

Use-case 3: Planned maintenance events for SAP

This use case, we will perform planned maintenance event in SAP application cluster. The seamless workflow integration enables teams to diagnose issues without disruptive context switching by viewing CloudWatch metrics directly in their IDE while using Q CLI to generate precise remediation commands.

 Prompt: **Check the status of the SAP ASCS/ERS cluster for the SAPDemo SAP application without restarting any services.**

> I'll check the status of the SAP ASCS/ERS cluster for the SAPDemo SAP application without restarting any services.

 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ec2 describe-instances --filters "Name=tag:Name,Values=*SAP-FIS-CS*" --query "Reservations[*].Instances[*].[InstanceId,Tags[?Key=='Name'].Value[0],State.Name,InstanceType,PrivateIpAddress]" --output table
```

:

↳ Purpose: Checking EC2 instances that are part of the SAP cluster

```
-----  
| DescribeInstances |  
+-----+-----+-----+-----+  
| i-064751ba1b4c8340f | SAP-FIS-CS-sapcnod2 | running | c5a.xlarge | 10.0.155.180 |  
| i-0e08f883d5c3b3747 | SAP-FIS-CS-sapcnod1 | running | c5a.xlarge | 10.0.134.80 |  
+-----+-----+-----+-----+  
:
```

:

• Completed in 1.666s

> Now let's check the cluster status on these nodes:

 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm_mon -1"]' --output json
```

:

↳ Purpose: Checking cluster status on node 1

> Let's check the SAP processes on both cluster nodes to see the status of ASCS and ERS:

 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["ps -ef | grep -i \"sap\\\\|ascs\\\\|ers\\\\\" | grep -v grep"]' --output json
```

:

↳ Purpose: Checking SAP processes on cluster node 1

> Now let's check the processes on the second node:

 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-064751ba1b4c8340f --document-name "AWS-RunShellScript" --parameters 'commands=["ps -ef | grep -i \"sap\\\\|ascs\\\\|ers\\\\\" | grep -v grep"]' --output json
```

```
:
└─ Purpose: Checking SAP processes on cluster node 2

> Let's check the status of the ERS instance specifically:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo su - ecdadm -c \"sapcontrol -nr 11 -function GetProcessList\""]' --output json
:
└─ Purpose: Checking SAP ERS status on node 1

> Let's check the ASCS status on node 2:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-064751ba1b4c8340f --document-name "AWS-RunShellScript" --parameters
'commands=["sudo su - ecdadm -c \"sapcontrol -nr 10 -function GetProcessList\""]' --output json
:
└─ Purpose: Checking SAP ASCS status on node 2

> Let's check the cluster logs to see if there are any recent issues:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo tail -20 /var/log/cluster/corosync.log"]' --output json
:
└─ Purpose: Checking recent cluster logs

> Let's check the pacemaker logs for more recent information:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo tail -20 /var/log/pacemaker/pacemaker.log"]' --output json
:
└─ Purpose: Checking recent pacemaker logs

> Let's check the cluster resource status in more detail:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo crm resource status"]' --output json
:
└─ Purpose: Checking cluster resource status

..

> Let's check if there are any constraints on the ERS resource:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo crm configure show | grep -A 10 constraints"]' --output json
:
```

↳ Purpose: Checking cluster constraints

..

> Let's try to get the full cluster configuration:

✂ Using tool: execute_bash (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm configure show"]' --output json
```

:

↳ Purpose: Checking full cluster configuration

..

> Based on all the information gathered, I'll provide you with a detailed status report on the SAP ASCS/ERS cluster for the SAPDemo SAP application:

SAP ASCS/ERS Cluster Status Report for SAPDemo

Cluster Overview

The SAPDemo SAP application is running in a high-availability configuration with two cluster nodes:

- **Node 1**: sapcnode1 (i-0e08f883d5c3b3747)
- **Node 2**: sapcnode2 (i-064751ba1b4c8340f)

Current Cluster Status

- **Cluster Stack**: corosync
- **Current DC (Designated Coordinator)**: sapcnode2
- **Quorum Status**: Partition with quorum (cluster has voting majority)
- **Last Configuration Change**: Sunday, July 13, 2025, 16:44:03 by ecdadm
- **Nodes Configured**: 2
- **Resource Instances**: 5 (1 DISABLED)
- **Node Status**: Both nodes are online

Resource Distribution

The SAP services are distributed across the cluster nodes as follows:

ASCS Resources (Primary SAP Central Services)

- **Resource Group**: grp_ECD_ASCS10
- **Currently Running On**: sapcnode2
- **Resources**:
 - rsc_ip_ECD_ASCS10 (Virtual IP: 192.168.48.201): Started on sapcnode2
 - rsc_sap_ECD_ASCS10 (SAP ASCS Instance): Started on sapcnode2
- **Status**: GREEN/Running
- **Processes**:
 - Message Server (ms.sapECD_ASCS10): Running (PID: 317462)
 - Enqueue Server (en.sapECD_ASCS10): Running (PID: 317463)

ERS Resources (Enqueue Replication Server)

- **Resource Group**: grp_ECD_ERS11
- **Currently Running On**: sapcnode1 (partially)
- **Resources**:
 - rsc_ip_ECD_ERS11 (Virtual IP: 192.168.48.202): Started on sapcnode1
 - rsc_sap_ECD_ERS11 (SAP ERS Instance): Stopped (disabled)
- **Status**: GRAY/Stopped
- **Processes**: EnqueueReplicator is stopped

Issue Identified

The ERS instance (rsc_sap_ECD_ERS11) is currently disabled in the cluster configuration. This is evident from:

1. The cluster status showing "1 DISABLED" resource
2. The resource status showing "rsc_sap_ECD_ERS11 (ocf::heartbeat:SAPInstance): Stopped (disabled)"
3. The cluster configuration showing "target-role=Stopped" for the ERS resource
4. The pacemaker logs showing "SAP Instance ECD-ERS11 stopped" at 16:44:14

Cluster Configuration Details

- **Colocation Constraint**: col_sap_ECD_no_both (-5000) prevents ASCS and ERS from running on the same node
- **Location Constraint**: loc_sap_ECD_failover_to_ers gives preference to nodes running ERS for ASCS failover
- **Order Constraint**: ord_sap_ECD_first_start_ascs ensures ASCS starts before ERS stops

```

• **STONITH**: Enabled using AWS EC2 fencing mechanism

## Historical Issues
There is one historical issue noted in the cluster status:
• **Failed Fencing Action**: A fencing operation (turning off of sapcnode2) failed on July 11, 2025. This appears to be a historical issue and doesn't affect current operations.

## Conclusion
The SAP ASCS/ERS cluster for the SAPDemo application is partially operational:
• The ASCS instance is running correctly on sapcnode2
• The ERS instance is intentionally disabled (target-role=Stopped in the configuration)
• The virtual IP for ERS is still active on sapcnode1
• Both cluster nodes are online and the cluster has quorum

This configuration appears to be intentional based on the cluster configuration, but it means that enqueue replication is not currently active. In a normal high-availability setup, both ASCS and ERS would be running to ensure enqueue table replication in case of a failover.

>

```

In this next scenario, we use CloudWatch MCP Server to investigate the pacemaker cluster logs to determine the status of SAP ASCS. CloudWatch MCP server invokes tools such as `describe_log_groups`, `analyze_log_groups`, and `execute_log_insights_query` for analysis.

 Prompt: **Check CloudWatch logs related to SAPDemo application for SAP ASCS ERS Pacemaker cluster events, alarms and issues**

```

> Check CloudWatch logs related to SAPDemo application for SAP ASCS ERS Pacemaker cluster events, alarms and issues. only use cloudwatch mcp server for this analysis.

> I'll analyze the CloudWatch logs related to the SAPDemo application for SAP ASCS ERS Pacemaker cluster events, alarms, and issues using only the CloudWatch MCP server.

Let's start by identifying the relevant log groups for the SAPDemo application:

✂ Using tool: describe_log_groups (trusted) from mcp server awslabscloudwatch_mcp_server
:
• Running describe_log_groups with the param:
: {
:   "arguments": {
:     "log_group_name_prefix": "PACEMAKER",
:     "region": "us-east-1"
:   },
:   "name": "describe_log_groups"
: }
:
• Completed in 0.760s

> Now, let's analyze the PACEMAKER_HIGH_AVAILABILITY log group for the SAPDemo application to look for ASCS and ERS related events:

✂ Using tool: analyze_log_group (trusted) from mcp server awslabscloudwatch_mcp_server
:
• Running analyze_log_group with the param:
: {
:   "arguments": {

```

```
: "end_time": "2025-07-13T22:00:00+00:00",
: "log_group_arn": "arn:aws:logs:us-east-1:930487150879:log-group:PACEMAKER_HIGH_AVAILABILITY-
LaunchWizard-SapNWOnHanaHA-SAPDemo",
: "region": "us-east-1",
: "start_time": "2025-07-06T00:00:00+00:00"
: },
: "name": "analyze_log_group"
: }
```

```
:
• Completed in 11.798s
```

> Now, let's search for specific events related to the ASCS and ERS components:

✂ Using tool: `execute_log_insights_query` (trusted) from mcp server `awslabscldwatch_mcp_server`

```
:
• Running execute_log_insights_query with the param:
: {
:   "arguments": {
:     "end_time": "2025-07-13T22:00:00+00:00",
:     "log_group_names": [
:       "PACEMAKER_HIGH_AVAILABILITY-LaunchWizard-SapNWOnHanaHA-SAPDemo"
:     ],
:     "query_string": "fields @timestamp, @message\n| filter @message like /ASCS/ or @message like /ERS/ or
@message like /rsc_sap_ECD_ASCS/ or @message like /rsc_sap_ECD_ERS/\n| filter @message like
/error[fail|warning|critical|Unexpected|OFFLINE|UNCLEAN|transition|stonith/\n| sort @timestamp desc\n| limit 30",
:     "region": "us-east-1",
:     "start_time": "2025-07-06T00:00:00+00:00"
:   },
:   "name": "execute_log_insights_query"
: }
```

```
:
• Completed in 6.572s
```

> Let's also check for any ERS-related issues in the logs:

✂ Using tool: `execute_log_insights_query` (trusted) from mcp server `awslabscldwatch_mcp_server`

```
:
• Running execute_log_insights_query with the param:
: {
:   "arguments": {
:     "end_time": "2025-07-13T22:00:00+00:00",
:     "log_group_names": [
:       "PACEMAKER_HIGH_AVAILABILITY-LaunchWizard-SapNWOnHanaHA-SAPDemo"
:     ],
:     "query_string": "fields @timestamp, @message\n| filter @message like /enrepserver/ or @message like /ERS/ or
@message like /rsc_sap_ECD_ERS/\n| filter @message like
/error[fail|warning|critical|Unexpected|OFFLINE|UNCLEAN|transition|stonith|not running|GRAY/\n| sort @timestamp
desc\n| limit 20",
:     "region": "us-east-1",
:     "start_time": "2025-07-06T00:00:00+00:00"
:   },
:   "name": "execute_log_insights_query"
: }
```

```
:
• Completed in 7.578s
```

> Let me try with a more focused query:

✂ Using tool: `execute_log_insights_query` (trusted) from mcp server `awslabscldwatch_mcp_server`

```
:
• Running execute_log_insights_query with the param:
: {
:   "arguments": {
```

```
: "end_time": "2025-07-13T22:00:00+00:00",
: "log_group_names": [
:   "PACEMAKER_HIGH_AVAILABILITY-LaunchWizard-SapNWOOnHanaHA-SAPDemo"
: ],
: "query_string": "fields @timestamp, @message\n| filter @message like /enrepserver/ or @message like /ERS/ or\n@message like /rsc_sap_ECD_ERS/\n| filter @message like\n/error|fail|warning|critical|Unexpected|OFFLINE|UNCLEAN|transition|stonith|not running|GRAY/\n| sort @timestamp\ndesc\n| limit 20",
: "region": "us-east-1",
: "start_time": "2025-07-13T18:00:00+00:00"
: },
: "name": "execute_log_insights_query"
: }

:
• Completed in 2.848s
```

The context window has overflowed, summarizing the history...

✓ Conversation history has been compacted successfully!

> I'll analyze these CloudWatch logs for the SAP cluster activity, focusing on the ERS component that was previously identified as having issues in our conversation.

SAP ERS Component Status Analysis

Based on the logs from July 13, 2025, I can see the activity related to the SAP ERS (Enqueue Replication Server) component in the SAPDemo environment. These logs show the recent cluster operations for the rsc_sap_ECD_ERS11 resource.

Key Events Timeline (July 13, 2025):

- 20:14:14 - The cluster cleared failure counts for the ERS component on node 2:
 - Deleted failure records: status-2-fail-count-rsc_sap_ECD_ERS11.start_0
 - Deleted last failure record: status-2-last-failure-rsc_sap_ECD_ERS11.start_0
- 20:14:16 - Confirmed ERS was not running on sapclnode2:
 - Transition 352 action 4 (rsc_sap_ECD_ERS11_monitor_0 on sapclnode2) confirmed: not running | rc=7 call-id=459
- 20:15:18 - IP resource migration:
 - Stopped IP resource (rsc_ip_ECD_ERS11) on node 2
 - Started IP resource on sapclnode1
- 20:16:49 - ERS resource configuration change:
 - Transition 358 aborted by rsc_sap_ECD_ERS11-meta_attributes-target-role doing modify target-role=Started
 - Requested start of ERS on sapclnode1: Requesting local execution of start operation for rsc_sap_ECD_ERS11 on sapclnode1
- 20:17:10 - ERS successfully started on sapclnode1:
 - @transition-magic=0;0;17:359:0:388f78c8-909c-49ff-bc39-746815d865d6, @call-id=588, @rc-code=0, @op-status=0, @exec-time=21627
 - Started monitoring the ERS service: Requesting local execution of monitor operation for rsc_sap_ECD_ERS11 on sapclnode1
- 20:17:11 - ERS monitoring confirmed successful:
 - @transition-magic=0;0;19:360:0:388f78c8-909c-49ff-bc39-746815d865d6, @call-id=589, @rc-code=0, @op-status=0, @exec-time=793

Analysis:

This log data shows that the ERS component (rsc_sap_ECD_ERS11) has been successfully started on sapclnode1. The cluster first cleared previous failure records, then migrated the IP resource (rsc_ip_ECD_ERS11) from node 2 to node 1, and finally started the ERS service on node 1.

The successful start is confirmed by the @rc-code=0 in the logs, indicating no errors. The monitoring operation is also successful, showing that the service is running properly.

This aligns with our previous findings where we had identified issues with the ERS component and cleared CloudWatch alarms for it. The logs now confirm that the ERS service has been successfully started and is being monitored by the cluster, which explains why the service is now running despite previous alarms.

Would you like me to analyze any specific aspect of these logs in more detail or provide recommendations for ongoing monitoring of the ERS component?

>

🗨️ Prompt: **start SAP ERS service on node sapclnode1**

> start SAP ERS services on node sapclnode1

> I'll start the SAP ERS services on node sapclnode1.

🔧 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm resource status rsc_sap_ECD_ERS11"]' --output json
```

:

↳ Purpose: Checking the current status of the ERS resource

..

> Let's check the result:

🔧 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm get-command-invocation --command-id a5b77536-b1aa-4113-ab62-fda8309a0acb --instance-id i-0e08f883d5c3b3747 --output json
```

:

↳ Purpose: Checking the result of the ERS status command

..

> Now I'll enable the ERS resource by changing its target-role attribute:

🔧 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm resource meta rsc_sap_ECD_ERS11 target-role=Started"]' --output json
```

:

↳ Purpose: Enabling the ERS resource

..

> Let me try with the correct syntax:

🔧 Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm resource update rsc_sap_ECD_ERS11 meta target-role=Started"]' --output json
```

:

↳ Purpose: Enabling the ERS resource with correct syntax

..

> Let me try with the correct command for this version of Pacemaker:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm configure edit rsc_sap_ECD_ERS11"]' --output json
```

:

↳ Purpose: Enabling the ERS resource with correct command

..

> Let me try a different approach using a temporary file:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm configure show rsc_sap_ECD_ERS11 > /tmp/ers_config.txt && sudo sed -i \"/s/target-role=Stopped/target-role=Started/g\" /tmp/ers_config.txt && cat /tmp/ers_config.txt"]' --output json
```

:

↳ Purpose: Creating a temporary file with the updated resource configuration

..

> Now let's apply the updated configuration:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm configure load update /tmp/ers_config.txt"]' --output json
```

:

↳ Purpose: Applying the updated resource configuration

..

> Now let's check the status of the cluster to see if the ERS resource is starting:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo crm_mon -1"]' --output json
```

:

↳ Purpose: Checking cluster status after enabling ERS

..

> Great! Now let's check the status of the SAP ERS processes specifically:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters 'commands=["sudo su - ecdadm -c \"sapcontrol -nr 11 -function GetProcessList\""]' --output json
```

:

↳ Purpose: Checking SAP ERS processes

..

> Now let's check the enqueue replication state on the ASCS instance:

✂ Using tool: `execute_bash` (trusted)

:

• I will run the following shell command:

```
aws ssm send-command --instance-ids i-064751ba1b4c8340f --document-name "AWS-RunShellScript" --parameters 'commands=["sudo su - ecdadm -c \"sapcontrol -nr 10 -function EnqGetStatistic | grep -i repl\""]' --output json
```

```

:
└─ Purpose: Checking enqueue replication state on ASCS
..

> Let's check the system instance list to confirm all components are running:

✂ Using tool: execute_bash (trusted)
:
• I will run the following shell command:
aws ssm send-command --instance-ids i-0e08f883d5c3b3747 --document-name "AWS-RunShellScript" --parameters
'commands=["sudo su - ecdadm -c \"sapcontrol -nr 11 -function GetSystemInstanceList\""]' --output json
:
└─ Purpose: Checking SAP system instance list
..

> I've successfully started the SAP ERS services on node sapclnode1. Here's a summary of the actions taken and the
current status:

# SAP ERS Service Start Report

## Actions Performed
1. Identified that the ERS resource (rsc_sap_ECD_ERS11) was disabled in the cluster configuration with target-
role=Stopped
2. Created a temporary configuration file with the updated setting target-role=Started
3. Applied the updated configuration to the cluster
4. Verified that the cluster started the ERS resource on sapclnode1

## Current Status

### Cluster Status
• **Cluster State**: Healthy with quorum
• **Last Change**: Sunday, July 13, 2025, 18:03:19 by root via cibadmin on sapclnode1
• **Nodes Online**: 2 (sapclnode1 and sapclnode2)
• **Resources Configured**: 5 (all active)

### Resource Distribution
• **ASCS Resources** (grp_ECD_ASCS10):
  • Virtual IP (rsc_ip_ECD_ASCS10): Running on sapclnode2
  • SAP ASCS Instance (rsc_sap_ECD_ASCS10): Running on sapclnode2
• **ERS Resources** (grp_ECD_ERS11):
  • Virtual IP (rsc_ip_ECD_ERS11): Running on sapclnode1
  • SAP ERS Instance (rsc_sap_ECD_ERS11): Running on sapclnode1

### SAP ERS Process Status
• **Process Name**: enrepserver (Enqueue Replicator)
• **Status**: GREEN/Running
• **Start Time**: July 13, 2025, 18:03:23
• **Process ID**: 362784

### Enqueue Replication Status
• **Replication State**: GREEN
• The enqueue replication is now active and functioning correctly

### System Instance List
The SAP system now shows all components as operational:
• Central Services (ASCS) on ecdascnode: GREEN
• Enqueue Replication Server (ERS) on ecdersnode: GREEN
• Application Server on sapbaseas1: GREEN
• Application Server on sapfiswork: GREEN

## Conclusion
The SAP ERS services have been successfully started on node sapclnode1. The enqueue replication is now active and
functioning correctly, providing high availability for the SAP enqueue service. The cluster is now in a fully operational
state with all resources running as expected.

```