

Resolución de problemas de POD con comandos para Kubernetes y CEE OPS-Center

Contenido

[Introducción](#)

[Resolución de problemas de POD con comandos para Kubernetes y CEE OPS-Center](#)

[1. CLI k8s](#)

[2. k8s Registros y núcleo completo](#)

[3. Crear depuración TAC en CEE](#)

[4. Descargar depuración de TAC](#)

[5. Recopile registros de CEE para todos los POD SMF](#)

[6. Acceso a Grafana](#)

Introducción

Este documento describe cómo resolver problemas de POD con comandos para Kubernetes y CEE OPS-Center.

Resolución de problemas de POD con comandos para Kubernetes y CEE OPS-Center

1. CLI k8s

1.1 Lista de todos los espacios de nombres

Comando:

```
kubectl get namespace
```

Ejemplo:

```
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 Enumera todos los servicios para un espacio de nombres determinado:

Comando:

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

Ejemplo:

```
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-cd1	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, 88
ops-center-smf-data-ops-center	ClusterIP	10.103.164.204	<none>	8008/TCP, 8080/TCP, 20
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 Enumera todos los pods para un espacio de nombres determinado:

Comando:

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

Ejemplo:

```
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-xv1ln	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 Enumere los detalles completos de nombres de grupos de dispositivos específicos (etiquetas, imágenes, puertos, volúmenes, eventos, etc.).

Comando:

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

Ejemplo:

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

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

2. k8s Registros y núcleo completo

2.1 Obtener nombre del contenedor para un grupo de dispositivos específico:

Comando:

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

Ejemplo:

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

Contenedores:

```
smf-service:
--
ContainersReady    True
PodScheduled       True
```

2.2 Busque registros cuando se observe un desperfecto de grupo de dispositivos en Kubernetes:

Comando:

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

Ejemplo:

```
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

Comando:

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

Ejemplo:

```
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] #####  
#####
```

```
2020-06-09 20:26:16.343197 I | smf-service [INFO] [main.go:19] [smfservice] #####  
#####
```

```
2020-06-09 20:26:16.343210 I | smf-service [INFO] [main.go:20] [smfservice] SMF-
```

```
2020-06-09 20:26:16.343221 I | smf-service [INFO] [main.go:21] [smfservice] #####  
#####
```

```
2020-06-09 20:26:16.343232 I | smf-service [INFO] [main.go:22] [smf-service] #####  
#####  
2020/06/09 20:26:16.343 smf-service [DEBUG] [Tracer.go:181] [unknown] Loaded initial tracing configurat  
aegerTransportType: , TracerEndpoint: , ServiceName: smf-service, TracerServiceName: , EnableTracePerce  
.  
.  
2020/06/09 20:44:28.157 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.core] Rest message re  
2020/06/09 20:44:28.158 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.core] Set Ping as nam  
2020/06/09 20:44:28.159 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.application.core] Ping s  
2020/06/09 20:44:30.468 smf-service [DEBUG] [MetricsServer_v1.go:305] [infra.application.core] Checkpoi  
2020/06/09 20:44:31.158 smf-service [DEBUG] [RestRouter.go:24] [infra.rest_server.core] Rest message re  
2020/06/09 20:44:31.158 smf-service [DEBUG] [RestRouter.go:43] [infra.rest_server.core] Set Ping as nam  
2020/06/09 20:44:31.158 smf-service [INFO] [ApplicationEndpoint.go:333] [infra.application.core] Ping s
```

```
smf-service-n0-0 <<< POD name  
smf-service <<< Container Name  
smf-data <<< Namespace
```

2.3 Verificar si se generaron vaciados de núcleo:

Comando:

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

Ejemplo:

```
cisco@brusmi-master1:~$ ls -lrt /var/lib/systemd/coredump/  
total 0
```

Nota: El archivo de núcleo se generará en la `/var/lib/systemd/coredump/` ruta en la VM respectiva. El núcleo también está disponible en el panel del TAC.

3. Crear depuración TAC en CEE

3.1 Inicio de sesión en el centro de operaciones de cee desde máster k8:

```
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
1fs	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-ops-center-79cf55b49b-6wrh9
```

```
[brusmi/cee] cee#
```



Nota: En el ejemplo mencionado anteriormente, el espacio de nombres CEE es "cee-
cee". Debe reemplazar este nombre en caso de que lo necesite.

3.2 Genere el ID de paquete del TAC para hacer referencia a los archivos de recopilación recuperados:

Comando:

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

Ejemplo:

```
[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

Además, puede incluir filtros adicionales como namespace o pod_name de la siguiente manera:

Comando:

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

Ejemplo:

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

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



Nota: Se recomienda generar una ID de paquete de TAC para un período de tiempo de ranura (1 h o 2 horas como máximo).

3.3 Mostrar el estado de cada servicio:

```
[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!

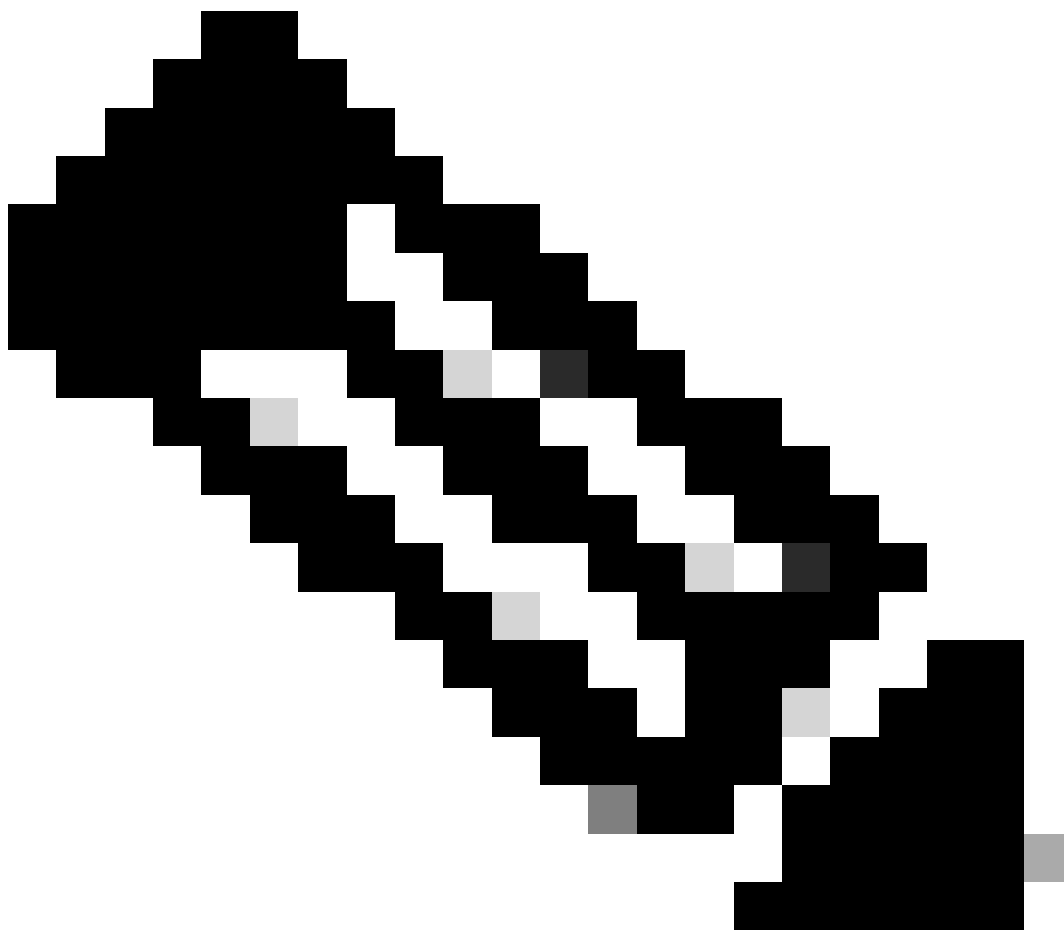
[brusmi/cee] cee#

[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



Nota: Si no hay espacio en disco disponible, elimine los archivos de depuración antiguos.

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

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 Cree una ID de depuración del TAC para recopilar sólo las métricas:

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

4. Descargar depuración de TAC

Actualmente, existen tres opciones diferentes para descargar TAC Debug desde CEE:

4.1 SFTP de Master VIP (menos recomendado, tarda mucho).

4.1.1 Obtenga la URL para descargar los registros recopilados en **tac package ID** :

Comando:

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

Ejemplo:

```
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 Comprimir y obtener el archivo tac-debug de **show-tac-manager** pod:

a. Obtenga el ID del grupo show-tac.

Comando:

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

Ejemplo:

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

b. Ejecute el comando `exec` en `show-tac pod`, y comprima los registros de depuración del TAC.

Comando:

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

Ejemplo:

```
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 1591
```

```
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-bf1rc:/home/tac$ ls
1591662490 1591662529 1592265088 tac-debug_1591662529.tar.gz
```

4.1.3 Copie el archivo en el /tmp directorio del VIP maestro:

Comando:

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

Ejemplo:

```
cisco@brusmi-master1:~$ kubectl cp cee-cee/show-tac-manager-85985946f6-bf1rc:/home/tac/tac-debug_1591662529.tar.gz /tmp/
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 Transferencia de archivos vía sftp desde Master VIP.

4.2 Descargue TAC Debug with wget command (macOS/Ubuntu).

4.2.1 Obtener el link show-tac de la salida "k8s get ingress":

```
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 Ingrese el wget comando desde su terminal de PC:

```
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. Recopile registros de CEE para todos los POD SMF

5.1 Inicio de sesión en el `smf-data` centro de operaciones desde máster k8:

```
cisco@brusmi-master1:~$ ssh -p 2024 admin@$(kubectl get svc -n smf-data | grep ^ops-center | awk '{print $2}')
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
```

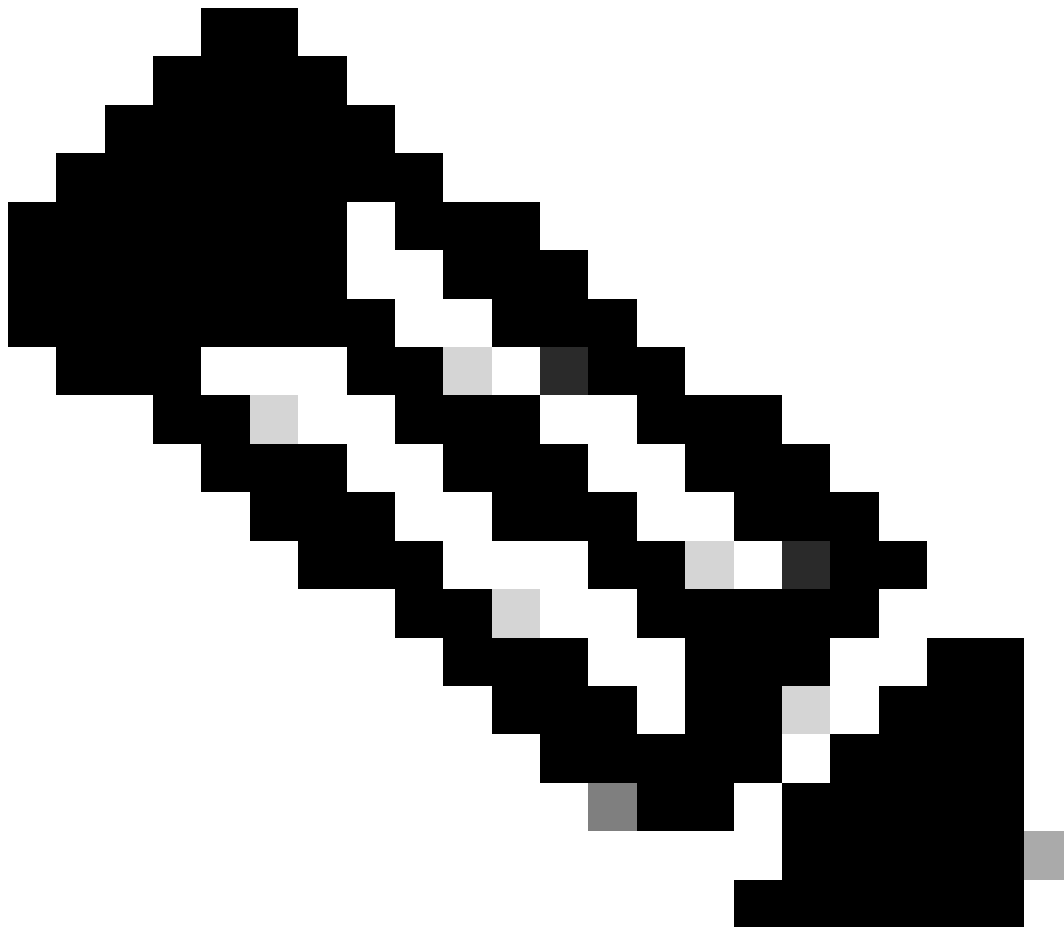
5.2 Confirme si "aplicación de nivel de registro" está habilitada:

```
[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 Inicio de sesión en el centro de operaciones Cee desde máster k8:

```
cisco@brusmi-master1:~$ ssh -p 2024 admin@$(kubectl get svc -n cee-cee | grep ^ops-center | awk '{print $2}')
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-ops-center-79cf55b49b-6wrh9
```

[brusmi/cee] cee#



Nota: En el ejemplo mencionado anteriormente, el espacio de nombres CEE es "cee-
cee". Debe reemplazar este nombre en caso de que lo necesite.

5.4 Detenga los registros de todos los POD SMF que comiencen por "smf-"(smf-nodemgr, smf-
protocol, smf-rest, smf-service, smf-udp-proxy). Recopile los registros durante unos segundos y utilice Ctrl+C
para detener la recopilación de datos:

```
[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.co

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



Nota: Puede ser más específico en caso de que necesite recopilar registros de un grupo de dispositivos, un contenedor o varios grupos de dispositivos en particular.

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-pro
```

6. Acceso a Grafana

6.1 Obtener la URL para acceder a 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 Abra una página web con HTTPS de la siguiente manera:

<https://grafana.192.168.208.10.xxx.x>

Acerca de esta traducción

Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).