Troubleshoot Containers and Pods

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
- Container needs to be redeployed when
  - Configuration changes
  - New image is available
- RH do not recommend editing running container to fix error
  - As changes not captured in source control
- RH recommend
  - make changes in source control
  - build new image and redeploy application
- Custom alterations to running container are incompatible with elegant architecture, reliability and resilience for environment

CLI Troubleshooting Tools

- `kubectl describe`: Display the details of a resource.
- `kubectl edit`: Edit a resource configuration by using the system editor.
- `kubectl patch`: Update a specific attribute or field for a resource.
- `kubectl replace`: Deploy a new instance of the resource.
- `kubectl cp`: Copy files and directories to and from containers.
- `kubectl exec`: Execute a command within a specified container.
- `kubectl explain`: Display documentation for a specified resource.
- `kubectl port-forward`: Configure a port forwarder for a specified container.
- `kubectl logs`: Retrieve the logs for a specified container.
- `oc status`: Display the status of the containers in the selected namespace.
- `oc rsync`: Synchronize files and directories to and from containers.
- `oc rsh`: Start a remote shell within a specified container.

Troubleshooting and remediation

- Begins with
  - phases of inspection
  - data gathering
- Use of oc logs, oc get command
- For details configuration, use of oc describe command

```
[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...
```

Editing Resources - oc edit command

- oc edit command opens specified resource in default editor (vi)
- Change the editor $ EDITOR=nano; export EDITOR

```
[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...
```

Editing Resources - oc patch command

- oc patch command do not open up an editor
- Example: update container to use different image

```
[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"}]'
```

Note

For more information about patching resources and the different merge methods, refer to Update API Objects in Place Using kubectl patch [https://kubernetes.io/docs/tasks/manage-kubernetes-objects/update-api-object-kubectl-patch/].

Copy Files to and from Containers

- Copy files and directories between local system and container.
- Objective
  - Configuration file can be updated, inspected, or corrected then recopy back to running container
  - Backup purposes
  - Retrieving application logs for archiving
- Requirement, tar binary must be present in the container
- Example 1: Copy file from running container to local directory

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

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

Example: oc cp command

- Copy file from running container to local directory

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

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

- Copy file from local directory to 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.bak
```

- Use of -c <container-name> option to copy file to or from specific container in pod

Use of oc rsync Command

- Copy only delta changes between local and container
- Uses the rsync tool
- This tool must be present on local host and in the running container
- If rsync is not found, then tar archive is created on local system then get sent and extracted in the container
- If both rsync and tar not found, operation fails

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

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

Remote Container Access

- Expose network port
  - made application / service in container expose to public user
- Port-forward local port on local system to port on pod
  - For troubleshooting application
  - Admin user connect to new port and inspect problematic app
  - After remediation, application can be re-deployed without port-forward connection
  - Connections made through kubelet agent
  - The kubelet delivers stream data to target pod and port, and vice versa for egress data from pod

Forwarding Ports - OpenShift

- Forward a local port to a pod port
- Port-forward mapping exists only in the workstation where the oc cmd is run
- Mapping forwards connections to a single pod
- Example: Forward port 30306 from developer machine to port 3306 on db pod, where a MYSQL server accept network connections

```
$ oc port-forward db 30306:3306
```

- When running this command, be sure to leave the terminal window running. Closing the window or cancelling the process stops the port mapping immediately

Connect to Running Containers

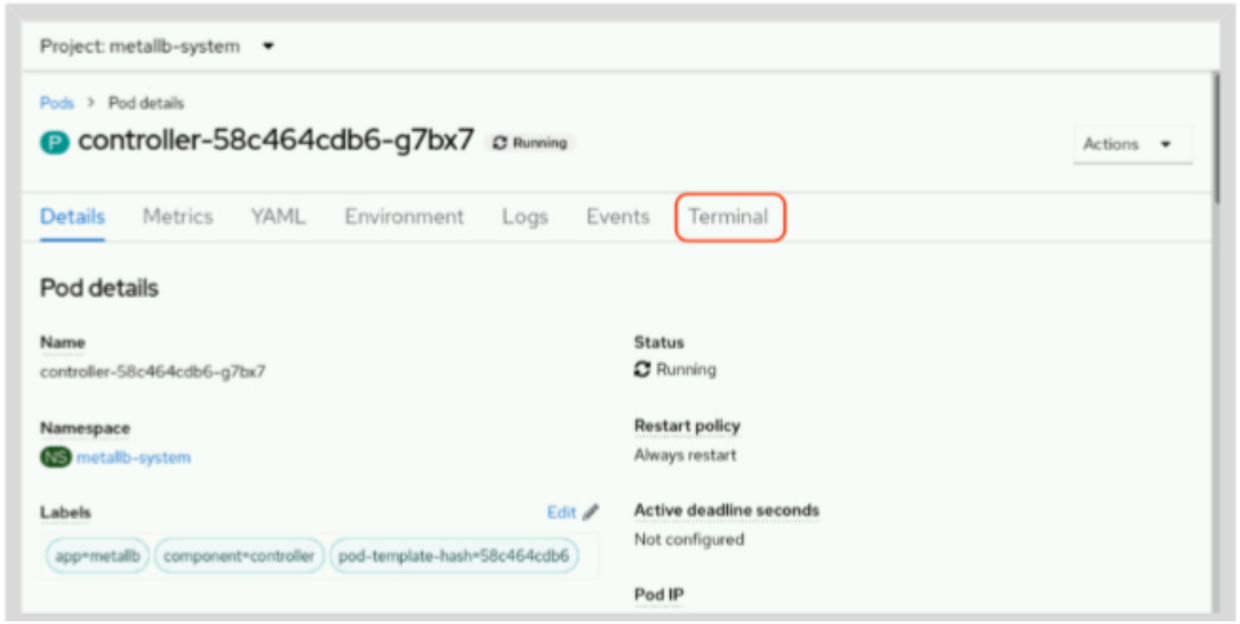
- Case use:
  - Needs cli access to shell for forensic inspections
  - Run any available commands within specified container
  - Access specific logs that is generated on standard output
- Use of oc rsh command
- Caveat:
  - do not accept -n namespace option
  - use -c to connect to specific container in pod

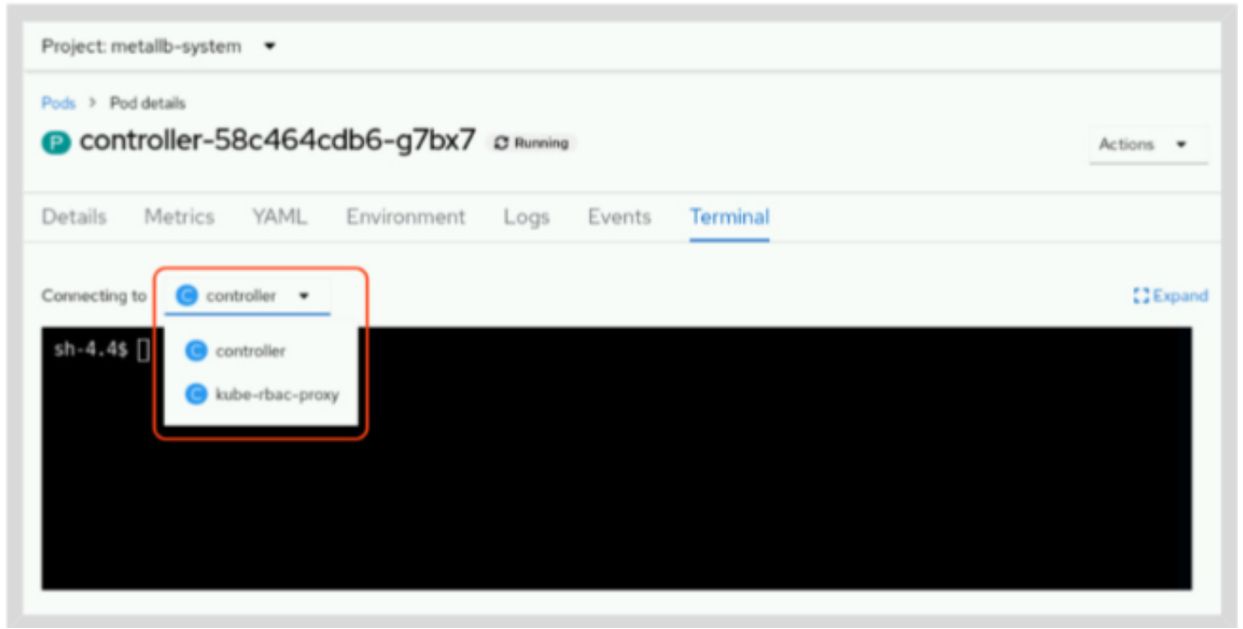
```
[user@host ~]$ oc rsh tomcat-app-jw53r
sh-4.4#
```

- Comparison:
  - oc exec can be used to execute a command remotely.
  - oc rsh command provides an easier way to keep a remote shell open persistently.

Connect to Running Containers

Using web console





Executing Commands in Container

- The syntax

```
oc exec POD | TYPE/NAME [-c container_name] -- COMMAND [arg1 ... argN]
```

- Interactively execute the ls command within container

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

- Non-interactively execute the ls command within container

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

Container Events and Logs

- Case Use:
  - View historical actions
  - Insights into both lifecycle and health of deployment
  - Provides chronological details of container actions
- View logs of 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...
```

Container Events and Logs

- Event resource is a report of event somewhere in cluster

```
[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 machine
21m         Normal   Created          pod/php-app-5d9b84b588-kzfxd    Created
 container php-webapp
21m         Normal   Started          pod/php-app-5d9b84b588-kzfxd    Started
 container php-webapp
```

Available Linux Commands in Container

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

- Troubleshoot cluster, its components, or running applications by connecting remotely
- Tools might not be available from administrator computer or alternative machine or some containerized application
- Alternatively deploy a container within cluster for the investigation and remediation
  - To include necessary tools
  - to provide reliable environment
- Build and update this "toolbox" image
  - store it in secured private registry
  - consists of common necessary tools
  - can test how resources operate inside cluster
- Regular user can use the toolbox image to help with application troubleshooting
- Example: a regular user could run a pod with a MySQL client to connect to another pod that runs a MySQL server.

Guided Exercise: Troubleshoot and fix a failed MySQL pod and manually initialize a database with test data.

You should be able to:

- Investigate errors with creating a pod.
- View the status, logs, and events for a pod.
- Copy files into a running pod.
- Connect to a running pod by using port forwarding.

You should be able to:

- Deploy a pod from a container image.
- Retrieve the status and events of a pod.
- Troubleshoot a failed pod.
- Edit pod resources.
- Copy files to a running pod for diagnostic purposes.
- Use port forwarding to connect to a running pod.

Lab: Run Applications as Containers and Pods

Summary

MANY RESOURCES IN KUBERNETES AND RHOCP CREATE OR AFFECT PODS.

RESOURCES ARE CREATED IMPERATIVELY OR DECLARATIVELY. THE IMPERATIVE STRATEGY INSTRUCTS THE CLUSTER WHAT TO DO. THE DECLARATIVE STRATEGY DEFINES THE STATE THAT THE CLUSTER MATCHES.

THE OC NEW-APP COMMAND CREATES RESOURCES THAT ARE DETERMINED VIA HEURISTICS.

THE MAIN WAY TO DEPLOY AN APPLICATION IS BY CREATING A DEPLOYMENT.

THE WORKLOAD API INCLUDES SEVERAL RESOURCES TO CREATE PODS. THE CHOICE BETWEEN RESOURCES DEPENDS ON FOR HOW LONG AND HOW OFTEN THE POD NEEDS TO RUN.

A *JOB* RESOURCE EXECUTES A ONE-TIME TASK ON THE CLUSTER VIA A POD. THE CLUSTER RETRIES THE JOB UNTIL IT SUCCEEDS, OR IT RETRIES A SPECIFIED NUMBER OF ATTEMPTS.

RESOURCES ARE ORGANIZED INTO PROJECTS AND ARE SELECTED VIA LABELS.

A *ROUTE* CONNECTS A PUBLIC-FACING IP ADDRESS AND A DNS HOSTNAME TO AN INTERNAL-FACING *SERVICE* IP ADDRESS. SERVICES PROVIDE NETWORK ACCESS BETWEEN PODS, WHEREAS ROUTES PROVIDE NETWORK ACCESS TO PODS FROM USERS AND APPLICATIONS OUTSIDE THE RHOCP CLUSTER.