# IBM Db2 SAP Guides

## SAP Guides

# IBM Db2 SAP Guides: SAP Guides

# Table of Contents

# IBM DB2 SAP Guides

This section of the SAP on AWS technical documentation provides instructions on how to set up Amazon Web Services resources to deploy IBM Db2 High Availability Disaster Recovery (HADR) with Pacemaker for SAP NetWeaver on Amazon Elastic Compute Cloud (Amazon EC2) instances. This section includes the following guides:



For information about specific SAP products, see the following sections of this documentation set:

- SAP technical guides
- SAP HANA technical guides
- SAP NetWeaver technical guides
- SAP BusinessObjects technical guides

**About this content set**

SAP on AWS technical documentation provides detailed information on how to migrate, implement, configure, and operate SAP solutions on AWS.

**Additional resources from AWS**

- SAP and AWS: announcements, solutions, support, pricing, FAQ
- Find an AWS SAP partner
- AWS for SAP blog
- Case Studies
- AWS presentations from SAPPHIRE NOW 2018
- Questions and support

**Resources from SAP**

- SAP notes and Knowledge Base articles
- SAP Note 1656250: SAP on AWS: Supported instance types (requires SAP One Support Launchpad user account)

# SAP on AWS – IBM Db2 HADR with Pacemaker

SAP on AWS – IBM Db2 HADR with Pacemaker

Deployment and Operations Guide

December 2020

## About This Guide

This guide provides instructions on how to set up Amazon Web Services resources to deploy IBM Db2 High Availability Disaster Recovery (HADR) with Pacemaker for SAP NetWeaver on Amazon Elastic Compute Cloud (Amazon EC2) instances. This guide is for users who are responsible for planning, architecting and deploying IBM Db2 on AWS for SAP NetWeaver-based applications.

## Overview

Instructions in this document are based on recommendations provided by SAP and IBM on Db2 deployment on Linux via the SAP notes and KB articles listed in Table 1.

*Table 1 - SAP NetWeaver on IBM Db2 OSS Notes*

| SAP OSS Note | Description |
|---|---|
| **1656099** | SAP Applications on AWS: Supported DB/OS and Amazon EC2 products |
| **1656250** | SAP on AWS: Supported instance types |
| **1612105** | DB6: FAQ on Db2 High Availability Disaster Recovery (HADR) |
| **101809** | DB6: Supported Db2 Versions and Fix Pack Levels |
| **1168456** | SAP Db2 support info |
| **1600156** | SAP Db2 support on AWS |

This document follows best practices from AWS, IBM and SAP for SAP NetWeaver deployments on Linux. See the Additional Reading (p. 45) section of this document for more detail.

This guide is part of a content series that provides detailed information about hosting, configuring, and using SAP technologies in the Amazon Web Services Cloud. For the other guides in the series, ranging from overviews to advanced topics, see https://aws.amazon.com/sap/docs/

**What this guide doesn't do**

This document doesn't provide guidance on how to set up network and security constructs like Amazon Virtual Private Cloud (Amazon VPC), subnets, route tables, access control lists (ACLs), Network Address

Translation (NAT) Gateway, AWS Identity and Access Management (IAM) Roles, or AWS Security Groups. It doesn't cover the high availability (HA) setup for the SAP Application Server Central Services/Enqueue Replication Server (ASCS/ERS), and focuses only on the database (DB) layer when covering the single points of failure (SPOF) for the SAP applications.

# Considerations

## Specialized Knowledge

To understand this document, you should have a good understanding of AWS services, general networking concepts, Linux operating systems, and IBM Db2 administration.

Before you follow the instructions in this guide, we recommend that you become familiar with the following AWS services. (If you are new to AWS, see Getting Started with AWS

- Amazon EC2
- Amazon EBS
- Amazon VPC
- AWS CloudFormation
- AWS Systems Manager
- Amazon Simple Storage Service (Amazon S3)
- AWS Identity and Access Management (IAM)

# Technical Requirements

- Before you start the installation and configuration of IAM Db2 High Availability Disaster Recovery (HADR), ensure that you meet the following requirements:
- Your operating system is a supported Red Hat or SUSE version. Check SAP product availability matrix (PAM). Login required.
- Your database version is IBM Db2 10.5 or higher.
- Bring your own license (BYOL) for IBM Db2 and SAP application.
- Install AWS SAP Data Provider on Amazon EC2 instances after installing IBM Db2 database.
- An AWS account with permission to create resources.
- Access to SAP installation media for database and application.
- AWS Business or Enterprise level support (1656250 - SAP on AWS: Support prerequisites). Login required.

# Planning

## Architecture Options

SAP NetWeaver applications based on IBM Db2 can be installed in three different ways:

- **Standard system or single host installation**— In this option, Advanced Business Application Programming (ABAP) Application Server Central Services/System Central Services (ASCS/SCS) and the database primary application server (PAS) of SAP NetWeaver run in a single Amazon EC2 instance. This option is suited for non-critical and non-production workloads.

- **Distributed system**— In distributed systems, ASCS/SCS and the database PAS of SAP NetWeaver can run on separate Amazon EC2 instances. For example, you can choose to run ASCS and PAS on one Amazon EC2 instance, and the database on another Amazon EC2 instance, or other combinations. This option is suited for production and non-production workloads.
- **High availability system**— For your SAP application to be highly available, you will need to protect the single point of failures. The database is one of the single points of failure in SAP applications.

AWS recommends that you deploy primary and standby IBM Db2 databases in different Availability Zones (AZs) within an AWS region. Figure 1 provides a high-level architecture for IBM Db2 high availability in AWS. This option is suited for business-critical applications.

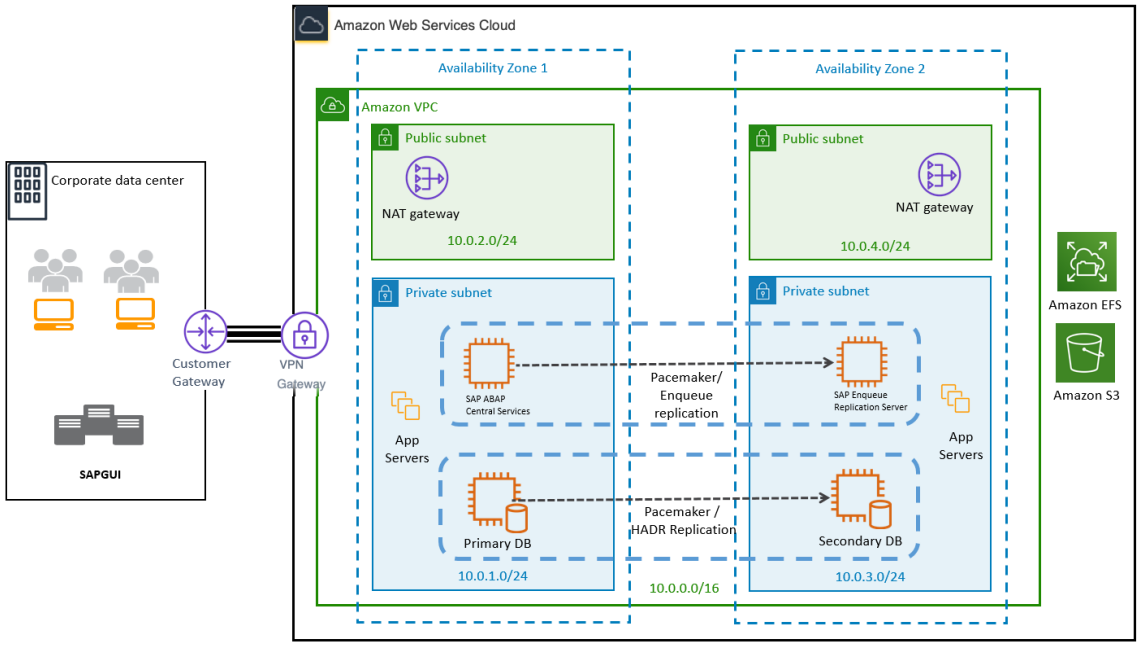


*Figure 1 – High-level HA architecture for SAP with IBM Db2 on AWS*

# Security

AWS provides security capabilities and services to securely run your SAP applications on the AWS platform. In the context of IBM Db2 for SAP applications, you can use network services and features such as Amazon VPC, AWS Virtual Private Network (AWS VPN), AWS Direct Connect, Amazon EC2 Security Groups, network access controls lists (NACLs), route tables, and more to restrict the access to your database.

## Network Security

The databases of SAP applications don't usually require direct user access. The end users access the application using SAP Graphical User Interface (GUI), SAP Web Dispatcher, or SAP Fiori. We recommend that you limit direct access to the EC2 instances to administrators only, for maintenance purpose.

IBM Db2 listens on TCP port 5912 by default. Depending on your VPC design, you should configure Amazon EC2 Security Groups, network and cryptography libraries (NaCls), and route tables to allow traffic to TCP Port 5912 from SAP primary application servers and additional application servers (PAS/AAS) and ABAP SAP Central Services/SAP Central Services (ASCS/SCS). To learn more about configuring the security group, see Security groups for your VPC.

## Encryption

Encryption is a security mechanism that converts plain text (readable data) into ciphertext. AWS offers built-in encryption for Amazon EBS data volumes, boot volumes, and snapshots. The encryption process occurs automatically, and you don't need to manage encryption keys. This mechanism protects your EBS volumes at rest, and data in transit that passes between EC2 servers. This encryption level is offered at no additional cost.

You also can use the native IBM Db2 native database encryption feature if required.

## Sizing

SAP Quick Sizer is used to size SAP environment for new implementations. However, if you are migrating your existing SAP applications based on IBM Db2 to AWS, consider using the following tools to right-size your SAP environment based on current utilization.

- **SAP Early Watch Alerts (EWA)**:—SAP EWA reports are provided by SAP regularly. These reports provide an overview of historical system utilization. Analyze these reports to see if your existing SAP system is over-utilized or under-utilized. Use this information to right-size your environment.
- **Linux native tools**:—Gather and analyze historical utilization data for CPU/Memory to right-size your environment. In case your source is IBM AIX, you can make use of nmon reports as well.
- **AWS Services**— Use services such as AWS Migration Evaluator or AWS Application Discovery Services that help with collecting usage and configuration data about your on-premises servers. Use this information to analyze and right-size your environment.

Because it's easy to scale up or scale down your Amazon EC2 instances on AWS, consider the following while sizing your SAP environment on AWS.

- You don't need to over-provision storage to meet future demand.
- SAP Quick Sizer tools provide sizing guidance based on assumptions that on 100% load (as per your inputs to tool), system utilization will not be more than 65%, so there is some buffer built into SAP Quick Sizer recommendation. See SAP's Quick Sizer guidance for details. (Login required.)

# Operating System

You can deploy your SAP workload on SUSE Linux Enterprise Server (SLES) for SAP, Red Hat Enterprise Linux for SAP with High Availability and Update Services (RHEL for SAP with HA and US), or RHEL for SAP Solutions.

SLES for SAP and RHEL for SAP with HA and US are available in the AWS Marketplace under an hourly or an annual subscription model.

## SLES

SLES for SAP provides additional benefits, including Extended Service Pack Overlap Support (ESPOS), configuration and tuning packages for SAP applications, and High Availability Extensions (HAE). See the SUSE SLES for SAP product page. We strongly recommend using SLES for SAP instead of SLES for all your SAP workloads.

If you plan to use Bring Your Own Subscription (BYOS) images provided by SUSE, ensure that you have the registration code required to register your instance with SUSE to access repositories for software updates.

## RHEL

RHEL for SAP with HA and US provides access to Red Hat Pacemaker cluster software for High Availability, extended update support, and the libraries that are required to configure pacemaker HA. For details, see the RHEL for SAP Offerings on AWS FAQ in the *Red Hat knowledgebase*.

If you plan to use the BYOS model with RHEL, either through the Red Hat Cloud Access program or another means, ensure that you have access to a RHEL for SAP Solutions subscription. For details, see Overview of the Red Hat Enterprise Linux for SAP Solutions subscription in the *Red Hat knowledgebase*.

The correct subscription is required to download the required packages for configuring the Pacemaker cluster.

## Compute

AWS provides a wide array of SAP supported Amazon EC2 instances for your SAP workloads. See SAP Note 1656099 - SAP Applications on AWS: Supported DB/OS and Amazon EC2 products for details. Based on the results of your sizing exercise, you can deploy your IBM Db2 on any of the SAP supported Amazon EC2 instances that meets your requirement.

## Storage

Amazon EBS volumes are designed to be highly available and durable. EBS volume data is replicated across multiple servers in an AZ to prevent the loss of data from the failure of any single component. Due to this built in protection, you don't have to configure RAID 1 for volumes containing database transaction log files and Db2 binaries.

We don't recommend RAID 5 for container files for data, index, or temporary tablespaces on AWS for the following reasons:

- As mentioned previously, volumes are replicated within AZ by default.
- Parity write operations of RAID 5 consume some of the Input/Output Operations Per Second (IOPS) available to your volume and will reduce the overall Input/Output (IO) available for database operations by about 20-30% over RAID 0 configuration.

# Network

Ensure that you have your network constructs set up to deploy resources related to SAP NetWeaver. If you haven't already set up network components like Amazon VPC, subnets, and route tables, you can use AWS Quick Start for VPC to easily deploy scalable VPC architecture. See the AWS Quick Start for VPC reference deployment guide.

See the series of VPC Subnet Zoning Pattern blogs for VPC patterns that you should consider for SAP applications.

# Business Continuity

We recommend that you architect your business-critical applications to be fault tolerant. Depending on your availability requirements, there are different ways to achieve this. In this section we will discuss how you can set up highly available IBM Db2 for SAP applications.

## High Availability

High availability for IBM Db2 database on AWS can be configured with IBM HADR with Pacemaker:

One of the requirements for automated failover with IBM Db2 HADR on AWS is Pacemaker. Implementing a Pacemaker cluster in AWS is similar to deploying it in an on-premises setting. On AWS, you need to deploy the cluster nodes in separate subnets, and we recommend that you have these subnets in different AZs.

Figure 2 provides an overview of architecture for IBM Db2 HADR with Pacemaker on AWS. This includes the following components:

- A VPC configured with two private subnets across two AZs. This provides the network infrastructure for your IBM Db2 deployment.
- In private subnet, Linux servers are configured with Pacemaker to protect the IBM Db2 database.
- Overlay IP address (similar to a virtual IP address) that is relocatable between the primary and standby Db2 databases.

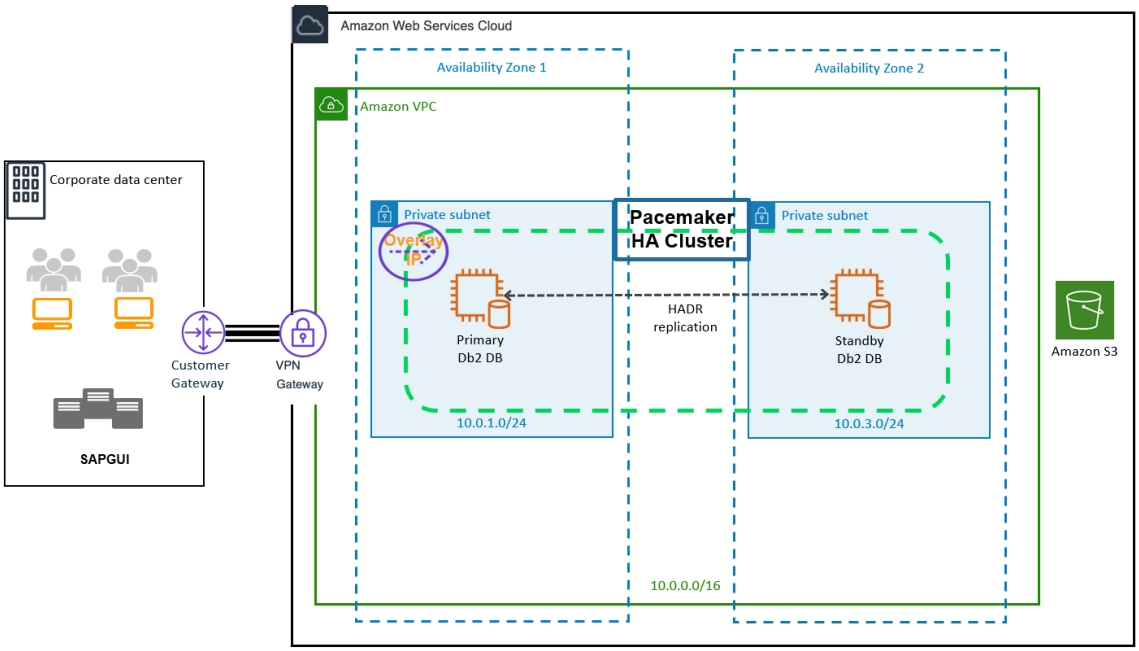


*Figure 2 - IBM Db2 HADR with Pacemaker*

# Deployment

This section discusses high level deployment process and steps. Table 2 lists the steps in the order they should be done, and each step's purpose.

*Table 2 – Steps to set up AWS resources to deploy IBM Db2 HADR with Pacemaker for SAP NetWeaver on Amazon EC2 instances*

| Activity | Purpose |
|---|---|
| **Step 1: Db2 Virtual Hostname** (p. 8) | Decide on the virtual hostname for your Db2 database (for example, `dbhadb2`). |
| **Step 2: AWS Overlay IP** (p. 8) | Decide on the Overlay IP for the dbhadb2 name (for example, `192.168.1.90`). |

| Activity | Purpose |
|---|---|
| **Step 3: Provision AWS Resources** (p. 8) | Provision AWS resources and configure security. |
| **Step 4: SAP App and Db2 Install** (p. 8) | Install SAP tier and Db2 primary and standby databases. |
| **Step 5: Db2 HADR Setup** (p. 12) | Configure Db2 HADR replication. |
| **Step 6: Pacemaker Cluster Setup** (p. 15) | Configure the Pacemaker cluster. |
| **Step 7: Post Setup Configuration** (p. 2) | Post tasks and manual configuration. |
| **Step 8: Testing and Validation** (p. 29) | Quality Assurance. |

# Step 1: Db2 Virtual Hostname

Decide on the virtual hostname for your Db2 database. For example, if your virtual hostname is `dbhadb2`, it would be configured in the SAP and `dbclient` profiles. See Step 7: Post Setup Configuration (p. 27) in this document for more information.

# Step 2: AWS Overlay IP

Decide on IP address to use for your Overlay IP. An Overlay IP address is an AWS-specific routing entry which can send network traffic to an instance, no matter which AZ the instance is located in.

One key requirement for the Overlay IP is that it should not be used elsewhere in your VPC or on-premises. It should be part of the private IP address range defined in RFC1918. For example, if your VPC is configured in the range of `0.0.0.0/8` or `172.16.0.0/12`, you can use the Overlay IP from the range of `192.168.0.0/16`. Based on the number of HA setups you plan to have in your landscape, you can reserve the IP address by reserving a block from the private IP address to ensure there is no overlap.

AWS worked on creating a resource agent, `aws-vpc-move-ip`, which is available along with the Linux Pacemaker. This agent updates the route table of the VPC where you have configured the cluster to always point to the primary DB.

All traffic within the VPC can reach the Overlay IP address via the route table. Traffic from outside the VPC, whether that is from another VPC or on-premises will require AWS Transit Gateway (TGW) or AWS NLB to reach the Overlay IP address. For more information on how to direct traffic to an Overlay IP address via AWS TGW or AWS NLB, see SAP on AWS High Availability with Overlay IP Address Routing.

# Step 3: AWS Resources

Deciding the right storage layout is important to ensure you can meet required IO. EBS gp2 volumes balance price and performance for a wide variety of workloads, while io1 volumes provide the highest performance consistently for mission-critical applications. With these two options, you can choose a storage solution that meets your performance and cost requirements. For more information, see Amazon EBS features for more information.

# Step 4: SAP Netweaver and IBM Db2 Deployment

See the SAP Standard Installation guide based on your installation release to get the technical steps and prerequisites for SAP installation.

## Step 4a: Create EC2 Instances for Deploying SAP NetWeaver ASCS

See SAP NetWeaver Environment Setup for Linux on AWS to learn how to set up an EC2 instance for SAP NetWeaver.

## Step 4b: Create EC2 Instances for IBM Db2 Primary and Standby Databases

Deploy two EC2 instances, one in each AZ, for your primary and standby databases.

## Step 4c: Disable Source/destination Check for the EC2 Instance Hosting the IBM Db2 Primary and Standby Databases

You need to disable source/destination check for your EC2 instance hosting primary and standby databases. This is required to route traffic via Overlay IP. See Changing the source or destination checking to learn more about how to disable source/destination check for your EC2 instance. You can use the following command line interface (CLI) to achieve this.

```
    # aws ec2 modify-instance-attribute --profile cluster --instance-id EC2-instance-id
--no-source-dest-check
```

## Step 4d: AWS IAM Role

For the Pacemaker setup, create two policies and attach them to the IAM role, which is attached to the Db2 primary and standby instance. This allows your EC2 instance to call the APIs which run during the failover process by Pacemaker.

- STONITH – Allows Ec2 instance to start, stop and reboot instances.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "Stmt1424870324000",
           "Effect": "Allow",
           "Action": [
               "ec2:DescribeInstances",
               "ec2:DescribeInstanceAttribute",
               "ec2:DescribeTags"
           ],
           "Resource": "*"
       },
       {
           "Sid": "Stmt1424870324001",
           "Effect": "Allow",
           "Action": [
               "ec2:ModifyInstanceAttribute",
               "ec2:RebootInstances",
               "ec2:StartInstances",
               "ec2:StopInstances"
           ],
           "Resource": [
```

```
                "arn:aws:ec2:region-name:account-id:instance/i-node1",
                "arn:aws:ec2:region-name:account-id:instance/i-node2"
            ]
        }
    ]
}
```

- Overlay IP – Allows the Ec2 instance to update the route table in case of failover.

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:ReplaceRoute",
            "Resource": "arn:aws:ec2:region-name:account-id:route-table/rtb-XYZ"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "ec2:DescribeRouteTables",
            "Resource": "*"
        }
    ]
}
```

Replace the following variables with the appropriate names:

- `region-name`: the name of the AWS region.
- `account-id`: The name of the AWS account in which the policy is used.
- `rtb-XYZ`: The identifier of the routing table which needs to be updated.
- `i-node1`: Instance ID for the Db2 primary instance.
- `i-node2`: Instance id for the Db2 standby instance.

## Step 4e: SAP Application and Db2 Software Install

**Prerequisites**:

- Before starting the installation, update the `/etc/hosts` files of database, ASCS, and application servers with the hostname and IP address of *your* database, ASCS and application servers. This ensures that all your instances can resolve each other's address during installation and configuration.
- You need to install the following packages in your instances: `tcsh.x86_64`, `ksh.x86_64`, `libaio.x86_64`, `libstdc++.x86_64`.
- Comment out the 5912 port entry in the `/etc/services` file (if it exists), as this port is used for the Db2 installation:

```
#fis            5912/tcp          # Flight Information Services
#fis            5912/udp          # Flight Information Services
#fis            5912/sctp         # Flight Information Services
```

**SAP application and Db2 software installation (high-level instructions)**:

1. Install SAP ASCS using software provisioning manager (SWPM) on the Amazon EC2 instance. Choose the installation option depending on the scenario; for example, distributed or HA in case you plan to install ERS for app layer high availability.

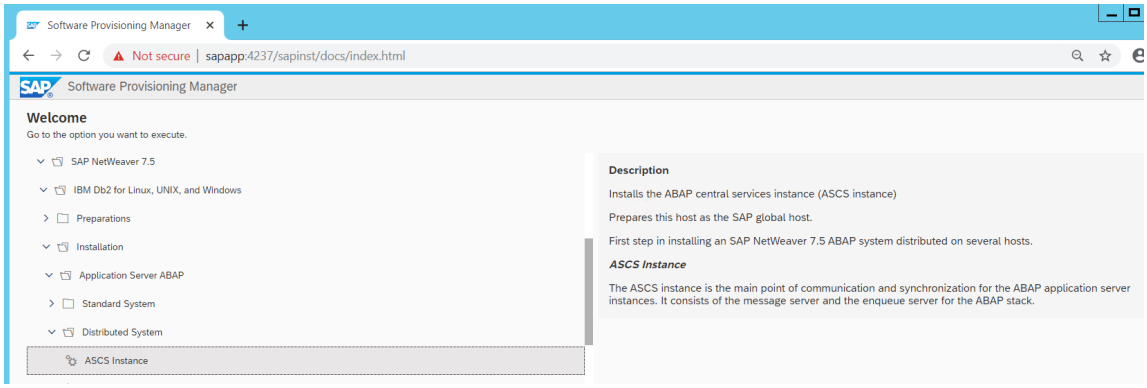


*Figure 3 – Install ASCS*

2. Install the primary database using SWPM on the Amazon EC2 instance hosted in AZ1.

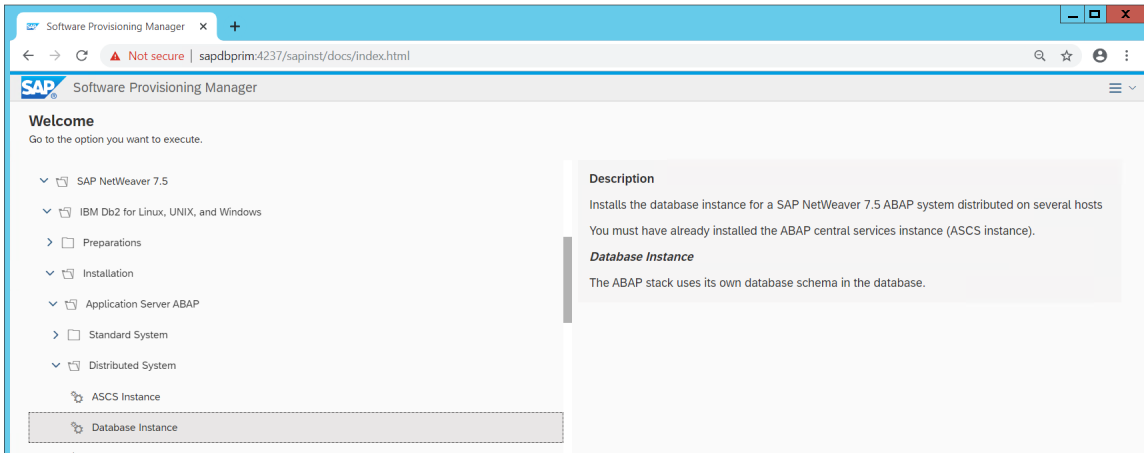


*Figure 4 – Install the primary database*

3. Take a backup of the primary database.
4. Install the PAS instance. This can be the same EC2 instance used in step 1 (p. 8) if you want to install ASCS and PAS on one host.

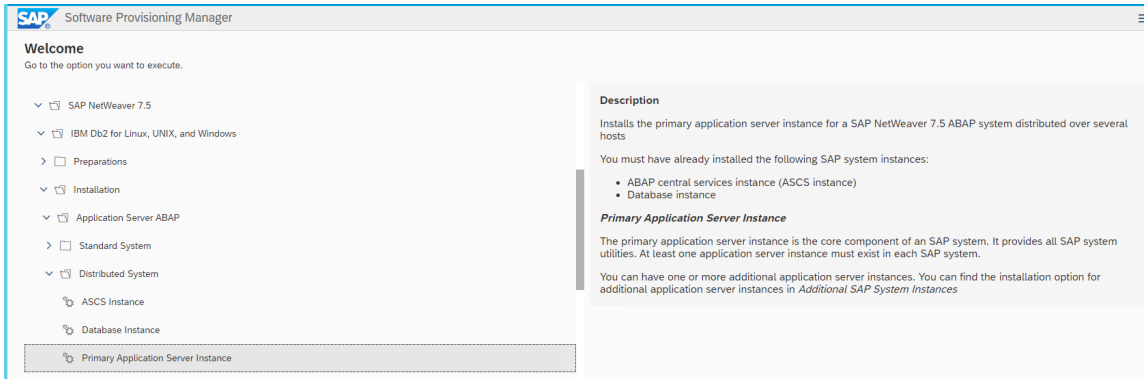


*Figure 5 – Install the PAS instance*

5. For the standby DB installation:

a. Use the SAP homogeneous system copy procedure from SWPM with the option of **System copy** > **Target systems** > **Distributed** > **Database instance**.
b. In the SWPM parameter screen, when asked for system load type, choose Homogenous System and the backup/restore option.
c. When prompted by the SWPM, restore the backup you took during step 3 (p. 8) on the standby DB. You can exit the installer, because the subsequent installation is already completed on the primary database server.

You should now have your ASCS, primary Db2 database, and PAS (if running on a different host than ASCS) installed and running in AZ1 of your setup. In AZ2, you should have the standby Db2 database installed and running. You can optionally install an additional application server if required to support the workload. We recommend that you distribute your application servers in both AZs.

# Step 5: Db2 HADR Setup

The following steps explain how to set up HADR between the primary and standby databases. For additional references, see:

- IBM Db2 HADR documentation
- IBM Support Page

Table 3 details the steps for setup. Update the configuration commands according to your environment.

*Table 3 – Db2 HADR setup*

| **System ID (SID)** | **STJ** |
|---|---|
| **Primary Db2 database hostname** | dbprim00 |
| **Standby Db2 database hostname** | dbsec00 |
| **Overlay IP** | 192.168.1.81 |

**To set up Db2 HADR**:

1. Find the state of the primary database before HADR configuration by executing the following command:

```
> db2 get db cfg for STJ | grep HADR
HADR database role                                      = STANDARD
HADR local host name                  (HADR_LOCAL_HOST) =
HADR local service name                (HADR_LOCAL_SVC) =
HADR remote host name                (HADR_REMOTE_HOST) =
HADR remote service name              (HADR_REMOTE_SVC) =
HADR instance name of remote server  (HADR_REMOTE_INST) =
HADR timeout value                       (HADR_TIMEOUT) = 120
HADR target list                     (HADR_TARGET_LIST) =
HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC
HADR spool log data limit (4KB)      (HADR_SPOOL_LIMIT) = AUTOMATIC(0)
HADR log replay delay (seconds)     (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 0
```

```
 HADR SSL certificate label             (HADR_SSL_LABEL) =
```

2. The `HADR_LOCAL_SVC and HADR_REMOTE_SVC` parameters require an entry in your `/etc/services` file. If the entry is unavailable, update the `/etc/services` file to include the entry. Here is a sample `/etc/services` file entry. The SID is STJ.

```
#  grep -i hadr /etc/services
STJ_HADR_1   5950/tcp   # DB2 HADR log shipping
STJ_HADR_2   5951/tcp   # DB2 HADR log shipping
```

3. Complete the following steps in your primary database (in this case, dbprim00) as the Db2 instance owner (in this case, `db2stj`):

```
db2 update db cfg for STJ using HADR_LOCAL_HOST dbprim00
db2 update db cfg for STJ using HADR_LOCAL_SVC STJ_HADR_1
db2 update db cfg for STJ  using HADR_REMOTE_HOST dbsec00
db2 update db cfg for STJ  using HADR_REMOTE_SVC STJ_HADR_2
db2 update db cfg for STJ  using HADR_REMOTE_INST db2stj
db2 update db cfg for STJ  using HADR_TIMEOUT 120
db2 update db cfg for STJ  using HADR_SYNCMODE NEARSYNC
db2 update db cfg for STJ  using HADR_PEER_WINDOW 300
db2 update db cfg for STJ  using LOGINDEXBUILD ON
```

Here is the state after the configuration was updated:

```
 > db2 get db cfg for STJ | grep HADR
 HADR database role                                      = STANDARD
 HADR local host name                 (HADR_LOCAL_HOST) = dbprim00
 HADR local service name               (HADR_LOCAL_SVC) = STJ_HADR_1
 HADR remote host name                (HADR_REMOTE_HOST) = dbsec00
 HADR remote service name              (HADR_REMOTE_SVC) = STJ_HADR_2
 HADR instance name of remote server  (HADR_REMOTE_INST) = db2stj
 HADR timeout value                       (HADR_TIMEOUT) = 120
 HADR target list                     (HADR_TARGET_LIST) =
 HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC
 HADR spool log data limit (4KB)      (HADR_SPOOL_LIMIT) = AUTOMATIC(0)
 HADR log replay delay (seconds)     (HADR_REPLAY_DELAY) = 0
 HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 300
 HADR SSL certificate label             (HADR_SSL_LABEL) =
```

4. Run the following steps in your standby database (in this case `dbsec00`) as the Db2 instance owner (in this case, `db2stj`).

```
db2 update db cfg for STJ using HADR_LOCAL_HOST dbsec00
db2 update db cfg for STJ using HADR_LOCAL_SVC STJ_HADR_2
db2 update db cfg for STJ  using HADR_REMOTE_HOST dbprim00
db2 update db cfg for STJ  using HADR_REMOTE_SVC STJ_HADR_1
db2 update db cfg for STJ  using HADR_REMOTE_INST db2stj
db2 update db cfg for STJ  using HADR_TIMEOUT 120
db2 update db cfg for STJ  using HADR_SYNCMODE NEARSYNC
db2 update db cfg for STJ  using HADR_PEER_WINDOW 300
db2 update db cfg for STJ  using LOGINDEXBUILD ON
```

Here's an example configuration:

```
> db2 get db cfg for STJ | grep HADR
HADR database role                                      = STANDBY
HADR local host name                  (HADR_LOCAL_HOST) = dbsec00
HADR local service name                (HADR_LOCAL_SVC) = STJ_HADR_2
HADR remote host name                (HADR_REMOTE_HOST) = dbprim00
HADR remote service name              (HADR_REMOTE_SVC) = STJ_HADR_1
HADR instance name of remote server  (HADR_REMOTE_INST) = db2stj
HADR timeout value                       (HADR_TIMEOUT) = 120
HADR target list                     (HADR_TARGET_LIST) =
HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC
HADR spool log data limit (4KB)      (HADR_SPOOL_LIMIT) = AUTOMATIC(1200000)
HADR log replay delay (seconds)     (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 300
HADR SSL certificate label             (HADR_SSL_LABEL) =
```

5. When using Linux pacemaker, use the following Db2 HADR parameters:
   - `HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 300`
   - `HADR timeout value (HADR_TIMEOUT) = 60`

   We recommend that you tune these parameters after testing the failover and takeover functionality. Because individual configuration can vary, the parameter might need adjustment.
6. After your primary and standby databases have been configured, start HADR on the standby server as the HADR standby.

```
db2 start hadr on database STJ as standby
```

7. Start HADR on the primary database.

```
db2 start hadr on database STJ as primary
DB20000I  The START HADR ON DATABASE command completed successfully.

db2pd -hadr -db STJ | head -20

Database Member 0 -- Database STJ -- Active --

                        HADR_ROLE = PRIMARY
REPLAY_TYPE = PHYSICAL
                    HADR_SYNCMODE = NEARSYNC
                       STANDBY_ID = 1
                    LOG_STREAM_ID = 0
                       HADR_STATE = PEER
                       HADR_FLAGS = TCP_PROTOCOL
               PRIMARY_MEMBER_HOST = dbprim00
                  PRIMARY_INSTANCE = db2stj
                                ...
HADR_CONNECT_STATUS = CONNECTED
```

# Step 6: Pacemaker Cluster Setup

In this section we'll discuss the cluster setup using Linux Pacemaker on both RHEL and SLES OS. The Linux Pacemaker works as a failover Orchestrator. It monitors both the primary and standby databases, and in the event of primary database server failure it initiates an automatic HADR takeover by the standby server. The resource agents this configuration uses are as following:

- STONITH resource agent for fencing.
- The db2 database resource, which is configured in a Primary/Standby configuration.
- The `AWS-vpc-move-ip` resource, which is built by the AWS team to handle the overlay IP switch from the Db2 primary instance to standby in the event of failure.

As mentioned in the Operating System (p. 5) section of this document, you need the correct subscription to download these resource agents.

**Important**: Change the shell environment for the `db2<sid>` user to `/bin/ksh`.

**To change the shell environment**:

1. Shut down both the database servers using `db2stop` while logged in as `db2<sid>`.
2. Install Kornshell (ksh) (if it's not already installed).
3. Run `sudo usermod -s /bin/ksh db2<sid>`.

# Step 6a. Setup on RHEL

This section focuses on setting up the Pacemaker cluster on the RHEL operating system.

**To set up the pacemaker cluster on RHEL**:

1. Basic cluster configuration – Install the required cluster packages using both database nodes.

```
yum install -y pcs pacemaker fence-agents-aws
yum install -y resource-agents
```

2. Start the cluster services.

```
systemctl start pcsd.service
systemctl enable pcsd.service
```

   **Note:** If you have subscribed to RHEL for SAP with HA and US products from AWS Marketplace, run `mkdir -p /var/log/pcsd /var/log/cluster` before starting `pcsd.service`.

3. Reset the password for user `hacluster` on both the DB nodes.

```
passwd hacluster
```

4. Authorize the cluster. Make sure that both nodes are able to communicate with each other using the hostname.

```
[root@dbprim00 ~]# pcs cluster auth dbprim00 dbsec00
Username: hacluster
Password:
dbprim00: Authorized
dbsec00: Authorized
[root@dbprim00 ~]#
```

5. Create the cluster.

```
[root@dbprim00 ~]# pcs cluster setup --name db2ha dbprim00 dbsec00
Destroying cluster on nodes: dbprim00, dbsec00...
dbsec00: Stopping Cluster (pacemaker)...
dbprim00: Stopping Cluster (pacemaker)...
dbprim00: Successfully destroyed cluster
dbsec00: Successfully destroyed cluster

Sending 'pacemaker_remote authkey' to 'dbprim00', 'dbsec00'
dbprim00: successful distribution of the file 'pacemaker_remote authkey'
dbsec00: successful distribution of the file 'pacemaker_remote authkey'
Sending cluster config files to the nodes...
dbprim00: Succeeded
dbsec00: Succeeded

Synchronizing pcsd certificates on nodes dbprim00, dbsec00...
dbprim00: Success
dbsec00: Success
Restarting pcsd on the nodes in order to reload the certificates...
dbprim00: Success
dbsec00: Success
[root@dbprim00 ~]# pcs cluster enable --all
dbprim00: Cluster Enabled
dbsec00: Cluster Enabled
[root@dbprim00 ~]# pcs cluster start --all
dbsec00: Starting Cluster...
dbprim00: Starting Cluster...
[root@dbprim00 ~]#
```

**Note**: Adjust the `corosync` timeout.

6. Go to `/etc/corosync/corosync.conf` and add or modify the token value of totem to `30000`.

```
[root@dbprim00 corosync]# more /etc/corosync/corosync.conf
totem {
     version: 2
     cluster_name: db2ha
     secauth: off
     transport: udpu
     token: 30000
}

nodelist {
     node {
          ring0_addr: dbprim00
          nodeid: 1
     }

     node {
          ring0_addr: dbsec00
```

```
            nodeid: 2
        }
    }

    quorum {
        provider: corosync_votequorum
        two_node: 1
    }

    logging {
        to_logfile: yes
        logfile: /var/log/cluster/corosync.log
        to_syslog: yes
    }
```

7. Run `pcs cluster sync` to sync the changes on the standby database node.

```
[root@dbprim00 corosync]# pcs cluster sync
dbprim00: Succeeded
dbsec00: Succeeded
```

8. Run `pcs cluster reload corosync` to make the changes effective.

```
[root@dbprim00 corosync]# pcs cluster reload corosync
Corosync reloaded
```

9. To ensure of the changes are in place, run `corosync-cmapctl | grep totem.token`.

```
[root@dbprim00 corosync]# corosync-cmapctl | grep totem.token
runtime.config.totem.token (u32) = 30000
runtime.config.totem.token_retransmit (u32) = 7142
runtime.config.totem.token_retransmits_before_loss_const (u32) = 4
totem.token (u32) = 30000
```

10. Before creating any resource, put the cluster in maintenance mode.

```
[root@dbprim00 ~]# pcs property set maintenance-mode='true'
```

11. Create the STONITH resource. You will need the EC2 instance IDs for this operation. The default `pcmk` action is reboot. Replace the instance ID for `dbprim00` and `dbsec00` with the instance IDs of your setup.

    If you want to have the instance remain in a stopped state until it has been investigated and then manually started, add `pcmk_reboot_action=off`. This setting is also required if you are running the Db2 on Amazon EC2 Dedicated Hosts.

```
[root@dbprim00 ~]# pcs stonith create clusterfence fence_aws
region=us-east-1 pcmk_host_map="dbprim00:i-09d1b1f105f71e5ed;dbsec00:i-
0c0d3444601b1d8c5" power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
op start timeout=300
op monitor timeout=60
```

12. Create the Db2 resource.

```
[root@dbprim00 ~]# pcs resource create Db2_HADR_STJ db2
instance=db2stj dblist=STJ master meta notify=true resource-
stickiness=5000 op demote timeout=240 op promote timeout=240 op
start timeout=240 op stop timeout=240 op monitor interval=20s
timeout=120s op monitor interval=22s role=Master timeout=120s
```

**Note**: The timeout values here are default, which works for most deployments. We recommend that you test the timeouts in the QA setup extensively based on the test cases mentioned in the Appendix, and then tune it accordingly.

13. Create the Overlay IP resource agent. First, add the Overlay IP in the primary node.

```
[root@dbprim00 ~]# ip address add 192.168.1.81/32 dev eth0
[root@dbprim00 ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default
qlen 1000
    link/ether 0e:10:e3:7b:6f:4f brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.116/24 brd 10.0.1.255 scope global noprefixroute dynamic eth0
        valid_lft 2885sec preferred_lft 2885sec
    inet 192.168.1.81/32 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::c10:e3ff:fe7b:6f4f/64 scope link
        valid_lft forever preferred_lft forever
[root@dbprim00 ~]#
```

14. Update the route table with the Overlay IP pointing to the Db2 primary instance:

```
aws ec2 create-route --route-table-id rtb-xxxxxxxx --destination-
cidr-block Overlay IP --instance-id i-xxxxxxxx
[root@dbprim00 ~]# aws ec2 create-route --route-table-id rtb-
dbe0eba1 --destination-cidr-block 192.168.1.81/32 --instance-id i-09d1b1f105f71e5ed
{
    "Return": true
}

[root@dbprim00 ~]# pcs resource create db2-oip aws-vpc-move-ip
ip=192.168.1.81 interface=eth0 routing_table=rtb-dbe0eba1
```

**Note**: If you are using a different route table for both the subnets where you are deploying the primary and standby databases, you can specify them using a comma (,) in the resource creation command:

```
pcs resource create db2-oip aws-vpc-move-ip ip=192.168.1.81 interface=eth0
routing_table=rtb-xxxxx1,rtb-xxxxx2
```

15. Create a colocation constraint to bind the Overlay IP resource agent with the Db2 primary instance.

```
    [root@dbprim00 ~]# pcs constraint colocation add db2-oip with master Db2_HADR_STJ-
master 2000
```

16. You can now remove the maintenance mode by using the following code:

```
     [root@dbprim00 ~]# pcs property set maintenance-mode='false'
```

This is the final configuration of the cluster:

```
     [root@dbprim00 ~]# pcs config show
     Cluster Name: db2ha
     Corosync Nodes:
       dbprim00 dbsec00
     Pacemaker Nodes:
       dbprim00 dbsec00

     Resources:
       Master: Db2_HADR_STJ-master
         Resource: Db2_HADR_STJ (class=ocf provider=heartbeat type=db2)
           Attributes: dblist=STJ instance=db2stj
           Meta Attrs: notify=true resource-stickiness=5000
           Operations: demote interval=0s timeout=120 (Db2_HADR_STJ-demote-interval-0s)
     monitor interval=20 timeout=60 (Db2_HADR_STJ-monitor-interval-20)
     monitor interval=22 role=Master timeout=60 (Db2_HADR_STJ-monitor-interval-22)
     notify interval=0s timeout=10 (Db2_HADR_STJ-notify-interval-0s)
     promote interval=0s timeout=120 (Db2_HADR_STJ-promote-interval-0s)
     start interval=0s timeout=120 (Db2_HADR_STJ-start-interval-0s)
     stop interval=0s timeout=120 (Db2_HADR_STJ-stop-interval-0s)
     Resource: db2-oip (class=ocf provider=heartbeat type=aws-vpc-move-ip)
     Attributes: interface=eth0 ip=192.168.1.81 routing_table=rtb-dbe0eba1
     Operations: monitor interval=60 timeout=30 (db2-oip-monitor-interval-60)
     start interval=0s timeout=180 (db2-oip-start-interval-0s)
     stop interval=0s timeout=180 (db2-oip-stop-interval-0s)

     Stonith Devices:
       Resource: clusterfence (class=stonith type=fence_aws)
         Attributes:
 pcmk_host_map=dbprim00:i-09d1b1f105f71e5ed;dbsec00:i-0c0d3444601b1d8c5
 pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=us-east-1
         Operations: monitor interval=60s (clusterfence-monitor-interval-60s)
     Fencing Levels:

     Location Constraints:
     Ordering Constraints:
     Colocation Constraints:
         db2-oip with Db2_HADR_STJ-master (score:2000) (rsc-role:Started) (with-rsc-
role:Master)
       Ticket Constraints:

     Alerts:
       No alerts defined

     Resources Defaults:
       No defaults set
     Operations Defaults:
       No defaults set
```

```
 Cluster Properties:
   cluster-infrastructure: corosync
   cluster-name: db2ha
   dc-version: 1.1.18-11.el7_5.4-2b07d5c5a9
   have-watchdog: false

   Quorum:
   Options:
[root@dbprim00 ~]#
```

# Step 6b. Setup on SLES

This section focuses on setting up the Pacemaker cluster on the SLES operating system.

**Prerequisite**: You need to complete this on both the Db2 primary and standby instances.

**To create an AWS CLI profile**:

The SLES operating system's resource agents use the AWS Command Line Interface (CLI). You need to create the AWS CLI profile for the root account on both instances: one with the default profile and the other with an arbitrary profile name (in this example, `cluster`) which creates output in text format. The region of the instance must be added as well.

1. Replace the string region-name with your target region in the following example.

```
dbprim00:~ # aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-1
Default output format [None]:
dbprim00:~ # aws configure --profile cluster
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-1
Default output format [None]: text
```

   You don't need to provide the Access Key and Secret Access key, because access is controlled by the IAM role you created earlier in the setup.
2. Add a second private IP address.
3. You are required to add a second private IP address for each cluster instance. Adding a second IP address to the instance allows the SUSE cluster to implement a two-ring corosync configuration. The two-ring corosync configuration allows the cluster nodes to communicate with each other using the secondary IP address if there is an issue communicating with each other over the primary IP address.

   See *To assign a secondary private IPv4 address to a network interface*.
4. Add a tag with an arbitrary "key" (in this case, `pacemaker`). The value of this tag is the hostname of the respective Db2 instance. This is required to enable AWS CLI to use filters in the API calls.
5. Disable the source/destination check.
6. Ensure that the source/destination check is disabled, as described in Step 4c (p. 9).
7. Avoid deletion of cluster-managed IP addresses on the `eth0` interface.
8. Check if the package `cloud-netconfig-ec2` is installed with the following command:

```
dbprim00:~ # zypper info cloud-netconfig-ec2
```

9. Update the file `/etc/sysconfig/network/ifcfg-eth0` if this package is installed. Change the following line to a 'no' setting or add the following line if the package is not yet installed:

```
dbprim00:~ # CLOUD_NETCONFIG_MANAGE='no'
```

10. Set up NTP (best with YaST). Use AWS time service at `169.254.169.123`, which is accessible from all EC2 instances. Enable ongoing synchronization.
11. Activate the public cloud module to get updates for the AWS CLI:

```
dbprim00:~ # SUSEConnect --list-extensions
dbprim00:~ # SUSEConnect -p sle-module-public-cloud/12/x86_64
Registering system to registration proxy https://smt-ec2.susecloud.net
Updating system details on https://smt-ec2.susecloud.net ...
 Activating sle-module-public-cloud 12 x86_64 ...
-> Adding service to system ...
-> Installing release package ...
Successfully registered system
```

12. Update your packages on both the with the command:

```
 dbprim00:~ # zypper -n update
```

13. Install the resource agent pattern `ha_sles`.

```
 dbprim00:~ # zypper install -t pattern ha_sles
```

**To configure pacemaker: Configuration of the corosync.conf file**:

1. Use the following configuration in the `/etc/corosync/corosync.conf` file on both the Db2 primary and standby instances:

```
# Read the corosync.conf.5 manual page
totem {
   version: 2
   rrp_mode: passive
   token: 30000
   consensus: 36000
   token_retransmits_before_loss_const: 10
   max_messages: 20
   crypto_cipher: none
   crypto_hash: none
   clear_node_high_bit: yes
   interface {
      ringnumber: 0
      bindnetaddr: <ip-local-node>
      mcastport: 5405
      ttl: 1
  }
  transport: udpu
}
```

```
      logging {
          fileline: off
          to_logfile: yes
          to_syslog: yes
          logfile: /var/log/cluster/corosync.log
          debug: off
          timestamp: on
          logger_subsys {
             subsys: QUORUM
             debug: off
        }
      }
      nodelist {
        node {
        ring0_addr: <ip-node-1>
        ring1_addr: <ip2-node-1>
        nodeid: 1
        }
        node {
        ring0_addr: <ip-node-2>
        ring1_addr: <ip2-node-2>
        nodeid: 2
        }
     }

     quorum {
        # Enable and configure quorum subsystem (default: off)
        # see also corosync.conf.5 and votequorum.5
        provider: corosync_votequorum
        expected_votes: 2
        two_node: 1
     }
```

2. Replace the variables `ip-node-1 / ip2-node-1` and `ip-node-2 / ip2-node-2` with the IP addresses of your Db2 primary and standby instances, respectively. Replace `ip-local-node` with the IP address of the instance on which the file is being created.

   The chosen settings for `crypto_cipher` and `crypto_hash` are suitable for clusters in AWS. They may be modified according to SUSE's documentation if you want strong encryption of cluster communication.

3. Start the cluster services and enable them on both the Db2 primary and standby instances.

```
          dbprim00:~ # systemctl start pacemaker
dbprim00:~ # systemctl enable pacemaker
Created symlink from /etc/systemd/system/multi-user.target.wants/pacemaker.service to /
usr/lib/systemd/system/pacemaker.service.
```

4. Check the configuration with the following command:

```
          dbprim00:~ # corosync-cfgtool -s
Printing ring status.
Local node ID 1
RING ID 0
        id      = 10.0.1.17
        status  = ring 0 active with no faults
RING ID 1
        id      = 10.0.1.62
        status  = ring 1 active with no faults
dbprim00:~ #
```

```
Cluster status:
dbprim00:~ # crm_mon -1
Stack: corosync
Current DC: dbsec00 (version 1.1.19+20181105.ccd6b5b10-3.13.1-1.1.19+20181105.ccd6b5b10)
 - partition with quorum
Last updated: Fri Apr 17 14:09:56 2020
Last change: Fri Apr 17 13:38:59 2020 by hacluster via crmd on dbsec00

2 nodes configured
0 resources configured

Online: [ dbprim00 dbsec00 ]

No active resources
```

**To prepare the cluster for adding resources**:

1. To avoid cluster starting partially defined resources, set the cluster to maintenance mode. This deactivates all monitor actions.

```
     dbprim00:~ # crm configure property maintenance-mode="true"
dbprim00:~ # crm status
Stack: corosync
Current DC: dbprim00 (version 1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10)
 - partition with quorum
Last updated: Fri Apr 17 14:30:51 2020
Last change: Fri Apr 17 14:30:50 2020 by root via cibadmin on dbprim00

2 nodes configured
0 resources configured

              *** Resource management is DISABLED ***
  The cluster will not attempt to start, stop or recover services

Online: [ dbprim00 dbsec00 ]

No resources
```

2. Configuring AWS-specific settings:

```
dbprim00:/ha-files # vi crm-bs.txt
dbprim00:/ha-files # more crm-bs.txt
property cib-bootstrap-options: \
stonith-enabled="true" \
stonith-action="off" \
stonith-timeout="600s"
rsc_defaults rsc-options: \
resource-stickiness=1 \
migration-threshold=3
op_defaults op-options: \
timeout=600 \
record-pending=true
```

The `off` setting forces the agents to shut down the instance. You have the option of changing it to reboot if required.

3. Add the following configuration to the cluster:

```
dbprim00:~ # crm configure load update crm-bs.txt
```

**To configure the AWS-specific STONITH resource #**:

1. Create a file with the following content:

```
primitive res_AWS_STONITH stonith:external/ec2 \
op start interval=0 timeout=180 \
op stop interval=0 timeout=180 \
op monitor interval=180 timeout=60 \
params tag=pacemaker profile=cluster
```

   The EC2 tag `pacemaker` entry needs to match the tag chosen for the EC2 instances, and the name of the profile needs to match the previously configured AWS profile as part of the prerequisite section.

2. Add the file to the configuration:

```
dbprim00:/ha-files # vi aws-stonith.txt
dbprim00:/ha-files # more aws-stonith.txt
primitive res_AWS_STONITH stonith:external/ec2 \
op start interval=0 timeout=180 \
op stop interval=0 timeout=180 \
op monitor interval=120 timeout=60 \
params tag=pacemaker profile=cluster

dbprim00:/ha-files # crm configure load update aws-stonith.txt
```

3. Create the Db2 Primary/Standby resource.
4. Create a file with the following content. Change the value for SID, as per your configuration.

```
primitive rsc_db2_db2stj_STJ db2 \
   params instance="db2stj" dblist="STJ" \
   op start interval="0" timeout="130" \
   op stop interval="0" timeout="120" \
   op promote interval="0" timeout="120" \
   op demote interval="0" timeout="120" \
   op monitor interval="30" timeout="60" \
   op monitor interval="31" role="Master" timeout="60"
ms msl_db2_db2stj_STJ rsc_db2_db2stj_STJ \
   meta target-role="Started" notify="true"
```

5. Add the file to the configuration:

```
dbprim00:/ha-files # vi db2res.txt
dbprim00:/ha-files # more db2res.txt
primitive rsc_db2_db2stj_STJ db2 \
   params instance="db2stj" dblist="STJ" \
   op start interval="0" timeout="130" \
   op stop interval="0" timeout="120" \
```

```
        op promote interval="0" timeout="120" \
        op demote interval="0" timeout="120" \
        op monitor interval="30" timeout="60" \
        op monitor interval="31" role="Master" timeout="60"
dbprim00:/ha-files # crm configure load update db2res.txt
```

6. Create the Overlay IP resource agent.

a. First, add the Overlay IP in the Db2 primary instance.

```
      dbprim00:~# ip address add 192.168.1.81/32 dev eth0
dbprim00:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen
 1000
    link/ether 0e:73:7f:b5:b2:95 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.17/24 brd 10.0.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 10.0.1.62/32 scope global eth0
       valid_lft forever preferred_lft forever
    inet 192.168.1.81/32 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::c73:7fff:feb5:b295/64 scope link
       valid_lft forever preferred_lft forever

[root@dbprim00 ~]#
```

b. Update the route table with the Overlay IP pointing to the Db2 primary instance.

```
    aws ec2 create-route --route-table-id rtb-xxxxxxxx --destination-cidr-block Overlay
 IP --instance-id i-xxxxxxxx
```

**Note**: If you are using different route table for both the subnets where you are deploying the primary and standby database, you can specify them with a comma (,) in the command preceding this note.

```
    dbprim00:~ # aws ec2 create-route --route-table-id rtb-dbe0eba1 --
    destination-cidr-block 192.168.1.81/32 --instance-id i-05fc8801284585362
    {
        "Return": true
    }
```

The `aws-vpc-move-ip` resource agent call the AWS command from the location `/usr/bin`, so ensure that there is a soft link pointing to the location where you have the `awscli` installed.

```
    dbprim00:/usr/bin # which aws
    /usr/local/bin/aws
    dbprim00:/usr/bin # ls -ltr aws
    lrwxrwxrwx 1 root root 18 Apr 18 17:44 aws -> /usr/local/bin/aws
```

c. Create the file with the following content, and replace the Overlay IP and the route table ID based on your configuration. If you have multiple route tables associated with the subnet to which your instances belong, you can use a comma-separated list of routing tables.

**Note**: Make sure you use the same profile name (which is cluster for this setup) that you used while configuring the AWS CLI.

```
dbprim00:/ha-files # vi aws-move-ip.txt
dbprim00:/ha-files # more aws-move-ip.txt
primitive res_AWS_IP ocf:suse:aws-vpc-move-ip \
     params ip=192.168.1.81 routing_table=rtb-dbe0eba1 interface=eth0
profile=cluster \
     op start interval=0 timeout=180 \
     op stop interval=0 timeout=180 \
     op monitor interval=60 timeout=60
dbprim00:/ha-files # crm configure load update aws-move-ip.txt
```

d. Create a colocation constraint to bind the Overlay IP resource agent with the Db2 primary instance.

```
dbprim00:/ha-files # more crm-cs.txt
colocation col_db2_db2stj_STJ 2000: res_AWS_IP:Started \
msl_db2_db2stj_STJ:Master
dbprim00:/ha-files # crm configure load update crm-cs.txt
dbprim00:/ha-files #
```

e. Adjust the resource-stickiness and migration-threshold values.

```
dbprim00:~ # crm configure rsc_defaults resource-stickiness=1000
dbprim00:~ # crm configure rsc_defaults migration-threshold=5000
```

f. You can now remove maintenance-mode.

```
dbprim00:~ # crm configure property maintenance-mode="false"
```

Final configuration of the cluster:

```
dbprim00:/ha-files # crm status
Stack: corosync
Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
Last updated: Sat Apr 18 18:45:53 2020
Last change: Sat Apr 18 16:01:26 2020 by root via cibadmin on dbprim00

2 nodes configured
4 resources configured

Online: [ dbprim00 dbsec00 ]

Full list of resources:
```

```
res_AWS_STONITH        (stonith:external/ec2): Started dbprim00
Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
Masters: [ dbprim00 ]
Slaves: [ dbsec00 ]
res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbprim00

dbprim00:/ha-files # crm configure show
node 1: dbprim00
node 2: dbsec00
primitive res_AWS_IP ocf:suse:aws-vpc-move-ip \
params ip=192.168.1.81 routing_table=rtb-dbe0eba1 interface=eth0 profile=cluster \
op start interval=0 timeout=180 \
op stop interval=0 timeout=180 \
op monitor interval=60 timeout=60
primitive res_AWS_STONITH stonith:external/ec2 \
op start interval=0 timeout=180 \
op stop interval=0 timeout=180 \
op monitor interval=120 timeout=60 \
params tag=pacemaker profile=cluster
primitive rsc_db2_db2stj_STJ db2 \
params instance=db2stj dblist=STJ \
op start interval=0 timeout=130 \
op stop interval=0 timeout=120 \
op promote interval=0 timeout=120 \
op demote interval=0 timeout=120 \
op monitor interval=30 timeout=60 \
op monitor interval=31 role=Master timeout=60
ms msl_db2_db2stj_STJ rsc_db2_db2stj_STJ \
meta target-role=Started notify=true
colocation col_db2_db2stj_STJ 2000: res_AWS_IP:Started msl_db2_db2stj_STJ:Master
property cib-bootstrap-options: \
have-watchdog=false \
dc-version="1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10" \
cluster-infrastructure=corosync \
maintenance-mode=false \
stonith-enabled=true \
stonith-action=off \
stonith-timeout=600s
rsc_defaults rsc-options: \
resource-stickiness=1000 \
migration-threshold=5000
op_defaults op-options: \
timeout=600 \
record-pending=true
```

# Step 7: Post Setup Configuration

To enable SAP to connect to the Db2 virtual name, post-setup configuration tasks must be performed.

To perform post-setup configuration tasks:

1. Edit your SAP profile files:

```
> vi DEFAULT.PFL

SAPDBHOST = dbhadb2
j2ee/dbhost = dbhadb2

rsdb/reco_trials  = 10
rsdb/reco_sleep_time  = 10
```

2. Update the two parameters (`SAPDBHOST` and `j2ee/dbhost`) to the virtual name you chose for your database server. You will have to update the `rsdb/reco*` parameters to greater than failover duration to avoid DB disconnect in case of failover. We recommend that you test these values in QA before setting it up in production.
3. Edit your Db2 client file:

```
> cd /sapmnt/STJ/global/db6
sappas01:stjadm 15> more db2cli.ini
; Comment lines start with a semi-colon.
[STJ]
Database=STJ
Protocol=tcpip
Hostname=dbhadb2
Servicename=5912
[COMMON]
Diagpath=/usr/sap/STJ/SYS/global/db6/db2dump
```

   Make sure the hostname parameter matches your Db2 virtual hostname.
4. After you change the entries and save your file, test your connection to the database server:

```
sappas01:stjadm 17> R3trans -d
This is R3trans version 6.26 (release 745 - 13.04.18 - 20:18:04).
unicode enabled version
R3trans finished (0000).
sappas01:stjadm 18> startsap
Checking db Database
Database is running
-----------------------------------------
Starting Startup Agent sapstartsrv
OK
Instance Service on host sappas01 started
-----------------------------------------
starting SAP Instance D00
Startup-Log is written to /home/stjadm/startsap_D00.log
-----------------------------------------
/usr/sap/STJ/D00/exe/sapcontrol -prot NI_HTTP -nr 00 -function Start
Instance on host sappas01 started
```

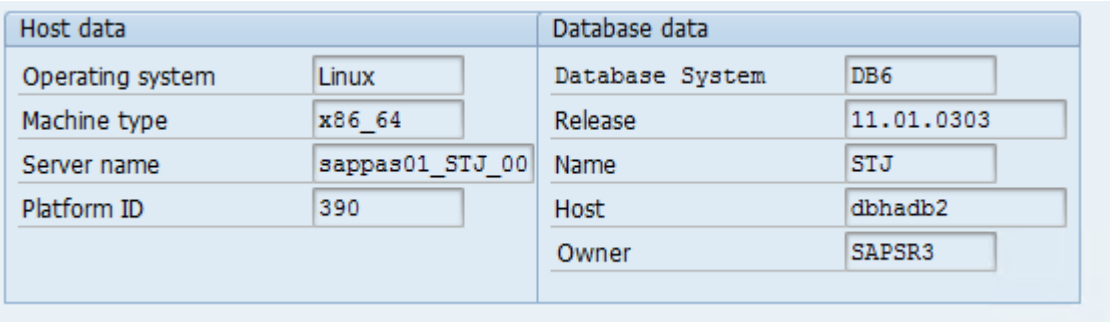


*Figure 6 – SAP system status information*

You can check get the status/information of HADR in the transaction **DB02/dbacockpit** > **Configuration** > **Overview**.

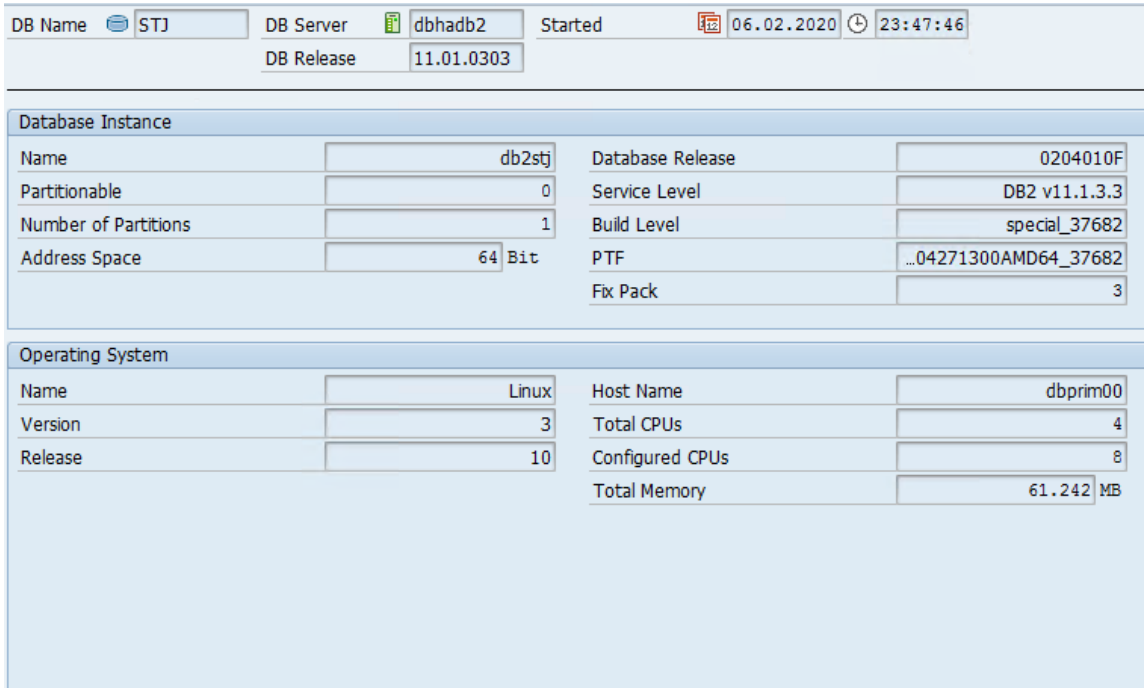


*Figure 7 – Transaction DB02 database instance information*

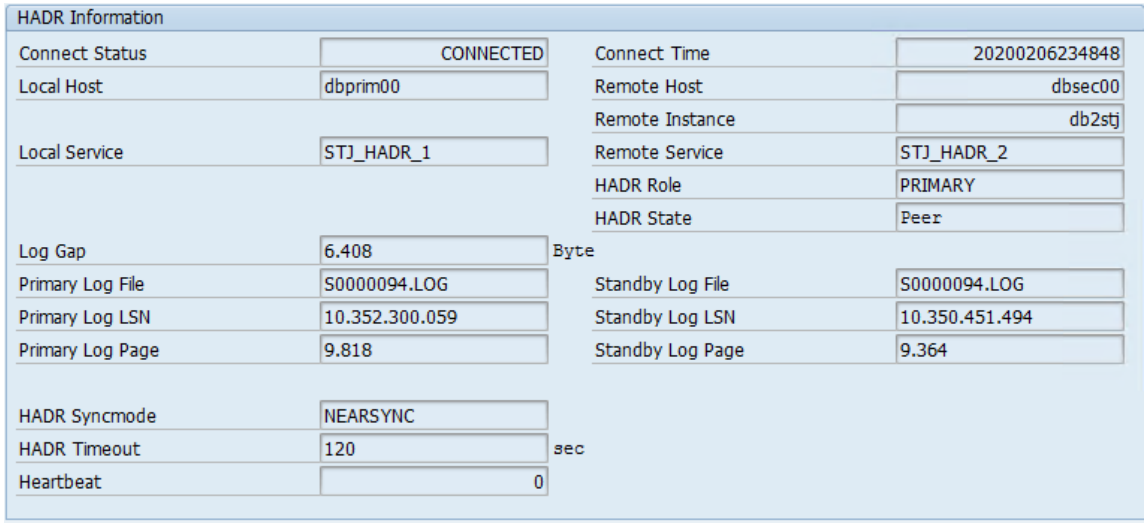


*Figure 8 – Transaction DB02 HADR information*

# Step 8: Testing and Validation

We recommend you define your failure scenarios and test them on your cluster. Unless otherwise specified, all tests are done with the primary node running on the primary server (`dbprim00`) and the standby node running on the standby server (`dbsec00`).

**Prerequisite**: Before running any tests, please ensure that:

- There is no error or failed action in the Pacemaker. This can be tested using pcs status. In case there is any failed action, check the cause in `/var/log/cluster/corosync.log` in the node on which it

has failed, and then take the corrective action. You can clean the failed action using `pcs/crm` resource cleanup.

- There is no unintended location constraint set up. Using the `pcs/crm` resource, move the master from primary to standby to set a location constraint on the primary node which prevents any resource from starting on it. This can be identified using the `pcs/crm` constraint show. Note the ID of the location constraint, and then run `pcs/crm` constraint delete `<id>` to remove it.
- The Db2 HADR synchronization is working. This can be checked using `db2pd -hadr -db <DBSID>` and comparing the `LOG_FILE`, `PAGE`, and `POS` for primary and standby.
- Refer to Appendix 1 (p. 33) for detailed test cases on RHEL setup.
- Refer to Appendix 2 (p. 39) for detailed test cases on SLES Setup

# Operations

In this section we will cover some of the native AWS services that help you with day-to-day operations of your IBM Db2 database for SAP applications.

## Monitoring

AWS provides multiple native services to monitor and manage your infrastructure and applications on AWS. Services like Amazon CloudWatch and AWS CloudTrail can be leveraged to monitor your underlying infrastructure and APIs, respectively.

CloudWatch provides ready-to-use key performance indicators (KPIs) that you can use to monitor CPU utilization and disk utilization.

You can also create custom metrics for monitoring IBM Db2.

With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS CloudTrail is enabled on all AWS accounts, and records your account activity upon account creation. You can view and download the last 90 days of your account activity for create, modify, and delete operations of supported services without the need to manually set up CloudTrail.

## Backup and Recovery

You need to regularly back up your operating system and database to recover them in case of failure. AWS provides various services and tools that you can use to back up your IBM Db2 database of SAP applications.

### AWS Backup

AWS Backup is a fully managed backup service centralizes and automates the backup of data across AWS services. Using AWS Backup, you can centrally configure backup policies and monitor backup activity for AWS resources, such as EBS volumes, Amazon EC2 instances, and Amazon Elastic File System (Amazon EFS). AWS Backup automates and consolidates backup tasks previously performed service-by-service, removing the need to create custom scripts and manual processes. AWS Backup provides a fully managed, policy-based backup solution, simplifying your backup management and enabling you to meet your business and regulatory backup compliance requirements.

### AMI

You can use the AWS Management Console or the AWS CLI to create a new Amazon Machine Image (Amazon AMI) of your existing SAP system. This can be used to recover your existing SAP system or create a clone.

The AWS CLI create image command creates a new AMI based on an existing Amazon EC2 instance. The new AMI contains a complete copy of the operating system and its configuration, software configurations, and optionally all EBS volumes that are attached to the instance.

A simple command to create an AMI with reboot (if running) of your EC2 instance (with instance ID `i-0b09a25c58929de26` as example) including all attached EBS volumes:

```
aws ec2 create-image --instance-id i-0b09a25c58929de26 --name "My server"
```

A simple command to create AMI without reboot (if running) of your EC2 instance (with instance ID `i-0b09a25c58929de26` as example) including all attached EBS volumes:

```
aws ec2 create-image --instance-id i-0b09a25c58929de26 --name "My server" --no-
reboot
```

## Amazon EBS Snapshots

You can back up your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved.

Snapshots are suited to backup SAP file systems like `/usr/sap/* , /sapmnt/*`. We do not recommend using snapshots to back up your volumes containing data and log files. If you decide to take snapshots for your database volume snapshot, keep in mind that for consistency you should use Microsoft's Volume Shadow Copy Service and use the run command to back up or shut down your database before Snapshots is triggered.

A simple command to create a snapshot of volume (with volume `id vol-1234567890abcdef0` as example):

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is my
 volume snapshot."
```

## Database Backups

One of following methods can be used for IBM Db2 database backup:

- **With native tools to take backup on disk**— Backup requires high throughput compared to Input/Output Operations Per Second (IOPS). We recommend using st1 disk, which provides maximum throughput of 500MB/s per volume. Once the backup completes on disk it can be moved to an Amazon S3 bucket via scripts.
- **With third party backint tools**— There are many third-party tools from partners like Commvault and Veritas that use SAP backint interface and store backups directly in Amazon S3 buckets.

## Storage

The storage services we use across this guide are:

### Amazon EBS

Amazon EBS provides persistent storage for SAP applications and databases. EBS volume size can be increased or their type can be changed (for example, gp2 to io1) without downtime requirements. For more information, see Modifying Amazon EBS volume.

Once you have extended the volume, you need to extend the drive with your Linux volume manager software.

### Amazon S3

Amazon S3 does not need you to explicitly provision storage at all – you simply pay for what you use.

# Operating System Maintenance

Operating system maintenance across large estates of EC2 instances can be managed by:

- Tools specific to each operating system such as SUSE Manager and Red Hat Smart Management.
- 3rd party products such as those available on the AWS Marketplace.
- Using AWS Systems Manager.

Here are some key operating system maintenance tasks that can help with:

## Patching

Follow SAP recommended patching processes to update your landscape on AWS. For operating system patching, with AWS Systems Manager Patch Manager you can roll out OS patches as per your corporate policies. There are multiple key features such as:

- Scheduling based on tags
- Auto-approving patches with lists of approved and rejected patches
- Defining patch baselines

AWS Systems Manager Patch Manager integrates with IAM, AWS CloudTrail, and Amazon CloudWatch Events to provide a secure patching experience that includes event notifications and the ability to audit usage. For details about the process, see How Patch Manager Operations Work. If AWS Systems Manager Patch Manager does not fulfil your requirements, there are third-party products available as well. Some of these are available via the AWS Marketplace.

## Maintenance Window

AWS Systems Manager Maintenance Windows enables you to define a schedule for when to perform potentially disruptive actions on your instances, such as patching an operating system, updating drivers, or installing software or patches.

## Automation Using Documents

AWS Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon EC2 instances and other AWS resources. Automation enables you to do the following:

- Build automation workflows to configure and manage instances and AWS resources.
- Create custom workflows or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
- Monitor automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

## Business Continuity

AWS recommends periodically scheduling business continuity process validations by executing disaster recovery tests. This planned activity helps to flush out any potential unknowns, and helps the organization deal with any real disaster in a streamlined manner. Depending on your disaster recovery architecture it may include:

- Backup/Recovery of databases from S3.
- Creation of systems from AMI and point-in-time recovery via snapshots.
- Changing EC2 instance size of pilot light systems.
- Validation of integration (AD/DNS, email, 3rd party, and more)

## Support

SAP requires customers to have a minimum AWS Business Support plan with AWS. This ensures that any critical issues raised with SAP are also handled by AWS on priority. AWS business support provides a less than one-hour response time for production-down scenarios. You can also choose to have an AWS enterprise support plan, which provides a less than 15-minute response time for business-critical systems, along with other benefits. See AWS Enterprise Support.

For any SAP application issues, AWS suggests raising an incident with SAP via the SAP support portal. After the first level of investigation, SAP can redirect the incident to AWS support if they find an infrastructure related issue which needs to be managed by AWS. However, if you choose to raise support issues for SAP applications with AWS support, we cannot redirect the tickets to SAP. For any infrastructure related issues, you can raise the issue directly with AWS support.

## Cost Optimization

Resources (CPU, memory, additional application servers, system copies for different tests/validations and more) required the SAP landscape change over time. AWS recommends monitoring system utilization, and the need for existing systems, on a regular basis to take actions that will reduce cost. In cases of databases like IBM Db2 as we cannot scale out only opportunity to right size database server is by scaling up/down or shutting it down if not required. A few suggestions to consider:

- Consider reserved instances or savings plans over on-demand instances if your requirement is to run 24-7, 365 days a year. Reserved instances provide up to 75% discount over on-demand instances. See Amazon EC2 pricing.
- Consider running occasionally required systems like training and sandbox on-demand for the duration required.
- Monitor CPU and memory utilization overtime for other non-production systems like Dev/QA, and right-size them when possible.

# Appendix 1: Testing on RHEL Setup

## Test Case 1: Manual Failover

**Procedure**: Use the command `pcs resource move <Db2 master resource name>`.

```
    [root@dbprim00 profile]# pcs resource move Db2_HADR_STJ-master
    Warning: Creating location constraint cli-ban-Db2_HADR_STJ-master-on-dbprim00 with a
score of -INFINITY for resource
```

```
    Db2_HADR_STJ-master on-dbprim00 with a score of -INFINITY for resource Db2_HADR_STJ-
    master on node dbprim00.
    This will prevent Db2_HADR_STJ-master from running on dbprim00
    until the constraint is removed. This will be the case even if
    dbprim00 is the last node in the cluster.
    [root@dbprim00 profile]#
```

**Expected result**: The Db2 primary node is moved from primary node to standby node.

```
    [root@dbprim00 profile]# pcs status
    Cluster name: db2ha
    Stack: corosync
    Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
    Last updated: Sat Feb  8 08:54:04 2020
    Last change: Sat Feb  8 08:53:02 2020 by root via crm_resource on dbprim00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    clusterfence   (stonith:fence_aws):    Started dbprim00
        Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
        Masters: [ dbsec00 ]
    Stopped: [ dbprim00 ]
    db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

    Daemon Status:
    corosync: active/enabled
    pacemaker: active/enabled
    pcsd: active/enabled
    [root@dbprim00 profile]#
```

**Followup actions**: Remove the location constraint.

When using a manual command for moving the resource, there is location constraint created on the node (in this case, the primary node) that prevents running the Db2 resource in standby mode.

**To remove the location constraint:**

1. Use the following command to remove the location constraint:

```
    # pcs config show
    Location Constraints:
    Resource: Db2_HADR_STJ-master
    Disabled on: dbprim00 (score:-INFINITY) (role: Started) (id:cli-ban-Db2_HADR_STJ-
master-on-dbprim00)

    [root@dbprim00 profile]# pcs constraint delete cli-ban-Db2_HADR_STJ-master-on-
dbprim00
```

2. Start the Db2 instance as standby on the new standby node, logged in as `db2<sid>`. Next, clean up the error logged in as root.

```
    db2stj> db2start
```

```
02/08/2020 09:11:29     0   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.

db2stj> db2 start hadr on database STJ as standby
DB20000I  The START HADR ON DATABASE command completed successfully.

[root@dbprim00 ~]# pcs resource cleanup
Cleaned up all resources on all nodes
[root@dbprim00 ~]# pcs status
Cluster name: db2ha
Stack: corosync
Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
Last updated: Sat Feb  8 09:13:17 2020
Last change: Sat Feb  8 09:12:26 2020 by hacluster via crmd on dbprim00

2 nodes configured
4 resources configured

Online: [ dbprim00 dbsec00 ]

Full list of resources:

clusterfence   (stonith:fence_aws):    Started dbprim00
Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
     Masters: [ dbsec00 ]
     Slaves: [ dbprim00 ]
db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

Daemon Status:
     corosync: active/enabled
     pacemaker: active/enabled
     pcsd: active/enabled
[root@dbprim00 ~]#
```

# Test Case 2: Shut Down the Primary EC2 Instance

**Procedure**: Using AWS Console or CLI to stop the EC2 instance and simulate EC2 failure.

**Expected result**: The Db2 primary node is moved to the standby server.

```
[root@dbsec00 db2stj]# pcs status
Cluster name: db2ha
Stack: corosync
Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
Last updated: Sat Feb  8 09:44:16 2020
Last change: Sat Feb  8 09:31:39 2020 by hacluster via crmd on dbsec00

2 nodes configured
4 resources configured

Online: [ dbsec00 ]
OFFLINE: [ dbprim00 ]

Full list of resources:

clusterfence   (stonith:fence_aws):    Started dbsec00
Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
     Masters: [ dbsec00 ]
     Stopped: [ dbprim00 ]
db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00
```

```
    Daemon Status:
         corosync: active/enabled
         pacemaker: active/enabled
         pcsd: active/enabled
```

**Followup action**: Start the EC2 instance and then start Db2 as standby on the standby instance as you did in Test Case 1 (p. 33). Do not include location constraint removal this time.

# Test Case 3: Stop the Db2 Instance on the Primary Instance

**Procedure**: Log in to the Db2 primary instance as `db2<sid> (db2stj)` and run `db2stop force`.

```
    db2stj> db2stop force
    02/12/2020 12:40:03     0   0   SQL1064N  DB2STOP processing was successful.
    SQL1064N  DB2STOP processing was successful.
```

**Expected result**: The Db2 primary node is failed over to standby server. The standby node continues to be on the old primary in a stopped state. There is a failed monitoring action.

```
    [root@dbsec00 db2stj]# pcs status
    Cluster name: db2ha
    Stack: corosync
    Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
    Last updated: Wed Feb 12 16:55:56 2020
    Last change: Wed Feb 12 13:58:11 2020 by hacluster via crmd on dbsec00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    clusterfence   (stonith:fence_aws):    Started dbsec00
    Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
        Masters: [ dbsec00 ]
        Stopped: [ dbprim00 ]
    db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

   Failed Actions:
   * Db2_HADR_STJ_start_0 on dbprim00 'unknown error' (1): call=34, status=complete,
exitreason='',
   last-rc-change='Wed Feb 12 16:55:32 2020', queued=1ms, exec=6749ms


   Daemon Status:
      corosync: active/enabled
      pacemaker: active/enabled
   pcsd: active/enabled
   [root@dbsec00 db2stj]#
```

**Followup action**: Start the EC2 instance, then start Db2 as standby on the standby instance as you did in Test Case 2 (p. 35). Clear the failed monitoring error.

# Test Case 4: End the Db2 Process (db2sysc) on the Node that Runs the Primary Database

**Procedure**: Log in to the Db2 primary instance as root and then run `ps -ef|grep db2sysc`. Note the process ID (PID) and then end it.

```
[root@dbprim00 ~]# ps -ef|grep db2sysc
root      5809 30644  0 18:54 pts/1    00:00:00 grep --color=auto
db2sysc
db2stj   26982 26980  0 17:12 pts/0    00:00:28 db2sysc 0
[root@dbprim00 ~]# kill -9 26982
```

**Expected result**: The Db2 primary node is failed over to the standby server. The standby node is in the old primary in a stopped state.

```
[root@dbprim00 ~]# pcs status
Cluster name: db2ha
Stack: corosync
Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
Last updated: Wed Feb 12 18:54:50 2020
Last change: Wed Feb 12 18:53:12 2020 by hacluster via crmd on dbsec00

2 nodes configured
4 resources configured

Online: [ dbprim00 dbsec00 ]

Full list of resources:

clusterfence   (stonith:fence_aws):    Started dbsec00
Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
    Masters: [ dbsec00 ]
    Stopped: [ dbprim00 ]
db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

Failed Actions:
* Db2_HADR_STJ_start_0 on dbprim00 'unknown error' (1): call=57, status=complete,
exitreason='',
   last-rc-change='Wed Feb 12 18:54:37 2020', queued=0ms, exec=6777ms


Daemon Status:
   corosync: active/enabled
   pacemaker: active/enabled
   pcsd: active/enabled
```

**Followup action**: Start the EC2 instance and start Db2 as standby on the standby instance, as you did in Test Case 2 (p. 35). Clear the failed monitoring alert.

# Test Case 5: End the Db2 Process (db2sysc) on the Node that Runs the Standby Database

**Procedure**: Log in to the Db2 standby instance as root and run `ps -ef|grep db2sysc`. Note the PID and then end it.

```
[root@dbsec00 db2stj]# ps -ef|grep db2sysc
db2stj   24194 24192  1 11:55 pts/1    00:00:01 db2sysc 0
root     26153  4461  0 11:57 pts/0    00:00:00 grep --color=auto
db2sysc
[root@dbsec00 db2stj]# kill -9 24194
```

**Expected result**: The `db2sysc` process is restarted on the Db2 standby instance. There is a monitoring failure event record in the cluster.

```
[root@dbprim00 ~]# pcs status
Cluster name: db2ha
Stack: corosync
Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 14 11:59:22 2020
Last change: Fri Feb 14 11:55:54 2020 by hacluster via crmd on dbsec00

2 nodes configured
4 resources configured

Online: [ dbprim00 dbsec00 ]

Full list of resources:

clusterfence   (stonith:fence_aws):    Started dbsec00
Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
    Masters: [ dbprim00 ]
    Slaves: [ dbsec00 ]
db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbprim00

Failed Actions:
* Db2_HADR_STJ_monitor_20000 on dbsec00 'not running' (7): call=345, status=complete,
exitreason='',
   last-rc-change='Fri Feb 14 11:57:57 2020', queued=0ms, exec=0ms


Daemon Status:
   corosync: active/enabled
   pacemaker: active/enabled
   pcsd: active/enabled


[root@dbsec00 db2stj]# ps -ef|grep db2sysc
db2stj   26631 26629  1 11:57 ?        00:00:01 db2sysc 0
root     27811  4461  0 11:58 pts/0    00:00:00 grep --color=auto db2sysc
```

**Follow-up action**: Clear the monitoring error.

# Test Case 6: Simulating a Crash of the Node that Runs the Primary Db2

**Procedure**: Log in to the Db2 primary instance as root and run `echo 'c' > /proc/sysrq-trigger`.

```
[root@dbprim00 ~]# echo 'c' > /proc/sysrq-trigger

##############################################################################################################
```

```
    Session stopped
        - Press <return> to exit tab
        - Press R to restart session
        - Press S to save terminal output to file

    Network error: Software caused connection abort
```

**Expected result**: The primary Db2 should failover to standby node. The standby is in a stopped state on the previous primary.

```
    [root@dbsec00 ~]# pcs status
    Cluster name: db2ha
    Stack: corosync
    Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
    Last updated: Fri Feb 21 15:38:43 2020
    Last change: Fri Feb 21 15:33:17 2020 by hacluster via crmd on dbsec00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

     clusterfence   (stonith:fence_aws):    Started dbsec00
     Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
        Masters: [ dbsec00 ]
        Stopped: [ dbprim00 ]
     db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

    Failed Actions:
    * Db2_HADR_STJ_start_0 on dbprim00 'unknown error' (1): call=15, status=complete,
exitreason='',
    last-rc-change='Fri Feb 21 15:38:31 2020', queued=0ms, exec=7666ms


    Daemon Status:
       corosync: active/enabled
       pacemaker: active/enabled
       pcsd: active/enabled
```

**Followup action**: Start the EC2 instance and then start Db2 as standby on the standby instance as you did in Test Case 2 (p. 35). Clear the monitoring error.

# Appendix 2: Testing on SLES Setup

## Test Case 1: Manual Failover

**Procedure**: Use the command `crm resource move <Db2 primary resource name> force` to move the primary Db2 instance to standby node.

```
    dbprim00: # crm resource move msl_db2_db2stj_STJ force
    INFO: Move constraint created for rsc_db2_db2stj_STJ
```

**Expected result**: The Db2 primary node is moved from the primary node (`dbprim00`) to the standby node (`dbsec00`).

```
    dbprim00:~ # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:03:20 2020
    Last change: Sat Apr 25 19:02:26 2020 by root via crm_resource on dbprim00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbsec00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
        Masters: [ dbsec00 ]
        Stopped: [ dbprim00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbsec00
```

**Follow-up actions**: Start the Db2 instance as standby on the new standby node, logged in as `db2<sid>`. Clean up the error logged in as root.

```
    db2stj> db2start
    04/25/2020 19:05:27     0   0   SQL1063N  DB2START processing was successful.
    SQL1063N  DB2START processing was successful.

    db2stj> db2 start hadr on database STJ as standby
    DB20000I  The START HADR ON DATABASE command completed successfully.
```

**Remove location constraint**: When using a manual command to move the resource, there is a location constraint created on the node (in this case primary node) which is run, preventing the Db2 resource from running in standby mode.

Use the following command to remove the location constraint.

```
     # dbprim00: # crm resource clear msl_db2_db2stj_STJ
```

Once the constraint is removed, the standby instance starts automatically.

```
     # dbprim00: # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:05:29 2020
    Last change: Sat Apr 25 19:05:18 2020 by root via crm_resource on dbprim00

    2 nodes configured
```

```
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbsec00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
         Masters: [ dbsec00 ]
        Slaves: [ dbprim00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbsec00
```

# Test Case 2: Shut Down the Primary EC2 Instance

**Procedure**: Using AWS console or CLI, stop the EC2 instance to simulate EC2 failure.

Expected Result: The Db2 primary node is moved to a standby server (`dbsec00`).

```
    dbsec00:~ # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:19:32 2020
    Last change: Sat Apr 25 19:18:16 2020 by root via crm_resource on dbprim00

    2 nodes configured
    4 resources configured

    Online: [ dbsec00 ]
    OFFLINE: [ dbprim00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbsec00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
        Masters: [ dbsec00 ]
        Stopped: [ dbprim00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbsec00
```

**Follow-up action**: Start the EC2 instance and the standby node should start on `dbprim00`.

# Test Case 3: Stop the Db2 Instance on the Primary Instance

**Procedure**: Log in to the Db2 primary instance (`dbprim00`) as `db2<sid> (db2stj)` and run db2stop force.

```
    db2stj> db2stop force
    02/12/2020 12:40:03     0   0   SQL1064N  DB2STOP processing was successful.
    SQL1064N  DB2STOP processing was successful.
```

**Expected result**: The Db2 primary node will failover on primary instance. The standby remains on the standby instance. There is a failed resource alert.

```
    dbsec00:~ # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:29:38 2020
    Last change: Sat Apr 25 19:23:04 2020 by root via crm_resource on dbprim00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbprim00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
        Masters: [ dbprim00 ]
        Slaves: [ dbsec00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbprim00

   Failed Resource Actions:
   * rsc_db2_db2stj_STJ_demote_0 on dbprim00 'unknown error' (1): call=74,
status=complete, exitreason='',
   last-rc-change='Sat Apr 25 19:27:21 2020', queued=0ms, exec=175ms
```

**Followup action**: Clear the failed cluster action.

```
    dbsec00:~ # crm resource cleanup
    Waiting for 1 reply from the CRMd. OK
```

# Test Case 4: End the Db2 Process (db2sysc) on the Node that Runs the Primary Database

**Procedure**: Log in to the Db2 primary instance as root and run `ps -ef|grep db2sysc`. Note the PID and then end it.

```
    dbprim00:~ # ps -ef|grep db2sysc
    db2stj   11690 11688  0 19:27 ?        00:00:02 db2sysc 0
    root     15814  4907  0 19:31 pts/0    00:00:00 grep --color=auto db2sysc
    [root@dbprim00 ~]# kill -9 11690
```

**Expected result**: The Db2 primary node is restarted on the primary instance. The standby node remains on the standby instance. There is a failed resource alert.

```
    dbsec00:~ # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:29:38 2020
    Last change: Sat Apr 25 19:23:04 2020 by root via crm_resource on dbprim00

    2 nodes configured
```

```
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbprim00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
        Masters: [ dbprim00 ]
        Slaves: [ dbsec00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbprim00

    Failed Resource Actions:
    * rsc_db2_db2stj_STJ_demote_0 on dbprim00 'unknown error' (1): call=74,
status=complete, exitreason='',
   last-rc-change='Sat Apr 25 19:27:21 2020', queued=0ms, exec=175ms
```

**Followup action**: Clear the failed cluster action.

```
    dbsec00:~ # crm resource cleanup
    Waiting for 1 reply from the CRMd. OK
```

# Test Case 5: End the Db2 Process (db2sysc) on the Node that Runs the Standby Database

**Procedure**: Log in to the standby DB instance (`dbsec00`) as root, then run `ps -ef|grep db2sysc`. Note the PID and then end it.

```
    dbsec00:~ # ps -ef| grep db2sysc
    db2stj   16245 16243  0 19:23 ?        00:00:04 db2sysc 0
    root     28729 28657  0 19:38 pts/0    00:00:00 grep --color=auto db2sysc
    dbsec00:~ # kill -9 16245
```

**Expected result**: The `db2sysc` process is restarted on the standby DB instance. There is a monitoring failure event recorded in the cluster.

```
    dbsec00:~ # crm status
    Stack: corosync
    Current DC: dbsec00 (version
1.1.19+20181105.ccd6b5b10-3.16.1-1.1.19+20181105.ccd6b5b10) - partition with quorum
    Last updated: Sat Apr 25 19:40:23 2020
    Last change: Sat Apr 25 19:23:04 2020 by root via crm_resource on dbprim00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    res_AWS_STONITH        (stonith:external/ec2): Started dbprim00
    Master/Slave Set: msl_db2_db2stj_STJ [rsc_db2_db2stj_STJ]
        Masters: [ dbprim00 ]
```

```
        Slaves: [ dbsec00 ]
    res_AWS_IP     (ocf::suse:aws-vpc-move-ip):    Started dbprim00

    Failed Resource Actions:
    * rsc_db2_db2stj_STJ_monitor_30000 on dbsec00 'not running' (7): call=387,
status=complete, exitreason='',
   last-rc-change='Sat Apr 25 19:39:24 2020', queued=0ms, exec=0ms
```

**Followup action**: Clear the monitoring error.

```
    dbsec00:~ # crm resource cleanup
    Waiting for 1 reply from the CRMd. OK
```

# Test Case 6: Simulating a Crash of the Node that Runs the Primary Db2

**Procedure**: Log in to the Db2 primary instance as root, then run `echo 'c' > /proc/sysrq-trigger`.

```
    dbprim00:~ # echo 'c' > /proc/sysrq-trigger
    Session stopped
        - Press <return> to exit tab
        - Press R to restart session
        - Press S to save terminal output to file

    Network error: Software caused connection abort
```

**Expected result**: The primary Db2 should failover to standby node.vThe standby is in a stopped state on the previous primary (`dbprim00`).

```
    [root@dbsec00 ~]# crm status
    Cluster name: db2ha
    Stack: corosync
    Current DC: dbsec00 (version 1.1.18-11.el7_5.4-2b07d5c5a9) - partition with quorum
    Last updated: Fri Feb 21 15:38:43 2020
    Last change: Fri Feb 21 15:33:17 2020 by hacluster via crmd on dbsec00

    2 nodes configured
    4 resources configured

    Online: [ dbprim00 dbsec00 ]

    Full list of resources:

    clusterfence   (stonith:fence_aws):    Started dbsec00
    Master/Slave Set: Db2_HADR_STJ-master [Db2_HADR_STJ]
        Masters: [ dbsec00 ]
        Stopped: [ dbprim00 ]
    db2-oip        (ocf::heartbeat:aws-vpc-move-ip):       Started dbsec00

    Failed Actions:
    * Db2_HADR_STJ_start_0 on dbprim00 'unknown error' (1): call=15, status=complete,
exitreason='',
   last-rc-change='Fri Feb 21 15:38:31 2020', queued=0ms, exec=7666ms
```

```
Daemon Status:
   corosync: active/enabled
   pacemaker: active/enabled
   pcsd: active/enabled
```

**Followup action**: Start the EC2 instance and then start Db2 as standby on the standby instance as you did in Test Case 2 (p. 35).

# FAQ

**Question:** Can I use Database Migration Service to migrate and deploy SAP NetWeaver on IBM Db2 based applications?

**Answer:** No, AWS DMS supports IBM Db2 as a source, but it is not certified by SAP for SAP NetWeaver-based applications.

# Contributors

Contributors to this document include:

- Manas Srivastava Sr. Partner SA, SAP
- Somckit Khemmanivanh Sr. SysDev, SAP

# Additional Reading

SAP on AWS technical documentation:

- SAP on AWS documentation
- SAP on AWS Whitepapers
- SAP on AWS Blog

SAP documentation:

- SAP Knowledge Base
- SAP Product Availability Matrix
- SAP Quick Sizer
- TCP/IP Ports of All SAP Products

# Document Revisions

| Date | Change |
|---|---|
| **December 2020** | First publication |

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.