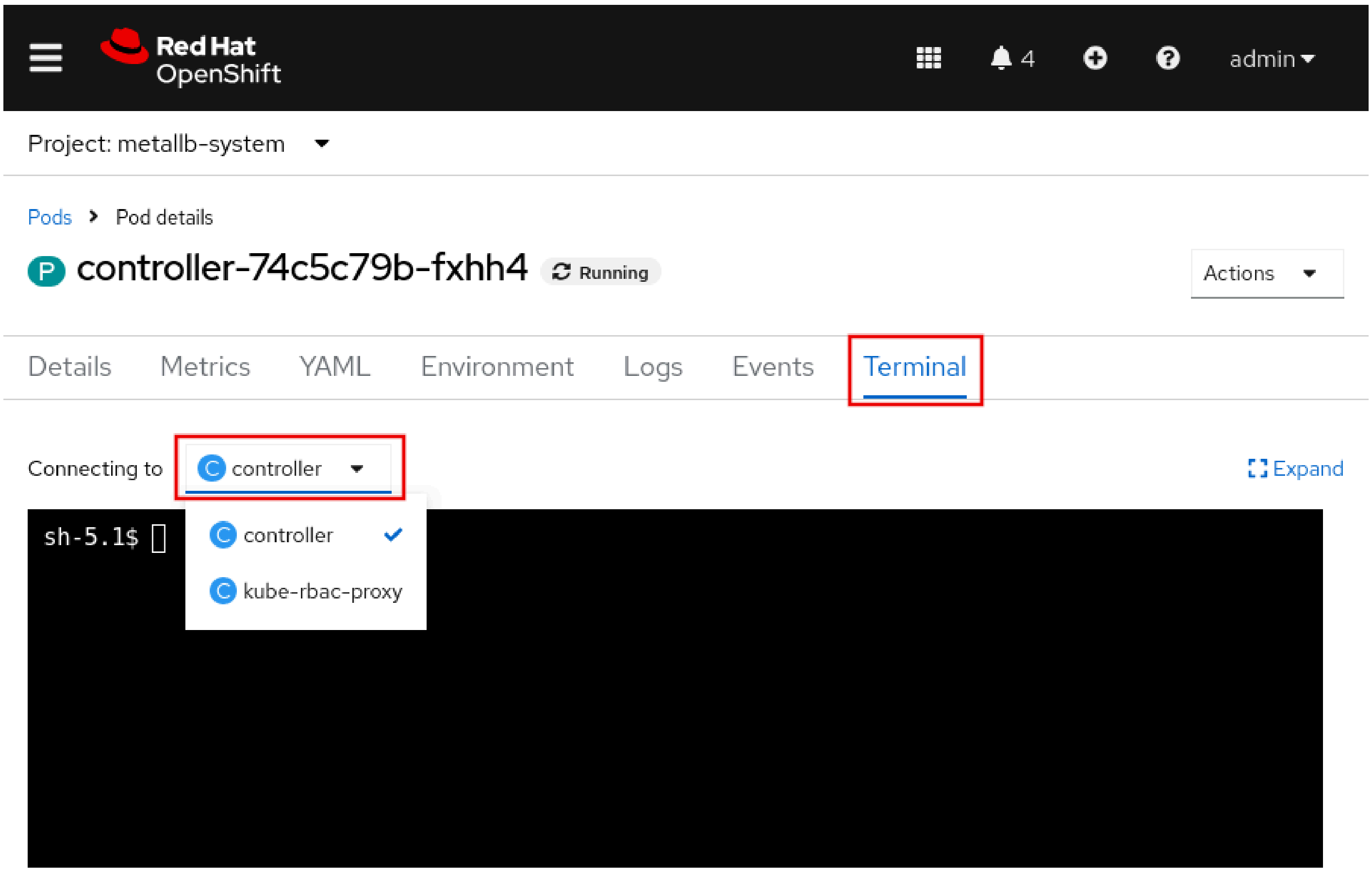
Administrators use CLI tools to connect to a container via a shell for forensic inspections. With this approach, you can connect to, inspect, and run any available commands within the specified container.

The following example demonstrates use of the `oc rsh` *`POD_NAME`* command to connect to a container via a shell:

```
[user@host ~]$ oc rsh tomcat-app-jw53r
sh-4.4#
```

If you need to connect to a specific container in a pod, then use the `-c` *`container_name`* option to specify the container name. If you omit this option, then the command connects to the first container in the pod.

You can also connect to running containers from the RHOCP web console. Click **Terminal** from the **Pod details** page, and select the running container from the **Connecting to** menu. If you have more than one container, then you can change between them to connect to the CLI.



```
sh-5.1$
```

Execute Commands in a Container

Passing commands to execute within a container from the CLI is another method for troubleshooting a running container. Use this method to send a command to run within the container, or to connect to the container, when further investigation is necessary.

Use the following command to pass and execute commands in a container:

```
[user@host ~]$ oc exec POD | TYPE/NAME [-c container_name] -- COMMAND [arg1 ... argN]
```

If you omit the `-c container_name` option, then the command targets the first container in the pod.

The following examples demonstrate the use of the `oc exec` command to execute the `ls` command in a container to list the contents of the container's root directory:

```
[user@host ~]$ oc exec -it mariadb-lc78h -- ls /
bin  boot  dev  etc  help.1  home  lib  lib64 ...
...output omitted...
```

```
[user@host ~]$ oc exec mariadb-lc78h -- ls /
bin
boot
dev
etc
...output omitted...
```

NOTE

It is common to add the `-it` flags to the `oc exec` command. These flags instruct the command to send `STDIN` to the container and `STDOUT`/`STDERR` back to the terminal. The format of the command output is impacted by the inclusion of the `-it` flags.

Container Events and Logs

Reviewing the historical actions for a container can offer insights into both the lifecycle and health of the deployment. Retrieving the cluster logs provides the chronological details of the container actions. Administrators inspect this log output for information and issues that occur in the running container.

For the following commands, use the `-c container_name` to specify a container in the pod. If you omit this option, then the command targets the first container in the pod.

The following examples demonstrate use of the `oc logs POD_NAME` command to retrieve the logs for a pod:

```
[user@host ~]$ oc logs BIND9-app-rw43j
Defaulted container "dns" out of: dns
.:5353
[INFO] plugin/reload: Running configuration SHA512 = 7c3d...3587
CoreDNS-1.9.2
...output omitted...
```

In Kubernetes, an `event` resource is a report of an event somewhere in the cluster. You can use the `oc get events` command to view pod events in a namespace:

```
[user@host ~]$ oc get events
LAST SEEN   TYPE     REASON           OBJECT                              MESSAGE
...output omitted...
21m         Normal   AddedInterface   pod/php-app-5d9b84b588-kzfxd     Add eth0 [10.8.
0.93/23] from ovn-kubernetes
21m         Normal   Pulled           pod/php-app-5d9b84b588-kzfxd     Container image
"registry.ocp4.example.com:8443/redhattraining/php-webapp:v4" already present on machin
e
21m         Normal   Created          pod/php-app-5d9b84b588-kzfxd     Created containe
r php-webapp
21m         Normal   Started          pod/php-app-5d9b84b588-kzfxd     Started containe
r php-webapp
```

Available Linux Commands in Containers

The use of Linux commands for troubleshooting applications can also help with troubleshooting containers. However, when connecting to a container, only the defined tools and applications within the container are available. You can augment the environment inside the container by adding the tools from this section or any other remedial tools to the container image.

Before you add tools to a container image, consider how the tools affect your container image:

- Additional tools increase the size of the image, which might impact container performance.
- Tools might require additional update packages and licensing terms, which can impact the ease of updating and distributing the container image.
- Hackers might exploit tools in the image.

Troubleshooting from Inside the Cluster

Administrators can alternatively author and deploy a container within the cluster for investigation and remediation. By creating a container image that includes the cluster troubleshooting tools, you have a reliable environment to perform these tasks from any computer with access to the cluster. This approach ensures that an administrator always has access to the tools for reliable troubleshooting and remediation of issues.

Additionally, administrators should plan to author a container image that provides the most valuable troubleshooting tools for containerized applications. In this way, you deploy this "toolbox" container to supplement the forensic process and to provide an environment with the required commands and tools for troubleshooting problematic containers. For example, the `toolbox` container can test how resources operate inside a cluster, such as to confirm whether a pod can connect to resources outside the cluster. Regular cluster users can also create a `toolbox` container to help with application troubleshooting. For example, a regular user could run a pod with a MySQL client to connect to another pod that runs a MySQL server.

Although this approach falls outside the focus of this course, because it is more application-level remediation than container-level troubleshooting, it is important to realize that containers have such capacity.

REFERENCES

For more information about troubleshooting pod issues, refer to the *Investigating Pod Issues* section in the *Troubleshooting* chapter in the Red Hat OpenShift Container Platform 4.18 *Support* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/support/index#investigating-pod-issues

For more information about how to copy files to and from pods, refer to the *Copying Files to or from an OpenShift Container Platform Container* section in the *Working with Containers* chapter in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-containers-copying-files

For more information about port forwarding, refer to the *Using Port Forwarding to Access Applications in a Container* section in the *Working with Containers* chapter in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-containers-port-forwarding-about_nodes-containers-port-forwarding

Kubernetes Documentation - Use Port Forwarding to Access Applications in a Cluster

For more information about executing remote commands in containers, refer to the *Executing Remote Commands in an OpenShift Container Platform Container* section in the *Working with Containers* chapter in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-containers-remote-commands-about_nodes-containers-remote-commands