Troubleshoot PODs with Commands for Kubernetes and CEE OPS-Center

Contents

Introduction

This document describes how to troubleshoot PODs with commands for Kubernetes and CEE OPS-Center.

Troubleshoot PODs with Commands for Kubernetes and CEE OPS-Center

1. k8s CLIs

1.1 List all namespace

Command:

```
kubectl get namespace
```

Example:

```
cisco@brusmi-master1:~$ kubectl get namespace

NAME               STATUS   AGE

cee-cee            Active   6d

default            Active   6d

kube-node-lease    Active   6d

kube-public        Active   6d

kube-system        Active   6d

lfs                Active   6d
```

```
nginx-ingress     Active   6d

smf-data          Active   6d

smi-certs         Active   6d

smi-vips          Active   6d
```

1.2 List all the services for a particular namespace:

Command:

```
kubectl get svc -n <namespace>
```

Example:

```
cisco@brusmi-master1:~$ kubectl get svc -n smf-data

NAME                                    TYPE        CLUSTER-IP       EXTERNAL-IP    PORT(S)

base-entitlement-smf                    ClusterIP   10.97.93.253     <none>         8000/TCP

datastore-ep-session                    ClusterIP   10.101.15.88     <none>         8882/TCP

datastore-notification-ep               ClusterIP   10.110.182.26    <none>         8890/TCP

datastore-tls-ep-session                ClusterIP   10.110.115.33    <none>         8883/TCP

documentation                           ClusterIP   10.110.85.239    <none>         8080/TCP

etcd                                    ClusterIP   None             <none>         2379/TCP,7070/TCP

etcd-smf-data-etcd-cluster-0            ClusterIP   10.103.194.229   <none>         2380/TCP,2379/TCP

grafana-dashboard-app-infra             ClusterIP   10.98.161.155    <none>         9418/TCP

grafana-dashboard-cdl                   ClusterIP   10.104.32.111    <none>         9418/TCP

grafana-dashboard-smf                   ClusterIP   10.106.64.191    <none>         9418/TCP

gtpc-ep                                 ClusterIP   10.99.49.25      x.x.x.201   9003/TCP,8080/TCP

helm-api-smf-data-ops-center            ClusterIP   10.109.206.198   <none>         3000/TCP

kafka                                   ClusterIP   None             <none>         9092/TCP,7070/TCP

li-ep                                   ClusterIP   10.106.134.35    <none>         9003/TCP,8080/TCP

local-ldap-proxy-smf-data-ops-center    ClusterIP   10.99.160.226    <none>         636/TCP,369/TCP

oam-pod                                 ClusterIP   10.105.223.47    <none>         9008/TCP,7001/TCP,887

ops-center-smf-data-ops-center          ClusterIP   10.103.164.204   <none>         8008/TCP,8080/TCP,202
```

```
smart-agent-smf-data-ops-center       ClusterIP   10.97.143.81     <none>         8888/TCP

smf-n10-service                       ClusterIP   10.102.197.22    10.10.10.205   8090/TCP

smf-n11-service                       ClusterIP   10.108.109.186   10.10.10.203   8090/TCP

smf-n40-service                       ClusterIP   10.111.170.158   10.10.10.206   8090/TCP

smf-n7-service                        ClusterIP   10.102.140.179   10.10.10.204   8090/TCP

smf-nodemgr                           ClusterIP   10.102.68.172    <none>         9003/TCP,8884/TCP,92

smf-protocol                          ClusterIP   10.111.219.156   <none>         9003/TCP,8080/TCP

smf-rest-ep                           ClusterIP   10.109.189.99    <none>         9003/TCP,8080/TCP,92

smf-sbi-service                       ClusterIP   10.105.176.248   10.10.10.201   8090/TCP

smf-service                           ClusterIP   10.100.143.237   <none>         9003/TCP,8080/TCP

swift-smf-data-ops-center             ClusterIP   10.98.196.46     <none>         9855/TCP,50055/TCP,5

zookeeper                             ClusterIP   None             <none>         2888/TCP,3888/TCP

zookeeper-service                     ClusterIP   10.109.109.102   <none>         2181/TCP,7070/TCP
```

1.3 List all pods for a particular namespace:

Command:

```
kubectl get pods -n <namespace>
```

Example:

```
cisco@brusmi-master1:~$ kubectl get pods -n smf-data

NAME                                                 READY   STATUS    RESTARTS   AGE

api-smf-data-ops-center-57c8f6b4d7-wt66s             1/1     Running   0          6d

base-entitlement-smf-fcdb664d-fkgss                  1/1     Running   0          6d

cache-pod-0                                          1/1     Running   0          6h53m

cache-pod-1                                          1/1     Running   0          6h53m

cdl-ep-session-c1-dbb5f7874-4gmfr                    1/1     Running   0          6h53m

cdl-ep-session-c1-dbb5f7874-5zbqw                    1/1     Running   0          6h53m

cdl-index-session-c1-m1-0                            1/1     Running   0          6h53m

cdl-slot-session-c1-m1-0                             1/1     Running   0          6h53m
```

```
documentation-5dc8d5d898-mv6kx                    1/1     Running    0          6d
etcd-smf-data-etcd-cluster-0                      1/1     Running    0          6h53m
grafana-dashboard-app-infra-5b8dd74bb6-xvlln      1/1     Running    0          6h53m
grafana-dashboard-cdl-5df868c45c-vbr4r            1/1     Running    0          6h53m
grafana-dashboard-smf-657755b7c8-fvbdt            1/1     Running    0          6h53m
gtpc-ep-n0-0                                      1/1     Running    0          6h53m
kafka-0                                           1/1     Running    0          6h53m
li-ep-n0-0                                        1/1     Running    0          6h53m
oam-pod-0                                         1/1     Running    0          6h53m
ops-center-smf-data-ops-center-7fbb97d9c9-tx7qd   5/5     Running    0          6d
smart-agent-smf-data-ops-center-6667dcdd65-2h7nr  0/1     Evicted    0          6d
smart-agent-smf-data-ops-center-6667dcdd65-6wfvq  1/1     Running    0          4d18h
smf-nodemgr-n0-0                                  1/1     Running    0          6h53m
smf-protocol-n0-0                                 1/1     Running    0          6h53m
smf-rest-ep-n0-0                                  1/1     Running    0          6h53m
smf-service-n0-0                                  1/1     Running    5          6h53m
smf-udp-proxy-0                                   1/1     Running    0          6h53m
swift-smf-data-ops-center-68bc75bbc7-4zdc7        1/1     Running    0          6d
zookeeper-0                                       1/1     Running    0          6h53m
zookeeper-1                                       1/1     Running    0          6h52m
zookeeper-2                                       1/1     Running    0          6h52m
```

1.4 List full details for specific pod names (labels, images, ports, volumes, events, and more).

Command:

```
kubectl describe pods <pod_name> -n <namespace>
```

Example:

```
cisco@brusmi-master1:~$ kubectl describe pods smf-service-n0-0 -n smf-data
```

```
smf-service-n0-0    <<< POD name

smf-data            <<< Namespace
```

2. k8s Logs and Full Core

2.1 Get Container name for specific pod:

Command:

```
kubectl describe pods <pod_name> -n <namespace> | grep Containers -A1
```

Example:

```
cisco@brusmi-master1:~$ kubectl describe pods smf-service-n0-0 -n smf-data | grep Containers -A1
```

Containers:

```
 smf-service:

--

  ContainersReady   True

  PodScheduled      True
```

2.2 Look for logs when a pod crash is observed on Kubernetes:

Command:

```
kubectl get pods -n <namespace> | grep -v Running
```

Example:

```
cisco@brusmi-master1:~$ kubectl get pods -n smf-data | grep -v Running

NAME                                                   READY   STATUS             RESTARTS   AGE
```

```
smart-agent-smf-data-ops-center-6667dcdd65-2h7nr   0/1     Evicted            0          5d23h
smf-service-n0-0                                   0/1     CrashLoopBackOff   2          6h12m
```

Command:

```
kubectl logs <pod_name> -c <container_name> -n <namespace>
```

Example:

```
cisco@brusmi-master1:~$ kubectl logs smf-service-n0-0 -c smf-service -n smf-data
/opt/workspace
-rwxrwxrwx 1 root root 84180872 Mar 31 06:18 /opt/workspace/smf-service
Launching: /opt/workspace/tini /opt/workspace/smf-service
2020-06-09 20:26:16.341043 I | proto: duplicate proto type registered: internalmsg.SessionKey
2020-06-09 20:26:16.341098 I | proto: duplicate proto type registered: internalmsg.NInternalTxnMsg
2020-06-09 20:26:16.343170 I | smf-service [INFO] [main.go:18] [smfservice] ##########################MA
#########
2020-06-09 20:26:16.343197 I | smf-service [INFO] [main.go:19] [smfservice] ##########################S
#########
2020-06-09 20:26:16.343210 I | smf-service [INFO] [main.go:20] [smfservice]                         SMF-S

2020-06-09 20:26:16.343221 I | smf-service [INFO] [main.go:21] [smfservice] ##########################S
#########
2020-06-09 20:26:16.343232 I | smf-service [INFO] [main.go:22] [smfservice] ##########################M
#########
2020/06/09 20:26:16.343 smf-service [DEBUG] [Tracer.go:181] [unknown] Loaded initial tracing configurati
aegerTransportType: , TracerEndpoint: , ServiceName: smf-service, TracerServiceName: , EnableTracePercen
.
.
2020/06/09 20:44:28.157 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.core] Rest message rec
2020/06/09 20:44:28.158 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.core] Set Ping as name
```

```
2020/06/09 20:44:28.159 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.application.core] Ping se

2020/06/09 20:44:30.468 smf-service [DEBUG] [MetricsServer_v1.go:305] [infra.application.core] Checkpoir

2020/06/09 20:44:31.158 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.core] Rest message rec

2020/06/09 20:44:31.158 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.core] Set Ping as name

2020/06/09 20:44:31.158 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.application.core] Ping se


smf-service-n0-0    <<< POD name

smf-service         <<< Container Name

smf-data            <<< Namespace
```

2.3 Verify if coredumps were generated:

Command:

```
ls -lrt /var/lib/systemd/coredump/
```

Example:

```
cisco@brusmi-master1:~$ ls -lrt /var/lib/systemd/coredump/

total 0
```

Note: The core file shall be generated in **/var/lib/systemd/coredump/** path in the respective VM. The core is also available on the TAC Dashboard.

---

3. Create TAC-Debug on CEE

3.1 Login into cee Ops-Center from Master k8s:

```
cisco@brusmi-master1:~$ kubectl get namespace

NAME              STATUS   AGE

cee-cee           Active   5d3h

default           Active   5d3h

kube-node-lease   Active   5d3h
```

```
kube-public       Active   5d3h

kube-system       Active   5d3h

lfs               Active   5d3h

nginx-ingress     Active   5d3h

smf-data          Active   5d3h

smi-certs         Active   5d3h

smi-vips          Active   5d3h



cisco@brusmi-master1:~$ ssh -p 2024 admin@$(kubectl get svc -n cee-cee | grep ^ops-center | awk '{print

 admin@10.102.44.219's password:

Welcome to the cee CLI on brusmi/cee

admin connected from 192.x.0.1 using ssh on ops-center-cee-cee-ops-center-79cf55b49b-6wrh9

[brusmi/cee] cee#
```

> **Note**: In the example mentioned previously, the CEE namespace is â€œcee-ceeâ€�. You must replace this name in case you require it.

3.2 Generate the TAC package ID to reference collection files retrieved:

Command:

```
tac-debug-pkg create from <Start_time> to <End_time>
```

Example:

```
[brusmi/cee] cee# tac-debug-pkg create from 2020-06-08_14:00:00 to 2020-06-08_15:00:00

response : Tue Jun  9 00:22:17 UTC 2020 tac-debug pkg ID : 1592948929
```

Also, you can include additional filters like namespace or pod_name as follows:

Command:

```
tac-debug-pkg create from <Start_time> to <End_time> logs-filter { namespace <namespace> pod_name <pod_n
```

Example:

```
[brusmi/cee] cee# tac-debug-pkg create from 2020-06-08_14:00:00 to 2020-06-08_15:00:00 logs-filter { nam

response : Tue Jun  9 00:28:49 UTC 2020 tac-debug pkg ID : 1591662529
```

Note: It is recommended to generate a tac package ID for a slot period of time (1 hr or max 2 hrs).

3.3 Display the status of each service:

```
[brusmi/cee] cee# tac-debug-pkg status

response : Tue Jun  9 00:28:51 UTC 2020

Tac id: 1591662529

Gather core: completed!

Gather logs: in progress

Gather metrics: in progress

Gather stats: completed!

Gather config: completed!

[brusmi/cee] cee#



[brusmi/cee] cee# tac-debug-pkg status

response : Tue Jun  9 00:43:45 UTC 2020

No active tac debug session          <<< If none active tac debug session is displayed, it means that a
```

Note: If there is no available disk space, please remove old debug files.

```
[brusmi/cee] cee# tac-debug-pkg create from 2020-06-08_09:00:00 to 2020-06-08_10:00:00 logs-filter { nam

response : Tue Jun  9 00:45:48 UTC 2020

Available disk space on node is less than 20 %. Please remove old debug files and retry.
```

```
[brusmi/cee] cee# tac-debug-pkg delete tac-id 1591662529
```

3.4 Create a TAC Debug ID to gather Metrics only:

```
[nyucs504-cnat/global] cee# tac-debug-pkg create from 2021-02-24_12:30:00 to 2021-02-24_14:30:00 cores f
response : Wed Feb 24 19:39:49 UTC 2021 tac-debug pkg ID : 1614195589
```

4. Download TAC Debug

Currently, there are three different options to download the TAC Debug from CEE:

4.1 SFTP from Master VIP (less recommended, it takes a long).

4.1.1 Get the URL to download the logs gathered on **tac package ID** :

Command:

```
kubectl get ingress -n <namespace> | grep show-tac
```

Example:

```
cisco@brusmi-master1:~$ kubectl get ingress -n cee-cee | grep show-tac

show-tac-manager-ingress                show-tac-manager.cee-cee-smi-show-tac.192.168.208.10.xxx.x
```

4.1.2 Compress and get the tac-debug file from **show-tac-manager** pod:

a. Get the ID of the show-tac pod.

Command:

```
kubectl get pods -n <namespace> | grep show-tac
```

Example:

```
cisco@brusmi-master1:~$ kubectl get pods -n cee-cee | grep show-tac
show-tac-manager-85985946f6-bflrc 2/2 Running 0 12d
```

b. Run exec command in **show-tac pod**, and compress the TAC Debug logs.

Command:

```
kubectl exec -it -n <namespace> <pod_name> bash
```

Example:

```
cisco@brusmi-master1:~$ kubectl exec -it -n cee-cee show-tac-manager-85985946f6-bflrc bash

Defaulting container name to show-tac-manager.

Use 'kubectl describe pod/show-tac-manager-85985946f6-bflrc -n cee-cee' to see all of the containers in

groups: cannot find name for group ID 101

groups: cannot find name for group ID 190

groups: cannot find name for group ID 303

I have no name!@show-tac-manager-85985946f6-bflrc:/show-tac-manager/bin$ cd /home/tac/

I have no name!@show-tac-manager-85985946f6-bflrc:/home/tac$ tar -zcvf tac-debug_1591662529.tar.gz 15916

1591662529/

1591662529/config/

1591662529/config/192.x.1.14_configuration.tar.gz.base64

1591662529/stats/

1591662529/stats/Stats_2020-06-08_14-00-00_2020-06-08_15-00-00.tar.gz

1591662529/manifest.json

1591662529/metrics/

1591662529/metrics/Metrics_2020-06-08_14-00-00_2020-06-08_15-00-00.tar.gz

1591662529/web/

1591662529/web/index.html

1591662529/logs/

1591662529/logs/brusmi-master1/

1591662529/logs/brusmi-master1/brusmi-master1_Logs_2020-06-08_14-00-00_2020-06-08_15-00-00.tar.gz

I have no name!@show-tac-manager-85985946f6-bflrc:/home/tac$ ls

1591662490  1591662529  1592265088  tac-debug_1591662529.tar.gz
```

4.1.3 Copy the file to **/tmp** directory on Master VIP:

Command:

```
kubectl cp <namespace>/<show-tac_pod_name>:/home/tac/<file_name.tar.gz> /tmp/<file_name.tar.gz>
```

Example:

```
cisco@brusmi-master1:~$ kubectl cp cee-cee/show-tac-manager-85985946f6-bflrc:/home/tac/tac-debug_159166

Defaulting container name to show-tac-manager.

tar: Removing leading `/' from member names

cisco@brusmi-master1:~$ cd /tmp

cisco@brusmi-master1:/tmp$ ls

cee.cfg

tac-debug_1591662529.tar.gz

tiller_service_acct.yaml
```

4.1.4 Transfer file via sftp from Master VIP.

4.2 Download the TAC Debug with **wget** command (macOS/Ubuntu).

4.2.1 Get show-tac link from "k8s get ingress" output:

```
cisco@brusmi-master1:~$ kubectl get ingress -n cee-cee | grep show-tac

show-tac-manager-ingress                show-tac-manager.cee-cee-smi-show-tac.192.168.208.10.xxx.x
```

4.2.2 Enter the **wget** command from your PC terminal:

```
wget -r -np https://show-tac-manager.cee-cee-smi-show-tac.192.168.208.10.xxx.x/tac/
<tac-id>/ --no-check-certificate --http-user=<NTID_username>
--http-password=<NTID_password>
```

5. Collect logs from CEE for all the SMF PODs

5.1 Login into **smf-data**Ops-Center from Master k8s:

```
cisco@brusmi-master1:~$ ssh -p 2024 admin@$(kubectl get svc -n smf-data | grep ^ops-center | awk '{print

 admin@10.103.164.204's password:

Welcome to the smf CLI on brusmi/data

admin connected from 192.x.0.1 using ssh on ops-center-smf-data-ops-center-7fbb97d9c9-tx7qd
```

5.2 Confirm if â€œlogging level applicationâ€� is enabled:

```
[brusmi/data] smf# show running-config | i logging

logging level application debug

logging level transaction debug

logging level tracing debug

logging name infra.config.core level application debug

logging name infra.config.core level transaction debug

logging name infra.config.core level tracing debug

logging name infra.message_log.core level application debug

logging name infra.message_log.core level transaction debug

logging name infra.resource_monitor.core level application off

logging name infra.rest_server.core level application debug
```

5.3 Login into cee Ops-Center from Master k8s:

```
cisco@brusmi-master1:~$ ssh -p 2024 admin@$(kubectl get svc -n cee-cee | grep ^ops-center | awk '{print

 admin@10.102.44.219's password:

Welcome to the cee CLI on brusmi/cee

admin connected from 192.x.0.1 using ssh on ops-center-cee-cee-ops-center-79cf55b49b-6wrh9

[brusmi/cee] cee#
```

Note: In the example mentioned previously, the CEE namespace is â€œcee-ceeâ€�. You must replace this name in case you require it.

5.4 Tail the logs of all the SMF PODs that start with â€œ**smf-**â€œ (**smf-nodemgr**, **smf-protocol**, **smf-rest** , **smf-service**,

smf-udp-proxy). Collect the logs for a few seconds, and use Ctrl+C to stop data collection:

```
[brusmi/cee] cee# cluster logs ^smf- -n smf-data

error: current-context must exist in order to minify

Will tail 5 logs...

smf-nodemgr-n0-0

smf-protocol-n0-0

smf-rest-ep-n0-0

smf-service-n0-0

smf-udp-proxy-0

[smf-service-n0-0] 2020/06/08 17:04:57.331 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:04:57.331 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:04:57.331 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.applic

[smf-service-n0-0] 2020/06/08 17:05:00.331 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:00.332 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:00.332 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.applic

[smf-service-n0-0] 2020/06/08 17:05:01.658 smf-service [DEBUG] [MetricsServer_v1.go:305] [infra.applicat

[smf-service-n0-0] 2020/06/08 17:05:03.330 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:03.330 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:03.330 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.applic

[smf-service-n0-0] 2020/06/08 17:05:06.330 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:06.330 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.cor

[smf-service-n0-0] 2020/06/08 17:05:06.330 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.applic

[smf-protocol-n0-0] 2020/06/08 17:04:58.441 smf-protocol [DEBUG] [RestRouter.go:24] [infra.rest_server.c

[smf-service-n0-0] 2020/06/08 17:05:06.661 smf-service [DEBUG] [MetricsServer_v1.go:305] [infra.applicat

[smf-protocol-n0-0] 2020/06/08 17:04:58.441 smf-protocol [DEBUG] [RestRouter.go:43] [infra.rest_server.c

[smf-protocol-n0-0] 2020/06/08 17:04:58.441 smf-protocol [INFO] [ApplicationEndpoint.go:333] [infra.appl

[smf-nodemgr-n0-0] 2020/06/08 17:04:57.329 smf-nodemgr [DEBUG] [CacheClient.go:118] [infra.cache_client.
```

Note: You can be more specific in case you need to collect logs from a particular pod, container or multiple pods.

```
### Specific pod ###

[brusmi/cee] cee# cluster logs smf-nodemgr-n0-0 -n smf-data

[brusmi/cee] cee# cluster logs smf-rest-ep-n0-0 -n smf-data


### Specific container ###

[brusmi/cee] cee# cluster logs smf-nodemgr -n smf-data

[brusmi/cee] cee# cluster logs smf-service -n smf-data

[brusmi/cee] cee# cluster logs zookeeper -n smf-data

[brusmi/cee] cee# cluster logs smf-rest-ep -n smf-data


### Multiple pods ###

[brusmi/cee] cee# cluster logs "(smf-service.|smf-rest.|smf-nodemgr.|smf-protocol.|gtpc-ep.|smf-udp-prox
```

6. Access into Grafana

6.1 Get the URL to access Grafana:

```
cisco@brusmi-master1:~$ kubectl get ingress -n cee-cee | grep grafana
grafana-ingress grafana.192.168.168.208.10.xxx.x 80, 443 6d18h
```

6.2 Open a web page with HTTPS as follows:

```
https://grafana.192.168.208.10.xxx.x
```