Procedure for CNDP PCF Site Isolation and Restoration

Contents

Introduction

This document describes the Cloud Native Deployment Platform (CNDP) Policy Control Function (PCF) site isolation and restoration.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- Policy Control Function
- Kubernetes

Note: Cisco recommends that you must have privilege root user access to CPS CLI.

Components Used

The information in this document is based on these software and hardware versions:

- PCF
- Kubernetes

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

PCF is normally deployed with two PCF sites to form a Geographically Redundant Pair. For Geo Replication (GR), you have to create two separate High Availability (HA) PCF systems independently and configure the Geo HA for communications with the remote sites.

PCF has many external interfaces to handle ingress and egress traffic to and from PCF, which includes N7, N28, Rx, and Lightweight Directory Access Protocol (LDAP) to handle Rest, diameter, and LDAP traffic.

Problem

When you perform any planned activities (for example, upgrade and more) or encounter any issues with one PCF site that causes traffic impact, which requires some time for resolution, it is required to isolate the respective PCF site from traffic to avoid any business impact.

Once the activity is finished or the PCF issue gets resolved, you must restore the site and introduce the traffic.

Procedure to Isolate and Restore PCF Site

PCF Site Isolation

Step 1. Set the system to shutdown mode.

Step 1.1. From Master-1 of the site that is isolated, login to the PCF ops center.

```
ssh -p 2024 admin@`kubectl get svc -A | grep " ops-center-pcf" | awk '{print $4}'`
```

Step 1.2. Configure the PCF Registration Status UNDISCOVERABLE.

It is required to update PCF registration status as UNDISCOVERABLE at Network Repository Function (NRF) to prevent N7 messages that flow from SMF to respective PCF, which in turn re-routes the N7 traffic towards Geo Redundant mated site.

In order to configure the PCF registration status to undiscoverable, use this configuration from the PCF Ops Center of the primary site:

```
config
service-registration
profile
nf-status UNDISCOVERABLE
top
commit
end
```

Note: Wait for a minute or two and perform the next steps.

- config – Enters the configuration mode.
- service-registration – Enters the service registration configuration mode.
- profile – Enters the profile configuration mode.
- nf-status { REGISTERED | UNDISCOVERABLE } – Specifies the PCF registration status. For

the site isolation feature, set the status to UNDISCOVERABLE. In this state, all the operations that involve the PCF instance are suspended.

Step 1.3. Configure the system to **shutdown** mode.

```
[pcf01/pcfapp] pcf# config terminal
Entering configuration mode terminal
[pcf01/pcfapp] pcf(config)# system mode shutdown
[pcf01/pcfapp] pcf(config)# commit
Commit complete.
```

Wait for the system to run to 100%.

Step 1.4. Verify that the system deployed status is false.

```
[pcf01/pcfapp] pcf# show system status
system status deployed false
system status percent-ready 100.0
```

Step 1.5. Retrieve the site-id of the system which is shut down.

```
[pcf01/pcfapp] pcf# show running-config cdl system-id
cdl system-id {siteID}
```

Step 2. CDL timer expiry notification configuration.

Step 2.1. Connect to master-1 of the Active site (mated site) and connect to the PCF ops center.

```
ssh -p 2024 admin@`kubectl get svc -A | grep " ops-center-pcf" | awk '{print $4}'`
```

Step 2.2. Configure the Active site CDL to send timer expiry notifications for the Isolated site.

```
[pcf01/pcfapp] pcf# config terminal
Entering configuration mode terminal
[pcf01/pcfapp] pcf(config)# cdl datastore session
[pcf01/pcfapp] pcf(config-datastore-session)# slot notification remote-system-id [ siteID ]
[pcf01/pcfapp] pcf(config-datastore-session)# commit
Commit complete.
```

Note: siteID is the id retrieved from the isolation site at Step 1.5.

PCF Site Restoration

Step 1. Disable CDL timer expiry notification configuration.

Step 1.1. Connect to master-1 of the Active site and connect to the PCF ops center.

```
ssh -p 2024 admin@`kubectl get svc -A | grep " ops-center-pcf" | awk '{print $4}'`
```

Step 2.1. CDL must be configured so that it does not send timer expiry notifications to the isolated site.

```
[pcf01/pcfapp] pcf# config terminal
```

```
Entering configuration mode terminal
[pcf01/pcfapp] pcf(config)# no cdl datastore session slot notification remote-system-id
[pcf01/pcfapp] pcf(config-datastore-session)# commit
Commit complete.
```

Step 2. Set PCF KAFKA OFFSET.

It’s a required action to set the Kafka Pods with the latest OFFSET to maintain CDL Session integrity and synchronization. Run these steps from the Active PCF Site before you attempt to take the other PCF Site into the active state.

Step 2.1. From Master-1 of the Active site, retrieve the Kafka pods.

```
cloud-user@pcf01-master1:~$ kubectl get pods -A | grep -i kafka
pcf-pcfapp kafka-0 2/2 Running 0 22m
pcf-pcfapp kafka-1 2/2 Running 0 20m
pcf-pcfapp kafka-2 2/2 Running 0 20m
```

Step 2.2. Login to Kafka-0 pod.

```
kubectl exec -it -n pcf-pcfapp kafka-0 bash
```

Step 2.3. Run a list command to find the consumer groups in the Kafka Groups.

```
kafka@kafka-0:/opt/kafka$ cd bin
kafka@kafka-0:/opt/kafka/bin$ ./kafka-consumer-groups.sh --list --bootstrap-server
localhost:9092
test-group
c1-c2-consumer-group
```

Step 2.4. Get the description of the consumer groups in Kafka. Ensure to use the correct consumer group name from the output from Step 2.3.

```
kafka@kafka-0:/opt/kafka/bin$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
describe --group c1-c2-consumer-group
```

Expected Output:

```
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
c1-c2-consumer-group kv.kafka.shard.1.1.1 0 1774202721 1774213158 10437 c1-c2-consumer-group-0-
65c85cd5-f43d-4767-971a-f8b53164538a /xx.xx.xx.xx c1-c2-consumer-group-0
c1-c2-consumer-group kv.kafka.shard.1.1.9 0 1638393629 1638393987 358 c1-c2-consumer-group-3-
2822cebd-5c98-4dbd-8d49-31d4b80bd415 /xx.xx.xx.xx c1-c2-consumer-group-3
c1-c2-consumer-group kv.kafka.shard.1.1.6 0 1718659693 1718660429 736
```

Step 2.5. Check the latest new OFFSET values.

```
kafka@kafka-0:/opt/kafka/bin$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
reset-offsets --group c1-c2-consumer-group --all-topics --to-latest --dry-run
```

Expected Output:

```
GROUP TOPIC PARTITION New-OFFSET
c1-c2-consumer-group kv.kafka.shard.1.1.1 0 1774213158
c1-c2-consumer-group kv.kafka.shard.1.1.9 0 1638393987
c1-c2-consumer-group kv.kafka.shard.1.1.6 0 1718660429
c1-c2-consumer-group kv.kafka.shard.1.1.2 0 1913886111
```

Step 2.6. Reset the OFFSET to the latest new values.

```
kafka@kafka-0:/opt/kafka/bin$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --reset-offsets --group c1-c2-consumer-group --all-topics --to-latest --execute
```

Expected Output:

```
GROUP TOPIC PARTITION New-OFFSET
c1-c2-consumer-group kv.kafka.shard.1.1.1 0 1774213158
c1-c2-consumer-group kv.kafka.shard.1.1.9 0 1638393987
c1-c2-consumer-group kv.kafka.shard.1.1.6 0 1718660429
```

Step 2.7. Check the current lag values.

```
kafka@kafka-0:/opt/kafka/bin$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group c1-c2-consumer-group
```

Expected Output:

```
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
c1-c2-consumer-group kv.kafka.shard.1.1.1 0 1774202721 1774213158 10437 c1-c2-consumer-group-0-65c85cd5-f43d-4767-971a-f8b53164538a /xx.xx.xx.xx c1-c2-consumer-group-0
c1-c2-consumer-group kv.kafka.shard.1.1.9 0 1638393629 1638393987 358 c1-c2-consumer-group-3-2822cebd-5c98-4dbd-8d49-31d4b80bd415 /xx.xx.xx.xx c1-c2-consumer-group-3
```

Step 3. Set the system to **Running** mode

Step 3.1. Open four terminals connected to the isolated site. Master-1 of the site is down.
Step 3.2. On the first terminal. ensure the script **/home/cloud-user/rs_0.sh** is on the master node.

```
ls -lrt /home/cloud-user/rs_0.sh
```

Step 3.3. On the second terminal run this command which is responsible to terminate rest-ep pods. Please ensure to use the correct namespace.

```
watch kubectl scale --replicas=0 deployment/pcf-rest-ep -n pcf-pcf
```

Step 3.4. Run this script which is responsible to terminate Rx diameter pods on the third terminal.

```
watch ./rs_0.sh
```

Step 3.5 Set the system to **running** mode from the PCF ops center on the fourth terminal.

```
[pcf01/pcf01] pcf#
[pcf01/pcf01] pcf# config
Entering configuration mode terminal
[pcf01/pcf01] pcf(config)# system mode running
[pcf01/pcf01] pcf(config)# commit
Commit complete.
```

Wait for the system to run to 100%.

Step 3.6. Now ensure that neither rest-ep nor Rx diameter is run.

```
cloud-user@pcf01-master-1:~$ kubectl get pods -A | egrep "diameter|rest-ep"
```

Step 3.7. Connect to Master-1 of both the sites and retrieve the remote site db-endpoint IP address (replication IP address for the mated site).

```
ssh -p 2024 admin@`kubectl get svc -A | grep " ops-center-pcf" | awk '{print $4}'` 'show
running-config | inc "db-endpoint host"'
```

Expected Output:

```
db-endpoint host xx.xx.xx.xx
```

Step 3.8 Check the number of connections between the CDL-EP and mated site Replication IP (there must be 5 connections).

```
for CDLEP in `kubectl get pods -A | grep cdl-ep | awk '{print $2}'`;do echo $CDLEP; kubectl exec
-it $CDLEP -n `kubectl get namespaces | grep "pcf-" | awk '{print $1}'` -- netstat -anp | grep
10.169.149.34| wc -l ; done
```

Expected Output:

```
cdl-ep-session-c1-d0-56995765b5-l2kz6
5
cdl-ep-session-c1-d0-56995765b5-mlxdx
5
```

Step 3.9. Verify there are not any recent “Connection to remote systemID has been lost” error messages at the CDL-EP.

```
for CDLEP in `kubectl get pods -A | grep cdl-ep | awk '{print $2}'`;do echo $CDLEP; kubectl logs
$CDLEP -n `kubectl get namespaces | grep "pcf-" | awk '{print $1}'` --since=15m| grep "has been
lost" ; done
```

The expected output in the clean state:

```
cdl-ep-session-c1-d0-56995765b5-l2kz6
cdl-ep-session-c1-d0-56995765b5-mlxdx
cdl-ep-session-c1-d0-56995765b5-nptr9
cdl-ep-session-c1-d0-56995765b5-rm7hh
```

Expected output if there is a problem:

```
2022/06/24 22:21:08.242 [ERROR] [RemoteEndointConnection.go:619] [datastore.ep.session]
Connection to remote systemID 2 has been lost
```

Step 3.10. Ensure sure all other pods run fine without any problems.

```
cloud-user@pcf01-master-1:~$ kubectl get pods -A
```

Step 3.11. Monitor the CDLs grafana graph and ensure the statistics show successful create/update stats.

Step 3.12. After a couple of minutes ensure the CDLs get in sync.

```
cloud-user@pcf01-master-1:~$ for i in `kubectl get pods -A | awk '{print $2}' | grep cdl-ep` ;
```

```
do echo $i ; kubectl exec -it $i -n `kubectl get namespaces | grep pcf- | awk '{print $1}'` --
./verify_geo_sync ; done
```

Expected Output:

```
2022/03/05 02:31:56 Geo sync is successful
```

Step 3.13. From the peer site, verify that the mirror maker is up and **running**.

```
pcf-pcf01 mirror-maker-0 1/1 Running 1 24d
```

Step 3.14. Interrupt the script on the other 3 terminals of the site which is just brought up.

Step 3.15. Run this script to recreate PCF Rx diameter pods.

```
./rs_1.sh
```

Step 3.16. Run this command to recreate the PCF rest-ep Pods.

> **Note**: Check site replicas details for a number of rest-ep replicas and you must use the correct namespace.

```
kubectl scale --replicas=8 deployment/pcf-rest-ep -n pcf-pcf
```

Step 3.17. After it is finished, ensure that the rest-ep or Rx diameter is run.

```
cloud-user@pcf01-master-1:~$ kubectl get pods -A | egrep "diameter|rest-ep|ldap"
pcf-pcf01 diameter-ep-rx-rx-584cd76c75-kwmhh1/1 Running 0 2m
pcf-pcf01 diameter-ep-rx2-rx-64cd75b7f6-drjrz 1/1 Running 0 2m
pcf-pcf01 diameter-ep-rx3-rx-544d4f9bf7-gfb9c 1/1 Running 0 2m
pcf-pcf01 ldap-pcf-pcf01-cps-ldap-ep-5884c6d76d-5tchw 1/1 Running 0 2m
pcf-pcf01 ldap-pcf-pcf01-cps-ldap-ep-5884c6d76d-6wtnm 1/1 Running 0 2m
pcf-pcf01 pcf-rest-ep-86b546f9db-5wzp6 1/1 Running 0 2m
pcf-pcf01 pcf-rest-ep-86b546f9db-6prmd 1/1 Running 0 2m
pcf-pcf01 pcf-rest-ep-86b546f9db-6pstm 1/1 Running 0 2m
pcf-pcf01 pcf-rest-ep-86b546f9db-dsz6c 1/1 Running 0 2m
pcf-pcf01 pcf-rest-ep-86b546f9db-dzlkw 1/1 Running 0 2m
```

Step 3.18. On the fourth terminal, configure the PCF Registration Status REGISTERED.

Once the activity is completed and the issue gets resolved, it is required to update the PCF registration status as REGISTERED at Network Repository Function (NRF) to allow N7 messages to flow from SMF to respective PCF.

In order to configure the PCF registration status to REGISTERED, use this configuration from the PCF Ops Center of the primary site:

```
config
service-registration
profile
nf-status REGISTERED
top
commit
end
```