SUSE Linux Enterprise Server for SAP Applications 12 SP3 for the AWS Cloud - Setup Guide

Publication Date: 2018-12-20

Contents

Fabian Herschel, Bernd Schubert, Stefan Schneider (AWS), Martin Tegtmeier (AWS), Guilherme G. Felix (AWS)

Revision 1.1 from 2018-12-20

1 SUSE Linux Enterprise Server for SAP Applications 12 SP3

1.1 SAP HANA SR Performance Optimized Scenario on the AWS Cloud

SUSE LLC

10 Canal Park Drive

Suite 200

Cambridge MA 02142

USA

http://www.suse.com/documentation

1.2 About This Guide

SUSE® Linux Enterprise Server for SAP Applications is optimized in various ways for SAP* applications. This guide provides detailed information about installing and customizing SUSE Linux Enterprise Server for SAP Applications for SAP HANA system replication in the performance optimized scenario.

1.2.1 Introduction

“SAP customers invest in SAP HANA” is the conclusion reached by a recent market study carried out by Pierre Audoin Consultants (PAC). In Germany alone, half of the companies expect SAP HANA to become the dominant database platform in the SAP environment. In many cases, the “SAP Business Suite* powered by SAP HANA*” scenario is already being discussed in concrete terms.

Naturally, SUSE is also accommodating this development by providing SUSE Linux Enterprise Server for SAP Applications – the recommended and supported operating system for SAP HANA. In close collaboration with SAP and hardware partners, therefore, SUSE provides two resource agents for customers to ensure the high availability of SAP HANA system replications.

1.2.1.1 Scale-Up vs. Scale-Out

The first set of scenarios include the architecture and development of scale-up solutions. For this scenarios SUSE developed the scale-up resource agent package SAPHanaSR. System replication will help to replicate the database data from one computer to another computer in order to compensate for database failures (single-box replication).

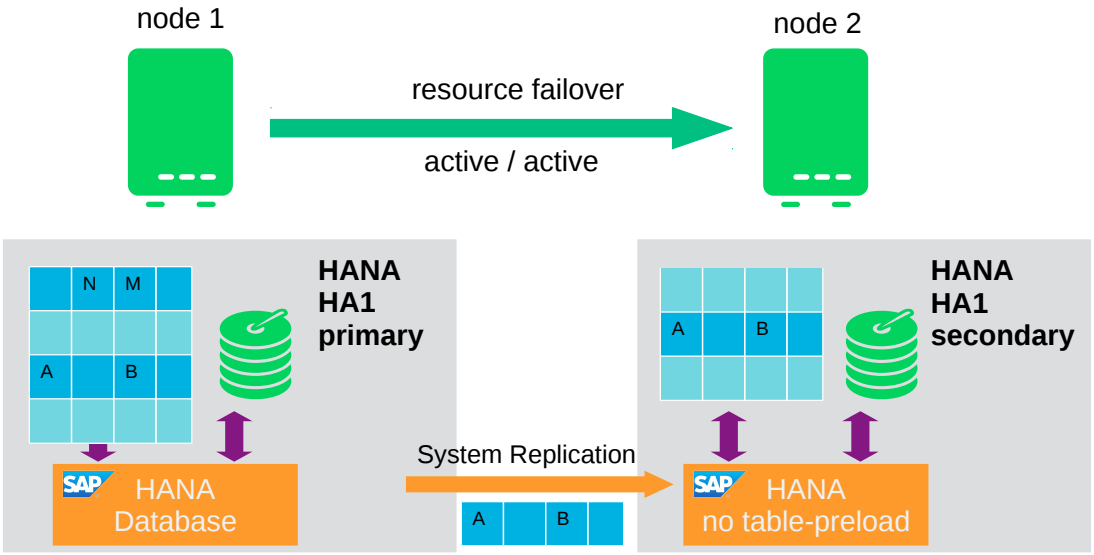

FIGURE 1: SAP HANA SYSTEM REPLICATION IN THE CLUSTER

The second set of scenarios include the architecture and development of scale-out solutions (multibox replication). For this scenarios SUSE developed the scale-out resource agent package SAPHanaSR-ScaleOut. With this mode of operation, internal SAP HANA high-availability (HA) mechanisms and the resource agent must work together or be coordinated with each other. SAP HANA system replication automation for scale-out will be described in an own document available in the resource library at https://www.suse.com/products/sles-for-sap/resource-library/.

1.2.2 Scale-Up Scenarios and Resource Agents

SUSE has implemented the scale-up scenario with the SAPHana resource agent (RA), which performs the actual check of the SAP HANA database instances This RA is configured as a master/slave resource. In the scale-up scenario, the master assumes responsibility for the SAP HANA databases running in primary mode, and the slave is responsible for instances that are operated in synchronous (secondary) status.

To make configuring the cluster as simple as possible, SUSE also developed it's SAPHanaTopology resource agent. This runs on all nodes of an SLE 12 HAE cluster and gathers information about the states and configurations of SAP HANA system replications. It is designed as a normal (stateless) clone.

SAP HANA System replication for Scale-Up is supported in the following scenarios or use cases:

- Performance optimized (A ⇒ B): This scenario and setup is described in this document. In the performance optimized scenario a HANA RDBMS A is synchronizing with a HANA RDBMS B on a second node. As the HANA RDBMS B on the second node is configured to preload the tables the takeover time is typically very short.

- Cost optimized (A ⇒ B, Q): This scenario and setup is described in an other document in the resource library (https://www.suse.com/products/sles-for-sap/resource-library/). In the cost optimized scenario the second node is also used for a non-productive HANA RDBMS system (like QAS or TST). Whenever a takeover is needed the non-productive system must be stopped first. As the productive secondary system on this node must be limited in using system resources, the table preload must be switched off and a possible takeover needs longer than in the performance optimized use case.

- Multi Tier (A ⇒ B → C): This scenario and setup is described in an other document in the resource library (https://www.suse.com/products/sles-for-sap/resource-library/). A Multi Tier system replication has an additional target, which must be connected to the secondary (chain topology).

- Multi-tenancy or MDC: Multi-tenancy is supported for all above scenarios and use cases. This scenario is supported since SAP HANA SPS09. The set-up and configuration from cluster point of view is the same for multi-tenancy and single container, so you can use the above documents for both kinds of scenarios. In our notation we mark a MDC like %A. This means MDC, performance optimized is abbreviated as %A ⇒ %B.

1.2.3 The Concept of the Performance Optimized Scenario

In case of failure of the primary SAP HANA on node 1 (node or database instance) the cluster first tries to start the takeover process. This allows to use the already loaded data at the secondary site. Typically the takeover is much faster than the local restart.

To achieve an automation of this resource handling process, we can utilize the SAP HANA resource agents included in SAPHanaSR. System replication of the productive database is automated with SAPHana and SAPHanaTopology.

You can setup the level of automation by setting the parameter **AUTOMATED_REGISTER**. If automated registration is activated the cluster will also automatically register a former failed primary to get the new secondary.

Note

The solution is not designed to manually 'migrate' the primary or secondary instance using HAWK or any other cluster client commands. We plan to include a migration support with one of the next resource agent updates. In the admin section we describe how to 'migrate' the pimary to the seconadary site.

1.2.4 Customers Receive Complete Package

With both the SAPHana and SAPHanaTopology resource agents, customers will therefore be able to integrate SAP HANA system replications in their cluster. This has the advantage of enabling companies to use not only their business-critical SAP systems but also their SAP HANA databases without interruption while noticeably reducing needed budgets. SUSE provides the extended solution together with best practices documentation.

SAP and hardware partners who do not have their own SAP HANA high-availability solution will also benefit from this SUSE Linux development.

1.3 Additional Documentation and Resources

Chapters in this manual contain links to additional documentation resources that are either available on the system or on the Internet.

For the latest documentation updates, see http://www.suse.com/documentation.

You can also find numerous white papers, a best-practices guide, and other resources at the SUSE Linux Enterprise Server for SAP Applications resource library: https://www.suse.com/products/sles-for-sap/resource-library/.

1.4 Feedback

Several feedback channels are available:

- Bugs and Enhancement Requests

- For services and support options available for your product, refer to http://www.suse.com/support/.
- To report bugs for a product component, go to https://scc.suse.com/support/ requests, log in, and select Submit New SR.

- User Comments
  - We want to hear your comments about and suggestions for this manual and the other documentation included with this product. Use the User Comments feature at the bottom of each page in the online documentation or go to http://www.suse.com/doc/feedback.html and enter your comments there.
- Mail
  - For feedback on the documentation of this product, you can also send a mail to doc-team@suse.de (mailto:doc-team@suse.de). Make sure to include the document title, the product version and the publication date of the documentation. To report errors or suggest enhancements, provide a concise description of the problem and refer to the respective section number and page (or URL).

1.5 Documentation Conventions

The following typographical conventions are used in this manual:

- /etc/passwd: directory names and file names
- *placeholder*: replace *placeholder* with the actual value
- PATH: the environment variable PATH
- **ls**, --help: commands, options, and parameters
- user: users or groups
- Alt, Alt–F1: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* › *Save As*: menu items, buttons
- *Dancing Penguins* (Chapter Penguins, ↑Another Manual): This is a reference to a chapter in another manual.

2 Supported Scenarios and Prerequisites

With the SAPHanaSR resource agent software package, we limit the support to Scale-Up (single-box to single-box) system replication with the following configurations and parameters:

- Two-node cluster.
- The cluster must include a valid STONITH method.
- Use one DHCP assigned IP address on one ENI (Elastic Network Interface) Only per cluster node.
    - The AWS STONITH mechanism is supported by SLE 12 HAE is supported with SAPHanaSR.
- Technical users and groups, such as *<sid>adm* are defined locally in the Linux system.
- Name resolution of the cluster nodes and the virtual IP address must be done locally on all cluster nodes.
- Time synchronization between the cluster nodes using NTP.
- Both SAP HANA instances have the same SAP Identifier (SID) and instance number.
- If the cluster nodes are installed in different data centers or data center areas, the environment must match the requirements of the SLE HAE cluster product. Of particular concern is the network latency and recommended maximum distance between the nodes. Please review our product documentation for SLE HAE about those recommendations.
- Automated registration of a failed primary after takeover.
    - As a good starting configuration for projects, we recommend to switch off the automated registration of a failed primary. The setup AUTOMATED_REGISTER = "false" is the default. In this case, you need to register a failed primary after a takeover manually. Use SAP tools like hanastudio or hdbnsutil.
    - For optimal automation, we recommend AUTOMATED_REGISTER = "true".
- Automated start of SAP HANA instances during system boot must be switched off.

You need at least SAPHanaSR version 0.152, SUSE Linux Enterprise Server for SAP Applications 12 SP2 and SAP HANA SPS09 (095) for all mentioned setups.

Note

Valid STONITH Method: Without a valid STONITH method, the complete cluster is unsupported and will not work properly.

This setup-guide focuses on the performance optimized setup. No other SAP HANA system (like QAS or TST) on the replicating node which needs to be stopped during takeover is considered in this scenario.

- Cost optimized setups with a non-productive SAP HANA (like QAS or TST) on the secondary node are supported. SUSE has published a setup-guide for cost optimized scenarios too (↑ SAP HANA SR Cost Optimized Scenario). We already have a SCN article for this kind of setups (http://scn.sap.com/docs/DOC-65899). You need to configure the SAP HANA database resources at the secondary side to run them side by side (see SAP documentation for cost optimized setups).
- Multi-tier setups (A ⇒ B → C) are supported. You need to disable automated re-registration (AUTOMATED_REGISTER = "false"), because SAP currently does not allow two targets to be connected to one primary (↑ SAP HANA SR Multi Tier Scenario).
- Multi-tenancy (MDC) databases are supported.
  - Multi-tenancy databases could be used in combination with any other setup (performance based, cost optimized and multi-tier).
  - In MDC configurations the SAP HANA RDBMS is treated as a single system including all database containers. Therefore cluster takeover decisions are based on the complete RDBMS status independent of the status of individual containers.
  - You need SAP HANA version >= SPS10 rev3 or SPS11+, if you need to stop tenants during production and like the cluster to be able to takeover. Older SAP HANA versions are marking the system replication as failed, if you stop a tenant.

If you need to implement a different scenario, we strongly recommend to define a PoC with SUSE. This PoC will focus on testing the existing solution in your scenario. The limitation of most of the above items is mostly due to testing limits.

Besides SAP HANA, you need SAP Host Agent to be installed on your system.

3 Scope of this Documentation

This document describes how to setup the cluster to control SAP HANA in System Replication Scenarios. The document focuses on the steps to integrate an already installed and working SAP HANA with System Replication.

This setup builds a SAP HANA HA cluster in two data-centers in Walldorf (WDF) and in Rot (ROT), installed on two SLES for SAP 12 SP3 systems.

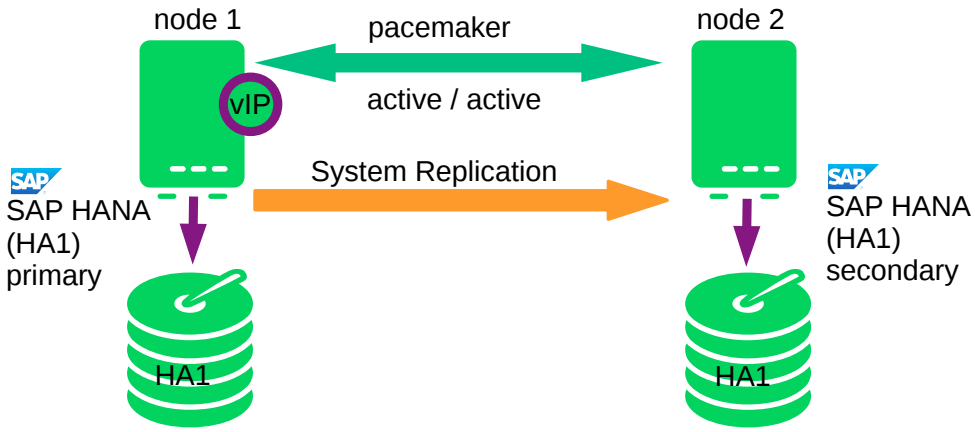

FIGURE 2: CLUSTER WITH SAP HANA SR - PERFORMANCE OPTIMIZED

TABLE 1: PARAMETERS USED IN THIS DOCUMENT

Parameter	Value	Role
Cluster node 1	suse01, 192.168.1.11	Cluster node name and IP address.
Cluster node 2	suse02, 192.168.1.12	Cluster node name and IP address.
SID	HA1	SAP Identifier
Instance number	10	Number of the SAP HANA database. For system replication also Instance Number + 1 is blocked.
Network mask	255.255.255.0	
Virtual IP address	10.0.0.1	

Parameter	Value	Role
Storage		Storage for HDB data and log files is connected “locally” (per node; not shared)
HAWK Port	7630	
NTP Server		Address or name of your time server

4 Using AWS Architectures in SLES pacemaker clusters

SLES pacemaker clusters will be installed in an AWS region. An AWS region consists of multiple availability zones. Availability zones are located in different data centers which are 10 to 50km apart. Availability zones have independent flood levels, electricity and network hookup. They are supposed to be independent. AWS recommends architectural patterns where redundant cluster nodes are being spread across availability zones (AZs) in order to allow a customer to overcome individual AZ failures.

An AWS Virtual Private Network (VPC) is spanning all AZs. We assume that a customer will have:

- identified two availability zones to be used
- created subnets in the two AZs which can host the two nodes of a SLES HAE cluster
- use a routing table which is attached to the two subnets

The virtual IP address for the HANA services will be an AWS Overlay IP address. This is an AWS specific routing entry which can send network traffic to an instance, no matter which AZ the instance is located in.

The SLES HAE cluster will update this routing entry as it is required. All SAP system components in the VPC will be able to reach an AWS instance with a SAP system component inside a VPC through this Overlay IP address.

Overlay IP addresses have one disadvantage, they have to to have a CIDR range which is outside of the VPC. Otherwise they would be part of a subnet and a given availability zone.

On premises users like HANA Studio will not be able to reach this IP address since the AWS Virtual Private Network (VPN) gateway will not route traffic to such an IP address.

4.1 Prerequisites for the AWS specific HA Installation

There are a number of prerequisites which have to be met before starting the installation:

- Have an AWS account
- Have an AWS user with admin rights. At least rights to:
  - Create security groups
  - Modify AWS routing tables
  - Create policies and attach them to IAM roles
- Understand your landscape:
  - Know your region and it's AWS name
  - Know your VPC and it's AWS id
  - Know which availability zones you want to use in your VPC
  - Have a subnet in each of the availability zones:
    - Have a routing table which is implicitly or explicitly attached to the two subnets
    - Have free IP addresses in the two subnets for your SAP installation
    - Allow network traffic in between the two subnets
    - Allow outgoing Internet access from the subnets

Please use the check list in the appendix to note down all information needed before starting the installation.

4.1.1 Security Groups

This section does not cover a discussion of SAP related ports in security groups. Add the ports and protocols being used by your HANA instance to the security group. This section lists the ports which need to be available for the SUSE cluster only.

The following ports and protocols need to be configured to allow the two cluster nodes to communicate with each other:

- Port 5405 for inbound UDP: Used to configure the corosync communication layer. Port 5405 is being used in common examples. A different port may be used depending on the corosync configuration.
- Port 7630 for inbound TCP: Used by the SUSE "hawk" web GUI.
- enable ICMP: Used through a ping command in the AWS IP-move agent of the SUSE cluster.

We assume that there are no restriction for outbound network communication.

4.1.2 AWS EC2 Instance Creation

Create the two EC2 instances to build up your SLE HAE cluster. The EC2 instances will be most likely located in two different availability zones to make them independent of each other.

AMI selection:

- Use a "SLES for SAP" AMI. Search for "suse-sles-sap-12-sp3" in the list of public AMIs. There is currently (May 2018) a BYOS (Bring you own Subscription) AMI available. Use this AMI to create a SAP HANA compliant configuration. Register the systems!
- Use the AWS Marketplace AMI *SUSE Linux Enterprise Server for SAP Applications 12 SP3* which already includes the required SUSE subscription.

Instantiate the two instances in two subnets belonging to two AWS Availability Zones with a joint routing table. The subnets have to be able to communicate with each other.

Note

Use a "SLES for SAP" AMI which includes the SLES HA Extension. SUSE will not allow to migrate from generic SLES AMIs to "SLES for SAP" AMIs on AWS!

4.1.3 Tagging the EC2 Instances

The EC2 instances will have host names which are automatically generated. Select host names which comply with SAP requirements, see SAP note 611361.

The cluster agents will have to be able to identify the EC2 instances in the correct way. This happens through instance tags.

Tag the two EC2 instances through the console or the AWS Command Line Interface (CLI) with arbitrarily chosen tag like *pacemaker* and the host name as it will be shown in the command *uname*. Use the same tag (like *pacemaker*) and the individual host names for both instances.

The screen shot below has been created by identifying an EC2 instance at the console. The last tab Tags has been clicked. The button Add/Edit Tags has then been clicked. A new tag with the key pacemaker and the hostname has been created. The host name in this example has been *suse-node52* .

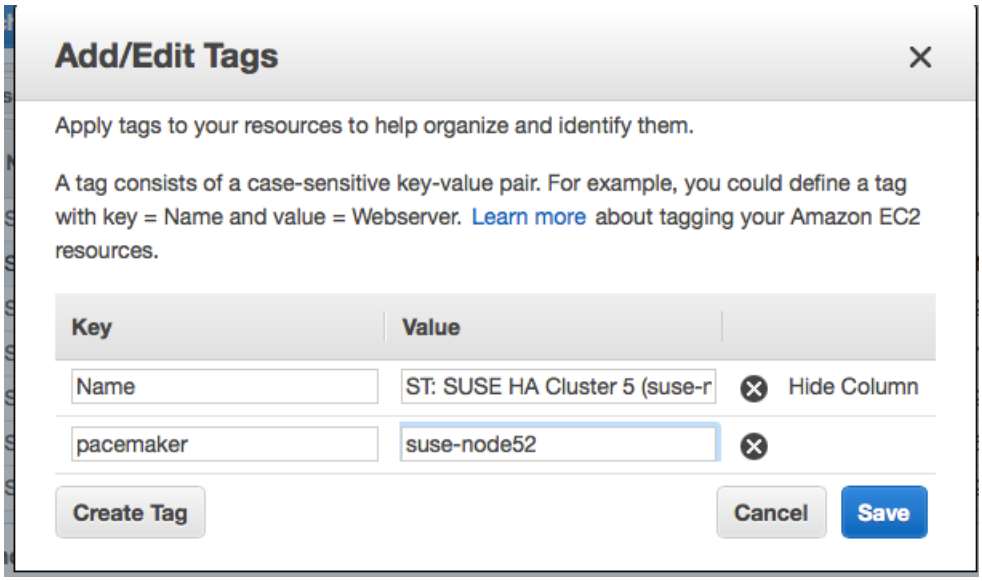

FIGURE 3: TAG EC2 INSTANCE

Tag both of your instances this way.

The AWS documentation (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html) explains how to tag EC2 instances.

Note

Use only ASCII characters in any AWS tag assigned to cluster managed resources.

4.1.4 Creating an AWS CLI Profile on both EC2 Instances

The SLES agents use the AWS Command Line Interface (CLI). They will use an AWS CLI profile which needs to be created for the root account *root* on both instances. The SUSE resources require a profile which creates output in text format. The name of the profile is arbitrary. The

name chosen in this example is *cluster*. The region of the instance needs to be added as well. Replace the string *region-name* with your target region in the following example.

One way to create such a profile is to create a file */root/.aws/config* with the following content:

```
[default]
region = region-name
[profile cluster]
region = region-name
output = text
```

The other way is to use the *aws configure* CLI command in the following way:

```
# aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: _region-name_
Default output format [None]:

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

This command sequence generates a default profile and a cluster profile.

4.1.5 Configure http Proxies

This action is not needed if the system has transparent access to the Internet. The resource agents execute AWS CLI (Command Line Interface) commands. These commands send http/https requests to an access point in the Internet. These access point are usually directly reachable. Systems which don't offer transparent Internet access will have to provide a http/https proxy. The configuration of the proxy access is described in full detail in the AWS documentation.

Please add the following environment variables to the root user's *.bashrc* file:

```
export HTTP_PROXY=http://a.b.c.d:n
export HTTPS_PROXY=http://a.b.c.d:m
```

Please add the following environment variables instead of the ones above if authentication is required:

```
export HTTP_PROXY=http://username:password@a.b.c.d:n
export HTTPS_PROXY=http://username:password@a.b.c.d:m
```

The AWS Data Provider for SAP will need to reach the instance meta data service directly. Please add the following environment variable to the root user's *.bashrc* file:

```
export NO_PROXY=169.254.169.254
```

4.1.5.1 Verify http Proxy Settings

Make sure that the SUSE instance is able to reach the URL http://169.254.169.254/latest/meta-data . The STONITH agent will access it. Allow your firewall to access it. Disable proxy access to this URL. The hypervisor is providing this information. The system will not access the Internet for it.

4.1.6 Disable the Source/Destination Check for the Cluster Instances

The source/destination check has to be disabled. This can be done through scripts using the AWS command line interface (AWS-CLI) or by using the AWS console. The following command needs to be executed one time for both EC2 instances, which are supposed to receive traffic from the Overlay IP address:

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

Replace the variable *EC2-instance* with the instance id of the two cluster AWS EC2 instances. The system on which this command gets executed needs temporarily a role with the following policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Sid": "Stmt1424870324000",
      "Effect": "Allow",
      "Action": [ "ec2:ModifyInstanceAttribute"],
      "Resource": [
      "arn:aws:ec2:region-name:account-id:instance/instance-a",
      "arn:aws:ec2:region-name:account-id:instance/instance-b"
      ]
   }
   ]
```

```
}
```

Replace the following individual parameter with the appropriate values:

- region-name (Example: us-east-1)
- account-id (Example: 123456789012)
- instance-a, instance-b (Example: i-1234567890abcdef)

Note

The string "instance" in the policy is a fixed string. It is not a variable which needs to be substituted!

The source/destination check can be disabled as well from the AWS console. It takes the execution of the following pull down menu in the console for both EC2 instances (see below).

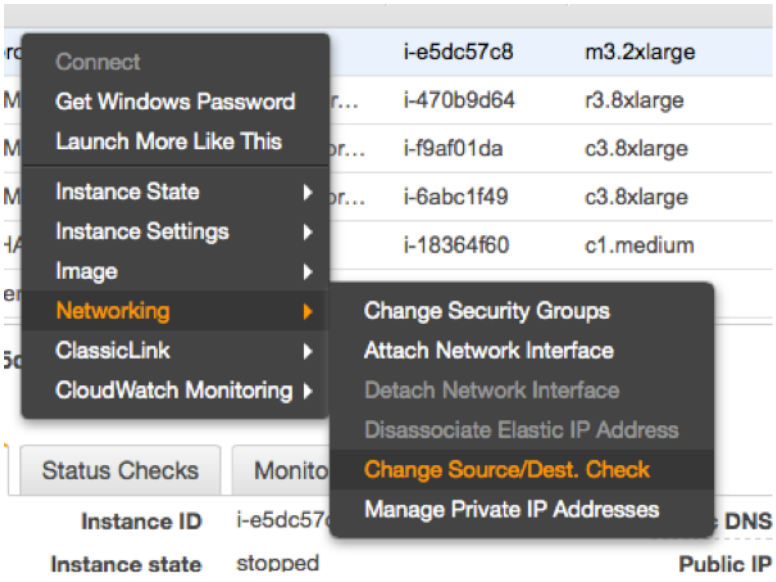

FIGURE 4: DISABLE SOURCE/DESTINATION CHECK AT CONSOLE

4.1.7 Avoid Deletion of Cluster Managed IP Address on the eth0 Interface

SLES 12 SP3 is the first SLES version which ships the cloud-netconfig package. This package will remove any secondary IP address which is managed by the cluster agents from the eth0

interface. This can cause service interruptions for users of the HA service. Perform the following task on all cluster nodes.

Check whether the package cloud-netconfig-ec2 is installed with the command

```
# zypper info cloud-netconfig-ec2
```

Update the file */etc/sysconfig/network/ifcfg-eth0* if this package is installed. Change the following line to a „no“ setting or add the line if the package isn't yet installed:

```
CLOUD_NETCONFIG_MANAGE='no'
```

4.1.8 AWS Roles and Policies

The HANA database server and replication server will run the SLES Pacemaker software and the agents. This software needs a number of AWS IAM privileges to operate the cluster. Create a new Role for every HANA cluster and associate this role to the two instances. Attach the following policies to this role.

4.1.8.1 AWS Data Provider Policy

Every cluster node will operate a SAP system. SAP systems on AWS require an installation of the “AWS Data Provider for SAP”. The data provider needs a policy to access AWS resources. Use the policy as described in the “AWS Data Provider for SAP Installation and Operations Guide“ section IAM Roles and attach it to the role of the instance. This policy can be used by all SAP systems. It takes only one policy in an AWS account. Use an existing one or create it. The policy doesn't contain any instance specific privileges. Attach this policy to the role of the two cluster instances.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "EC2:DescribeInstances",
                "EC2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "cloudwatch:GetMetricStatistics",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::aws-data-provider/config.properties"
        }
    ]
}
```

4.1.8.1.1 STONITH Policy

The instances of the SUSE cluster will need the privilege to start and stop the other nodes in the cluster. Create a policy with a name like *stonith-policy* with the following content and attach it to the cluster role:

```
{
    "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/instance-a",
                "arn:aws:ec2:region-name:account-id:instance/instance-b"

            ]
```

```
        }
    ]
}
```

Replace the variable *aws-account* with the appropriate AWS account identifier. Replace the variables *i-node1* and *i-node2* with the AWS instance-ids of your two cluster nodes. This policy is dependent of the instances of your cluster. You will need a separate policy for every cluster!

4.1.8.2 Overlay IP Agent Policy

The Overlay IP agent will change a routing entry in an AWS routing table. Create a policy with a name like *Manage-Overlay-IP-Policy* and attach it to the role of the cluster instances:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1424870324000",
          "Action": [
            "ec2:DescribeRouteTables",
            "ec2:ReplaceRoute"
          ],
            "Effect": "Allow",
            "Resource": "arn:aws:ec2:region-name:account-id:route-table/rtb-XYZ"
        }
    ]
}
```

This policy allows the agent to update the routing tables which get used. 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 getting used
- rtb-XYZ : The identifier of the routing table which needs to be updated

4.1.9 Add Overlay IP Addresses to Routing Table

Manually add a routing entry to the routing table which is assigned to the two subnets. This IP address is the virtual service IP address of the HANA cluster. The IP address has to be outside of the CIDR range of the VPC. Use the AWS console and search for “VPC”.

- Select VPC
- Click on “Route Tables” in the left column
- Select route table used the subnets from one of your SAP EC2 instances and their application servers
- Click on tabulator “Routes”
- Click on “Edit”
- Scroll to the end of the list and click on “Add another route”

Add the service IP address of the HANA database. Use as filter /32 (example: 192.168.10.1/32). Add the Elastic Network Interface (ENI) name to one of your existing instance. The resource agent will modify this later one automatically. Save your changes by clicking on “Save”.

Note

Important: The routing table, which will contain the routing entry has to be inherited to all subnets in the VPC which have consumers of the service. Please check the AWS VPC documentation (http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html) for more details on routing table inheritance.

4.2 Optional: Prepare System to support "crm report" Command

The **crm report** command generates a consolidated report of all Linux Cluster nodes. It is a very valuable command which helps to analyze a cluster.

Amazon strongly recommends to disable remote root login. In order to run the SLES **crm report** command in such an environment you have to enable the hacluster OS-user to execute crm shell commands as well as the hb_report command. A certificate based passwordless login of the hacluster user has to be configured across all cluster nodes:

1. Edit the sudoers file on all cluster nodes with the command **visudo** and configure cluster nodes, crm commands and user privileges for user hacluster:

```
Host_Alias COROSYNC_NODES = <node1>, <node2>, ...
Cmnd_Alias CRM_SHELL = /usr/sbin/crm*, /usr/share/crmsh/hb_report/hb_report
 hacluster
```

```
COROSYNC_NODES=(root : root) NOPASSWD: CRM_SHELL
```

2. Create a new ssh keypair (Command: ssh-keygen -t rsa) on one cluster node and use the private key as the root user's ssh id on all cluster nodes (/root/.ssh/id_<key-type>). Save the public key to the file authorized_keys for the user hacluster on all cluster nodes (/var/lib/heartbeat/cores/hacluster/.ssh/authorized_keys).
3. Test the passwordless ssh connection from each node's root account into other node's hacluster accounts (important since first access will prompt for host key confirmation)
4. Now that passwordless ssh works and hacluster can execute crm commands without password prompts execute the command crm report as root with the option -u hacluster

```
# crm report -u hacluster -f 08:00 my_cluster_report_on_AWS
```

5 Installing the SAP HANA Databases on both cluster nodes

Even though this document focuses on the integration of an already installed SAP HANA with already set up system replication into the pacemaker cluster, this chapter summarizes the test environment. Please always use the official documentation from SAP to install SAP HANA and to setup the system replication.

5.1 SAP HANA Installation

Preparation

- Read the SAP Installation and Setup Manuals available at the SAP Marketplace.
- Download the SAP HANA Software from SAP Marketplace.

Procedure 4.1: Installation and Checks

1. Install the SAP HANA Database as described in the SAP HANA Server Installation Guide.
2. Check if the SAP Host Agent is installed on all cluster nodes. If this SAP service is not installed, please install it now.
3. Verify that both databases are up and all processes of these databases are running correctly.

As Linux user <*sid*>adm use the command line tool **HDB** to get an overview of running HANA processes. The output of **HDB** info should show something like shown in the following screenshot:

```
suse02:~> HDB info
USER          PID  ...  COMMAND
ha1adm        6561 ...  -csh
ha1adm        6635 ...    \_ /bin/sh /usr/sap/HA1/HDB10/HDB info
ha1adm        6658 ...        \_ ps fx -U ha1 -o user,pid,ppid,pcpu,vsz,rss,args
ha1adm        5442 ...  sapstart pf=/hana/shared/HA1/profile/HA1_HDB10_suse02
ha1adm        5456 ...   \_ /usr/sap/HA1/HDB10/suse02/trace/hdb.sapHA1_HDB10 -d
-nw -f /usr/sap/HA1/HDB10/suse
ha1adm        5482 ...       \_ hdbnameserver
ha1adm        5551 ...       \_ hdbpreprocessor
ha1adm        5554 ...       \_ hdbcompileserver
ha1adm        5583 ...       \_ hdbindexserver
ha1adm        5586 ...       \_ hdbstatisticsserver
ha1adm        5589 ...       \_ hdbxsengine
ha1adm        5944 ...       \_ sapwebdisp_hdb
pf=/usr/sap/HA1/HDB10/suse02/wdisp/sapwebdisp.pfl -f /usr/sap/SL
ha1adm        5363 ...  /usr/sap/HA1/HDB10/exe/sapstartsrv
pf=/hana/shared/HA1/profile/HA1_HDB10_suse02 -D -u s
```

5.2 Post Installation Configuration

5.2.1 Back Up the Primary Database

Back up the primary database as described in the SAP HANA Administration Guide, Section *SAP HANA Database Backup and Recovery*. We provide an example with SQL Commands:

```
suse01:~ # hdbsql -u system -i 10 "BACKUP DATA USING FILE ('backup')"
```

If you have (for example) created a backup database user and a user key *hanabackup*, you can create an initial backup using the following command:

```
suse01:~ # hdbsql -U hanabackup "BACKUP DATA USING FILE ('backup')"
```

If the user key hanabackup has not been defined, please use an instance/user/ password combination for login.

Without a valid backup, you cannot bring SAP HANA into a system replication configuration.

5.2.2 HDB System Replication

For more information read the *Setting Up System Replication* section of the SAP HANA Administration Guide.

Procedure 4.2: Set up system replication on primary and secondary systems

1. Enable system replication on the primary system (**hdbnsutil** -sr_enable)
2. Register the secondary system with the primary system (**hdbnsutil** -sr_register)
3. Start the secondary system.

5.2.2.1 Enable Primary Node

As Linux user <*sid*>#adm enable the system replication at the primary node. You need to define a site name (like *WDF)*. This site name must be unique for all SAP HANA databases which are connected via system replication. This means the secondary must have a different site name.

```
suse01:~> hdbnsutil -sr_enable --name=WDF
checking local nameserver:
checking for active nameserver ...
nameserver is running, proceeding ...
configuring ini files ...
successfully enabled system as primary site ...
done.
```

5.2.2.2 Verify the State of System Replication

The command line tool hdbnsutil can be used to check the system replication mode and site name.

```
suse01:~> hdbnsutil -sr_state
checking for active or inactive nameserver ...
System Replication State
~~~~~~~~~~~~~~~~~~~~~~~~
mode: primary
site id: 1
site name: WDF
Host Mappings:
~~~~~~~~~~~~~~
done.
```

The mode has changed from “none” to “primary” and the site now has a site name and a site ID.

5.2.2.3 Enable the Secondary Node

The SAP HANA database instance on the secondary side must be stopped before the instance can be registered for the system replication. You can use your preferred method to stop the instance (like **HDB** or **sapcontrol**). After the database instance has been stopped successfully, you can register the instance using **hdbnsutil**. Again, use Linux user *<sid>*#adm:

```
suse02:~> HDB stop
...
suse02:~> hdbnsutil -sr_register --name=ROT \
     --remoteHost=suse01 --remoteInstance=10 \
     --replicationMode=sync --operationMode=logreplay
adding site ...
checking for inactive nameserver ...
nameserver suse02:30001 not responding.
collecting information ...
updating local ini files ...
done.
```

Now start the database instance again and verify the system replication status. On the secondary node, the mode should be one of „SYNC“, „SYNCMEM“ or „ASYNC“. The mode depends on the sync option defined during the registration of the client.

```
suse02:~> HDB start
...
suse02:~> hdbnsutil -sr_state
...
System Replication State
~~~~~~~~~~~~~~~~~~~~~~~~

mode: sync
site id: 2
site name: ROT
active primary site: 1
...
```

The remote Host is the primary node in our case, the parameter *remoteInstance* is the database instance number (here 10).

To view the replication state of the whole SAP HANA landscape use the following command as *<sid>*#adm user on the primary node.

```
suse01:~> HDBSettings.sh systemReplicationStatus.py
...
```

5.2.3 Manual re-establish SAP HANA SR to Original State

Bring the systems back to the original state:

1. Takeover HA1 to node 1
2. Wait till sync state is active
3. Stop HA1 on node 2
4. Re-register node 2 as secondary
5. Reconfigure global.ini
6. Start HA1 on node2

6 Configuration of the Cluster and SAP HANA Database Integration

This chapter describes the configuration of the cluster software SUSE Linux Enterprise High Availability Extension, which is part of the SUSE Linux Enterprise Server for SAP Applications, and SAP HANA Database Integration.

Procedure 5.1: Install and Configure the Cluster

1. Basic Cluster Configuration.
2. Configure Cluster Properties and Resources.

6.1 Installation

AWS "SLES for SAP" AMIs have already all HAE packages installed.

Update all packages to make sure that the latest revision of the AWS agents is installed.

```
suse01:~> zypper update
```

6.2 Basic Cluster Configuration

The first step is to set up the basic cluster framework.

6.2.1 Configuration of System Logging

SUSE recommends to use rsyslogd for logging in the SUSE cluster. This is a default configuration. Some AWS AMIs however use syslogd logging. Please perform the following commands as super user on all cluster nodes:

```
suse01:~> zypper install rsyslog
```

Depending on the installed packages a conflict may be shown, like the below example:

```
suse01:~ # zypper install rsyslog
Refreshing service 'SMT-http_smt-ec2_susecloud_net'.
Refreshing service 'cloud_update'.
Loading repository data...
Reading installed packages...
Resolving package dependencies...
Problem: syslog-ng-3.6.4-11.1.x86_64 conflicts with namespace:otherproviders(syslog)
provided by rsyslog-8.24.0-3.16.1.x86_64
 Solution 1: deinstallation of syslog-ng-3.6.4-11.1.x86_64
 Solution 2: do not install rsyslog-8.24.0-3.16.1.x86_64
Choose from above solutions by number or cancel [1/2/c] (c):
```

Select "Solution 1: deinstallation of syslog-ng", and then reboot both nodes.

6.2.2 Corosync Configuration

By default, the cluster service (pacemaker) is disabled and not set to start during boot, thus at this point the cluster should not be running. However, if you previsouly configured pacemaker and it is running proceed with a "stop" by using:

```
suse01:~ # systemctl stop pacemaker
```

The cluster service (pacemaker) status can be checked with:

```
suse01:~ # systemctl status pacemaker
```

6.2.2.1 Create Keys

On node-1, create a secret key used to encrypt all cluster communication:

```
suse01:~# corosync-keygen
```

A key new file will be created on /etc/corosync/authkey. This file needs to be copied to the same location on node-2. After generating and transferring the key file to the second node, verify that all permissions and ownerships on both nodes are the same:

```
suse01:~ # ls -l /etc/corosync/authkey
-r-------- 1 root root 128 Oct 23 10:51 /etc/corosync/authkey
```

6.2.3 Create the Corosync Configuration File

The configuration will have an IP address for node-1 which is supposed to be *ip-node-1*. node-2 has an IP address to which we refer as *ip-node-2*.

All cluster nodes are required to have a local configuration file "/etc/corosync/corosync.conf" which will be structured as follows. The relevant information is being located in the two sections describing interface and nodelist. The other entries can be configured as needed for a specific implementation. AWS requires a specific corosync configuration.

Note

Use the following configuration as an example for the file /etc/corosync/corosync.conf. Replace the IP addresses from the file below.

```
# Please read the corosync.conf.5 manual page
totem {
   version: 2
   token: 5000
   consensus: 7500
   token_retransmits_before_loss_const: 6
   secauth: on
   crypto_hash: sha1
   crypto_cipher: aes256
```

```
   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
      nodeid: 1
   }
   node {
      ring0_addr: ip-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
}
```

Replace the variables *ip-node-1*, *ip-node-2* and *ip-local-node* from the above sample file.

- **ip-local-node**: Use the IP address of the node where the file is being configured. This IP will be different between cluster nodes.
- **ip-node-1**: Add the IP address of cluster node node-1
- **ip-node-2**: Add the IP address of cluster node node-2

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 strong encryption of cluster communication is desired.

Note

Important: Change the password of the user hacluster

6.2.4 Check the Cluster for the first Time

Now it's time to check and optionally start the cluster for the first time on both nodes.

```
suse01:~ # systemctl status pacemaker
suse02:~ # systemctl status pacemaker
suse01:~ # systemctl start pacemaker
suse02:~ # systemctl start pacemaker
```

Check the cluster status with crm_mon. We use the option "-r" to also see resources, which are configured but stopped.

This is a manual start. We do not recommend to automatically rejoin a cluster node after a system crash with a reboot. We recommend a full inspection and a root cause analysis of the crash before rejoining the cluster.

```
# crm_mon -r
```

The command will show the "empty" cluster and will print something like the following screen output. The most interesting information for now is that there are two nodes in the status "online" and the message "partition with quorum".

```
Stack: corosync
Current DC: suse01 (version 1.1.16-6.5.1-77ea74d) - partition with quorum
Last updated: Wed Mar 21 15:19:34 2018
Last change: Wed Mar 14 13:00:59 2018 by root via cibadmin on suse01

2 nodes configured
0 resources configured

Online: [ suse01 suse02 ]
```

```
No resources
```

Checking the Configuration The configuration can be checked with the command:

```
corosync-cfgtool -s
```

It'll create a result like the following one for a cluster node with the IP address 10.79.8.23:

```
Printing ring status.
Local node ID 1
RING ID 0
 id = 10.79.8.23
 status = ring 0 active with no faults
```

The cluster in question has been using ring 0, the node had the ID 1.

Note

This is a manual start of the cluster. We do not recommend to automatically rejoin a cluster node after a system crash with a reboot. A full inspection and a root cause analysis of the crash is highly recommended before rejoining the cluster.

6.3 Configure Cluster Properties and Resources

This section describes how to configure constraints, resources, bootstrap and STONITH using the crm configure shell command as described in section *Configuring and Managing Cluster Resources (Command Line)*, ↑SUSE Linux Enterprise High Availability Extension.

Use the command **crm** to add the objects to CRM. Copy the following examples to a local file, edit the file and than load the configuration to the CIB:

```
suse01:~ # vi crm-fileXX
suse01:~ # crm configure load update crm-fileXX
```

6.3.1 Cluster Bootstrap and More

The first example defines the cluster bootstrap options, the resource and operation defaults.

```
suse01:~ # vi crm-bs.txt
```

```
# enter the following to the file crm-bs.txt
property $id="cib-bootstrap-options" \
   stonith-enabled="true" \
   stonith-action="poweroff" \
   stonith-timeout="150s"
rsc_defaults $id="rsc-options" \
   resource-stickiness="1000" \
   migration-threshold="5000"
op_defaults $id="op-options" \
   timeout="600"
```

The setting *poweroff* forces the agents to shutdown the instance. This is desirable in order to avoid split brain scenarios on the AWS platform.

Now we add the configuration to the cluster.

```
suse01:~ # crm configure load update crm-bs.txt
```

6.3.2 STONITH Device

The next configuration part defines a AWS specific STONITH resource.

```
suse01:~ # vi aws-stonith.txt
# enter the following to the file aws-stonith.txt
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=120 timeout=60 \
    meta target-role=Started \
    params tag=pacemaker profile=cluster
```

The "tag = pacemaker" entry needs to match the tag chosen for the EC2 instances. The value for this tag will contain the host name. The name of the profile ("cluster" in this example) will have to match the previously configured AWS profile.

Name this file for example aws-stonith.txt and add this file to the configuration. The following command has to be issued as super user. It uses the file name aws-stonith.txt:

```
suse01:~ # crm configure load update aws-stonith.txt
```

A working STONITH method is mandatory in order to run a supported SUSE cluster on AWS.

Note

Make sure to execute the STONITH tests as outlined in section *Troubleshooting* of this document to verify STONITH on both nodes. Checking the configuration for potential problems now will increase the chances to launch the cluster successfully.

6.3.3 Creation of a Move IP Resource

This step requires the overlay IP address and the name of the AWS routing table. Create a file with the following content:

```
suse01:~ # vi aws-move-ip.txt
# enter the following to the file aws-move-ip.txt
Create a file with the following content:
primitive res_AWS_IP ocf:suse:aws-vpc-move-ip \
   params ip=overlay-ip-address routing_table=rtb-table interface=eth0 profile=cluster \
   op start interval=0 timeout=180 \
   op stop interval=0 timeout=180 \
   op monitor interval=60 timeout=60
```

Replace the string *overlay-ip-address* by the overlay IP address. Replace *rtb-table* by the name of the AWS routing table. The name of the profile (cluster in this example) will have to match the previously configured AWS profile. Load this file into the cluster configuration by issuing the following command as super user:

```
suse01:~ # crm configure load update aws-move-ip.txt
```

Note

Make sure to execute the Move IP tests as outlined in section *Troubleshooting* of this document to verify them on both nodes. Checking the configuration for potential problems at current point in time will increase the chances to launch the cluster successfully.

6.3.4 SAPHanaTopology

Next we define the group of resources needed, before the HANA database instances can be started. Prepare the changes in a text file, for example crm-saphanatop.txt, and load these with the command: **crm configure load update crm-saphanatop.txt**

```
# vi crm-saphanatop.txt
# enter the following to the file crm-saphanatop.txt
primitive rsc_SAPHanaTopology_HA1_HDB10 ocf:suse:SAPHanaTopology \
        operations $id="rsc_sap2_HA1_HDB10-operations" \
        op monitor interval="10" timeout="600" \
        op start interval="0" timeout="600" \
        op stop interval="0" timeout="300" \
        params SID="HA1" InstanceNumber="10"
clone cln_SAPHanaTopology_HA1_HDB10 rsc_SAPHanaTopology_HA1_HDB10 \
        meta clone-node-max="1" interleave="true"
```

The most important parameters here are *SID* and *InstanceNumber*, which are in the SAP context quite self explaining. Beside these parameters, the timeout values or the operations (start, monitor, stop) are typical tuneables.

Replace the parameters *SID* and *InstanceNumber* with the respective information from your environment.

Again we add the configuration to the cluster:

```
suse01:~ # crm configure load update crm-saphanatop.txt
```

Note

Additional information about all parameters could be found with the command **man ocf_suse_SAPHanaTopology**

6.3.5 SAPHana

Next we define the group of resources needed, before the HANA database instances can be started. Edit the changes in a text file, for example crm-saphana.txt and load these with the command: crm configure load update crm-saphana.txt

TABLE 2: TYPICAL RESOURCE AGENT PARAMETER SETTINGS FOR DIFFERENT SCENARIOS

Parameter	Performance Optimized	Cost Optimized	Multi-Tier
PREFER_SITE_TAKEOVER	true	false	false / true
AUTOMATED_REGISTER	false / true	false / true	false
DUPLICATE_PRIMARY_TIMEOUT	7200	7200	7200

TABLE 3: DESCRIPTION OF IMPORTANT RESOURCE AGENT PARAMETERS

Parameter	Description
PREFER_SITE_TAKEOVER	Defines whether RA should prefer to takeover to the secondary instance instead of restarting the failed primary locally.
AUTOMATED_REGISTER	Defines whether a former primary should be automatically registered to be secondary of the new primary. With this parameter you can adapt the level of system replication automation. If set to false the former primary must be manually registered. The cluster will not start this SAP HANA RDBMS till it is registered to avoid double primary up situations.
DUPLICATE_PRIMARY_TIMEOUT	Time difference needed between two primary time stamps, if a dual-primary situation occurs. If the time difference is less than the time gap, than the cluster hold one or both instances in a "WAITING" status. This is to give a admin the chance to react on a failover. If the complete node of the former primary crashed, the former primary will be registered after the time difference is passed. If "only" the SAP HANA RDBMS has crashed, then the former primary will be registered immediately. After this registration to the new primary all data will be overwritten by the system replication.

Additional information about all parameters could be found with the command **man** ocf_suse_SAPHana

```
# vi crm-saphana.txt
# enter the following to the file crm-saphana.txt
primitive rsc_SAPHana_HA1_HDB10 ocf:suse:SAPHana \
        operations $id="rsc_sap_HA1_HDB10-operations" \
        op start interval="0" timeout="3600" \
        op stop interval="0" timeout="3600" \
        op promote interval="0" timeout="3600" \
        op monitor interval="60" role="Master" timeout="700" \
        op monitor interval="61" role="Slave" timeout="700" \
        params SID="HA1" InstanceNumber="10" PREFER_SITE_TAKEOVER="true" \
        DUPLICATE_PRIMARY_TIMEOUT="7200" AUTOMATED_REGISTER="false"
ms msl_SAPHana_HA1_HDB10 rsc_SAPHana_HA1_HDB10 \
        meta clone-max="2" clone-node-max="1" interleave="true"
```

We add the configuration to the cluster:

```
suse01:~ # crm configure load update crm-saphana.txt
```

The most important parameters here are again *SID* and *InstanceNumber*. Beside these parameters the timeout values for the operations (start, promote, monitors, stop) are typical tuneables.

Note

Additional information about all parameters could be found with the command: **man ocf_suse_SAPHana**.

6.3.6 Constraints

Two constraints are organizing the correct placement of the Overlay IP address for the client database access and the start order between the two resource agents SAPHana and SAPHanaTopology.

The AWS Move IP agent will need to operate on the same node as the SAP HANA database. A constraint will force it to be on the same node.

```
suse01:~ # vi crm-cs.txt
```

Enter the following to the file crm-cs.txt

```
colocation col_saphana_ip_HA1_HDB10 2000: res_AWS_IP:Started \
    msl_SAPHana_HA1_HDB10:Master
order ord_SAPHana_HA1_HDB10 Optional: cln_SAPHanaTopology_HA1_HDB10 \
    msl_SAPHana_HA1_HDB10
```

Add this file to the configuration. The following command has to be issued as super user. It uses the file name crm-cs.txt:

```
suse01:~ # crm configure load update crm-cs.txt
```

6.3.7 Cluster Status After Configuration

Now that the cluster has been configured, it should have 2 (two) online nodes, and 6 (six) resources.

Cluster status can be checked with crm status command:

```
suse01:~ # crm status
Stack: corosync
Current DC: suse01 (version 1.1.16-6.5.1-77ea74d) - partition with quorum
Last updated: Tue Oct 23 13:20:46 2018
Last change: Tue Oct 23 13:19:52 2018 by root via crm_attribute on suse01

2 nodes configured
6 resources configured

Online: [ suse01 suse02 ]

Full list of resources:

 res_AWS_STONITH (stonith:external/ec2): Started suse01
 res_AWS_IP (ocf::suse:aws-vpc-move-ip): Started suse01
 Clone Set: cln_SAPHanaTopology_HA1_HDB10 [rsc_SAPHanaTopology_HA1_HDB10]
  Started: [ suse01 suse02 ]
 Master/Slave Set: msl_SAPHana_HA1_HDB10 [rsc_SAPHana_HA1_HDB10]
  Masters: [ suse01 ]
  Slaves: [ suse02 ]
```

The Above example shows that the Overlay IP resource (res_AWS_IP) is "Started" on node suse01, along with SAPHanaTopology resource (cln_SAPHanaTopology_HA1_HDB10) running on both cluster nodes, and Master/Slave SAPHana (msl_SAPHana_HA1_HDB10), which on the above example is Master (Primary) on node suse01, and Secondary on node suse02.

7 Testing the Cluster

The lists of tests will be improved in the next update of this document.

As with any cluster testing is crucial. Please make sure that all test cases derived from customer expectations are implemented and passed fully. Else the project is likely to fail in production use.

The test prerequisite, if not described differently, is always that both nodes are booted, normal members of the cluster and the HANA RDBMS is running. The system replication is in sync (SOK).

7.1 Test Cases for Semi Automation

In the following test descriptions we assume **PREFER_SITE_TAKEOVER** = "**true**" and **AUTOMATED_REGISTER** = "**false**".

Note

The following tests are designed to be run in sequence and depend on the exit state of the preceding tests.

7.1.1 Test 1: Stop Primary Database on Node 1

Component: Primary Database

Description: The primary HANA database is stopped during normal cluster operation.

Procedure 6.1: Test procedure

- Stop the primary HANA database gracefully as <*sid*>adm.

```
suse01# HDB stop
```

Procedure 6.2: Recovery procedure

- Manually register the old primary (on node 1) with the new primary after takeover (on node 2) as *sid*adm.

```
suse01# hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --
replicationMode=sync  --name=WDF
```

Restart the HANA database (now secondary) on node 1 as root.

```
suse01# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse01
```

Expected:

1. The cluster detects the stopped primary HANA database (on node 1) and marks the resource failed.
2. The cluster promotes the secondary HANA database (on node 2) to take over as primary.
3. The cluster migrates the IP address to the new primary (on node 2).
4. After some time the cluster shows the sync_state of the stopped primary (on node 1) as SFAIL.
5. Because AUTOMATED_REGISTER = "false" the cluster does not restart the failed HANA database or register it against the new primary.
6. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
7. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.2 Test 2: Stop Primary Database on Node 2

Component: Primary Database

Description: The primary HANA database is stopped during normal cluster operation.

Procedure 6.3: Test procedure

- Stop the database gracefully as <*sid*>adm.

```
suse02# HDB stop
```

Procedure 6.4: Recovery procedure

- Manually register the old primary (on node 2) with the new primary after takeover (on node 1) as <*sid*>adm.

```
suse02# hdbnsutil -sr_register --remoteHost=suse01 --remoteInstance=10 --
replicationMode=sync  --name=ROT
```

Restart the HANA database (now secondary) on node 1 as root.

```
suse02# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse02
```

Expected:

1. The cluster detects the stopped primary HANA database (on node 2) and marks the resource failed.
2. The cluster promotes the secondary HANA database (on node 1) to take over as primary.
3. The cluster migrates the IP address to the new primary (on node 1).
4. After some time the cluster shows the sync_state of the stopped primary (on node 2) as SFAIL.
5. Because AUTOMATED_REGISTER = "false" the cluster does not restart the failed HANA database or register it against the new primary.
6. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
7. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.3 Test 3: Crash Primary Database on Node 1

Component: Primary Database

Description: Simulate a complete break-down of the primary database system.

Procedure 6.5: Test procedure

- Kill the primary database system using signals as <*sid*>adm.

```
suse01# HDB kill-9
```

Procedure 6.6: Recovery procedure

- Manually register the old primary (on node 1) with the new primary after takeover (on node 2) as <*sid*>adm.

```
suse01# hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --
replicationMode=sync  --name=WDF
```

Restart the HANA database (now secondary) on node 1 as root.

```
suse01# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse01
```

Expected:

1. The cluster detects the stopped primary HANA database (on node 1) and marks the resource failed.
2. The cluster promotes the secondary HANA database (on node 2) to take over as primary.
3. The cluster migrates the IP address to the new primary (on node 2).
4. After some time the cluster shows the sync_state of the stopped primary (on node 1) as SFAIL.
5. Because AUTOMATED_REGISTER = "false" the cluster does not restart the failed HANA database or register it against the new primary.
6. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
7. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.4 Test 4: Crash Primary Database on Node 2

Component: Primary Database

Description: Simulate a complete break-down of the primary database system.

Procedure 6.7: Test procedure

- Kill the primary database system using signals as <*sid*>adm.

```
suse02# HDB kill-9
```

Procedure 6.8: Recovery procedure

- Manually register the old primary (on node 2) with the new primary after takeover (on node 1) as <*sid*>adm.

```
suse02# hdbnsutil -sr_register --remoteHost=suse01 --remoteInstance=10 --
replicationMode=sync  --name=ROT
```

Restart the HANA database (now secondary) on node 1 as root.

```
suse02# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse02
```

Expected:

1. The cluster detects the stopped primary HANA database (on node 2) and marks the resource failed.
2. The cluster promotes the secondary HANA database (on node 1) to take over as primary.
3. The cluster migrates the IP address to the new primary (on node 1).
4. After some time the cluster shows the sync_state of the stopped primary (on node 2) as SFAIL.
5. Because AUTOMATED_REGISTER="false" the cluster does not restart the failed HANA database or register it against the new primary.
6. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
7. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.5 Test 5: Crash Primary Site Node (node 1)

Component: Cluster node of primary site

Description: Simulate a crash of the primary site node running the primary HANA database.

Procedure 6.9: Test procedure

- Crash the primary node by sending a 'fast-reboot' system request.

```
suse01# echo 'b' > /proc/sysrq-trigger
```

Procedure 6.10: Recovery procedure

1. AWS infrastructure has stopped the fenced instance. Restart it with AWS console or AWS CLI tools. Execute the following command after the instance has booted.

```
suse01# systemctl start pacemaker
```

2. Manually register the old primary (on node 1) with the new primary after takeover (on node 2) as <*sid*>adm.

```
suse01# hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --
replicationMode=sync  --name=WDF
```

Restart the HANA database (now secondary) on node 1 as root.

```
suse01# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse01
```

Expected:

1. The cluster detects the failed node (node 1) and declares it UNCLEAN and sets the secondary node (node 2) to status "partition WITHOUT quorum".
2. The cluster fences the failed node (node 1). The AWS infrastructure stops the instance.
3. The cluster declares the failed node (node 1) OFFLINE.
4. The cluster promotes the secondary HANA database (on node 2) to take over as primary.
5. The cluster migrates the IP address to the new primary (on node 2).
6. After some time the cluster shows the sync_state of the stopped primary (on node 2) as SFAIL.
7. Because AUTOMATED_REGISTER = "false" the cluster does not restart the failed HANA database or register it against the new primary.
8. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
9. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.6 Test 6: Crash Secondary Site Node (node 2)

Component: Cluster node of secondary site

Description: Simulate a crash of the secondary site node running the primary HANA database.

Procedure 6.11: Test procedure

- Crash the secondary node by sending a 'fast-reboot' system request.

```
suse02# echo 'b' > /proc/sysrq-trigger
```

Procedure 6.12: Recovery procedure

1. AWS infrastructure has stopped the fenced instance. Restart it with AWS console or AWS CLI tools. Execute the following command after the instance has booted.

   ```
   suse02# systemctl start pacemaker
   ```

2. Manually register the old primary (on node 2) with the new primary after takeover (on node 1) as <*sid*>adm.

```
suse02# hdbnsutil -sr_register --remoteHost=suse01 --remoteInstance=10 --
replicationMode=sync  --name=ROT
```

Restart the HANA database (now secondary) on node 2 as root.

```
suse02# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse02
```

Expected:

1. The cluster detects the failed secondary node (node 2) and declares it UNCLEAN and sets the primary node (node 1) to status "partition WITHOUT quorum".
2. The cluster fences the failed secondary node (node 2). The AWS infrastructure stops the instance.
3. The cluster declares the failed secondary node (node 2) OFFLINE.
4. The cluster promotes the secondary HANA database (on node 1) to take over as primary.
5. The cluster migrates the IP address to the new primary (on node 1).
6. After some time the cluster shows the sync_state of the stopped secondary (on node 2) as SFAIL.
7. Because AUTOMATED_REGISTER = "false" the cluster does not restart the failed HANA database or register it against the new primary.

8. After the manual register and resource cleanup the system replication pair is marked as in sync (SOK).
9. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.7 Test 7: Stop the Secondary Database on Node 2

Component: Secondary HANA database

Description: The secondary HANA database is stopped during normal cluster operation.

Procedure 6.13: Test procedure

- Stop the secondary HANA database gracefully as <*sid*>adm.

```
suse02# HDB stop
```

Procedure 6.14: Recovery procedure

- Cleanup the failed resource status of the secondary HANA database (on node 2) as root.

```
suse02# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse02
```

Expected:

1. The cluster detects the stopped secondary database (on node 2) and marks the resource failed.
2. The cluster detects the broken system replication and marks it as failed (SFAIL).
3. The cluster restarts the secondary HANA database on the same node (node 2).
4. The cluster detects that the system replication is in sync again and marks it as ok (SOK).
5. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.8 Test 8: Crash the Secondary Database on Node 2

Component: Secondary HANA database

Description: Simulate a complete break-down of the secondary database system.

Procedure 6.15: Test procedure

- Kill the secondary database system using signals as *<sid>*adm.

```
suse02# HDB kill-9
```

Procedure 6.16: Recovery procedure

- Cleanup the failed resource status of the secondary HANA database (on node 2) as root.

```
suse02# crm resource cleanup rsc_SAPHana_HA1_HDB10 suse02
```

Expected:

1. The cluster detects the stopped secondary database (on node 2) and marks the resource failed.
2. The cluster detects the broken system replication and marks it as failed (SFAIL).
3. The cluster restarts the secondary HANA database on the same node (node 2).
4. The cluster detects that the system replication is in sync again and marks it as ok (SOK).
5. The cluster "failed actions" are cleaned up after following the recovery procedure.

7.1.9 Test 9: Crash Secondary Site Node (node 2) running Secondary HANA Database

Component: Cluster node of secondary site

Description: Simulate a crash of the secondary site node running the secondary HANA database.

Procedure 6.17: Test procedure

- Crash the secondary node by sending a 'fast-reboot' system request.

```
suse02# echo 'b' > /proc/sysrq-trigger
```

Procedure 6.18: Recovery procedure

1. AWS infrastructure has stopped the fenced instance. Restart it with AWS console or AWS CLI tools. Execute the following command after the instance has booted.

```
suse02# systemctl start pacemaker
```

Expected:

1. The cluster detects the failed secondary node (node 2) and declares it UNCLEAN and sets the primary node (node 1) to status "partition WITHOUT quorum".
2. The cluster fences the failed secondary node (node 2). The AWS infrastructure stops the instance.
3. The cluster declares the failed secondary node (node 2) OFFLINE.
4. After some time the cluster shows the sync_state of the stopped secondary (on node 2) as SFAIL.
5. Once the fenced node (node 2) rejoins the cluster the former secondary HANA database is started automatically.
6. The cluster detects that the system replication is in sync again and marks it as ok (SOK).

7.1.10 Test 10 : Test Failure of Replication LAN

Component: Replication LAN

Description: This test is not applicable to AWS. There is no separate replication LAN.

8 Administration

8.1 Do and Don’t

In your project, you should:

- Define STONITH before adding other resources to the cluster
- Do intensive testing.
- Tune the timeouts of operations of SAPHana and SAPHanaTopology.
- Start with PREFER_SITE_TAKEOVER = ”true”, AUTOMATED_REGISTER = ”false” and DUPLICATE_PRIMARY_TIMEOUT = ”7200”.

In your project, avoid:

- Rapidly changing/changing back cluster configuration, such as: Setting nodes to standby and online again or stopping/starting the master/slave resource.
- Creating a cluster without proper time synchronization or unstable name resolutions for hosts, users and groups. Verify that you use ntp for time synchronization. Be consistent with your naming and the usage of domain names.
- Adding location rules for the clone, master/slave or IP resource. Only location rules mentioned in this setup guide are allowed.
- As "migrating" or "moving" resources in crm-shell, HAWK or other tools would add client-prefer location rules this activities are completely forbidden.

8.2 Monitoring and Tools

You can use the High Availability Web Konsole (HAWK), SAP HANA Studio and different command line tools for cluster status requests.

8.2.1 HAWK – Cluster Status and more

You can use an internet browser to check the cluster status.

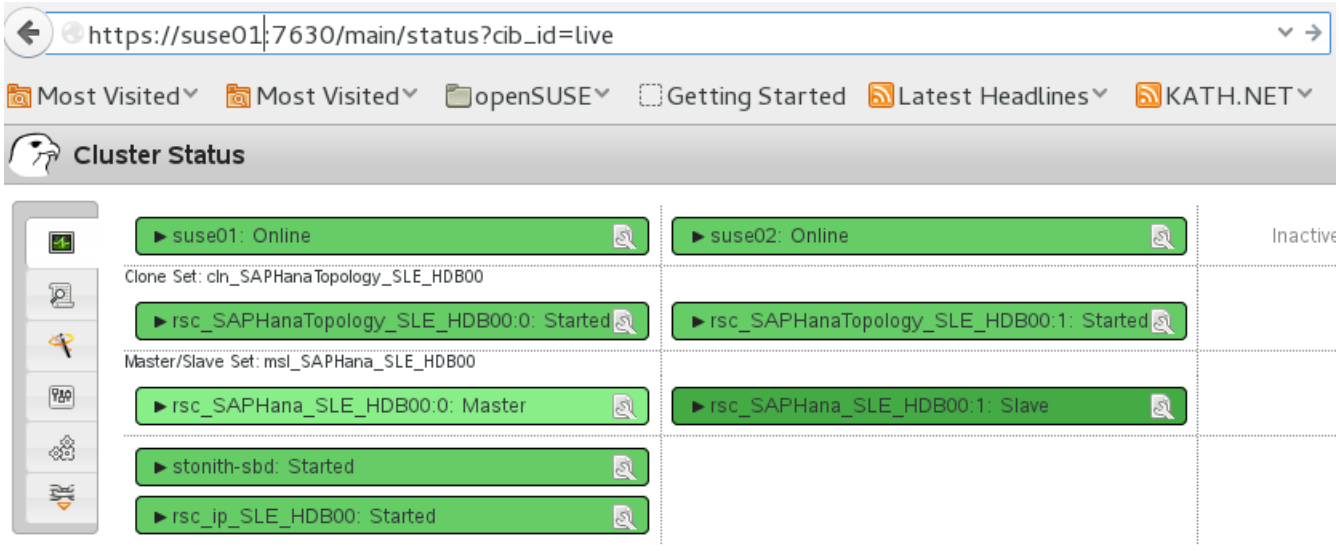

FIGURE 5: CLUSTER STATUS IN HAWK

If you set up the cluster using ha-cluster-init and you have installed all packages as described above, your system will provide a very useful web interface. You can use this graphical web interface to get an overview of the complete cluster status, doing administrative tasks or even configure resources and cluster bootstrap parameters. Read our product manuals for a complete documentation of this powerful user interface.

8.2.2 SAP HANA Studio

Database-specific administration and checks can be done with SAP HANA studio.

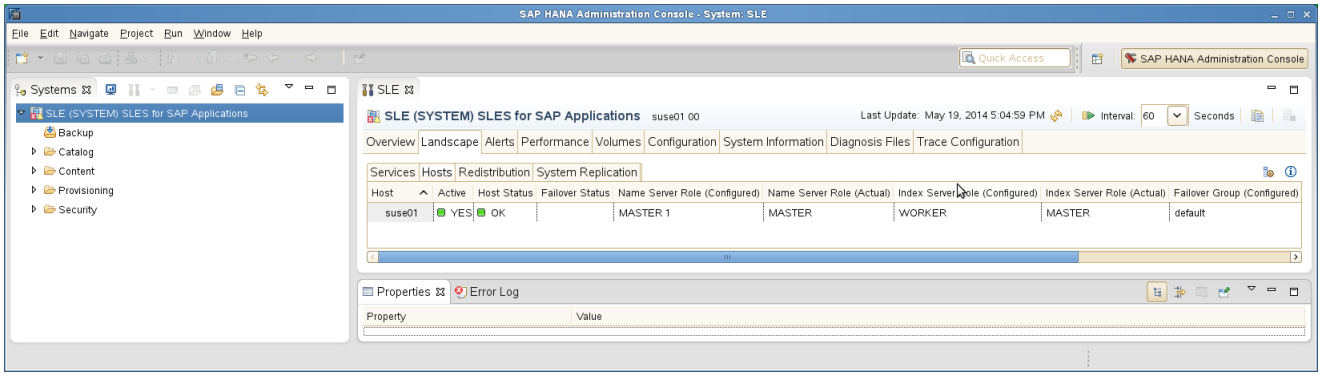

FIGURE 6: SAP HANA STUDIO – LANDSCAPE

8.2.3 Cluster Command-Line Tools

A simple overview can be obtained by calling **crm_mon**. Using option -r shows also stopped but already configured resources. Option -1 tells **crm_mon** to output the status once instead of periodically.

```
suse01:~ # crm_mon -1
Stack: corosync
Current DC: suse01 (version 1.1.16-6.5.1-77ea74d) - partition with quorum
Last updated: Tue Oct 23 15:24:45 2018
Last change: Tue Oct 23 15:23:56 2018 by root via crm_attribute on suse01

2 nodes configured
6 resources configured

Online: [ suse01 suse02 ]

Active resources:
```

```
 res_AWS_STONITH (stonith:external/ec2): Started suse01
 res_AWS_IP (ocf::suse:aws-vpc-move-ip): Started suse01
Clone Set: cln_SAPHanaTopology_HA1_HDB10 [rsc_SAPHanaTopology_HA1_HDB10]
 Started: [ suse01 suse02 ]
Master/Slave Set: msl_SAPHana_HA1_HDB10 [rsc_SAPHana_HA1_HDB10]
 Masters: [ suse01 ]
 Slaves: [ suse02 ]
```

See the manual page crm_mon(8) for details.

To show some of the SAPHana, SAPHanaTopology resource agent internal values, you can call the program **SAPHanaSR-showAttr**. The internal values, storage place and their parameter names may change in the next versions. The command **SAPHanaSR-showAttr** will always fetch the values from the correct storage place.

Do not use cluster commands like crm_attribute to fetch the values directly from the cluster, because in such cases your methods will be broken, when we need to move an attribute to a different storage place or even out of the cluster. For first **SAPHanaSR-showAttr** is a test program only and should not be used for automated system monitoring.

```
suse01:~ # SAPHanaSR-showAttr
Host \ Attr clone_state remoteHost roles       ... site    srmode sync_state ...
-------------------------------------------------------------------------------
suse01      PROMOTED    suse02     4:P:master1:... WDF      sync  PRIM       ...
suse02      DEMOTED     suse01     4:S:master1:... ROT      sync  SOK        ...
```

8.2.4 SAP HANA LandscapeHostConfiguration

Check the status of a SAPHana database and figure out if the cluster should react, you can use the script landscapeHostConfiguration be called as Linux user <*sid*>adm.

```
suse01:~> HDBSettings.sh landscapeHostConfiguration.py
| Host   | Host   | ... NameServer   | NameServer  | IndexServer | IndexServer |
|        | Active | ... Config Role  | Actual Role | Config Role | Actual Role |
| ------ | ------ | ... ------------ | ----------- | ----------- | ----------- |
| suse01 | yes    | ... master 1     | master      | worker      | master      |

overall host status: ok
```

Following the SAP HA guideline, the SAPHana resource agent interprets the return codes in the following way:

TABLE 4: INTERPRETATION OF RETURN CODES

Return Code	Interpretation
4	SAP HANA database is up and OK. The cluster does interpret this as a correctly running database
3	SAP HANA database is up and in status info. The cluster does interpret this as a correctly running database.
2	SAP HANA database is up and in status warning. The cluster does interpret this as a correctly running database.
1	SAP HANA database is down. If the database should be up and is not down by intention, this could trigger a takeover.
0	Internal Script Error – to be ignored.

8.3 Maintenance

It is highly recommended to register your systems to either a local SUSE Manager or SMT or remote with SUSE Customer Center to be able to receive updates to the OS or HAE.

8.3.1 Reconfiguration the Cluster after a Takeover

The nodes of the HAE Cluster will monitor each other. They will shut down unresponsive or misbehaving nodes prior to any failover actions in order to prevent data corruption. Setting the AWS stonith-action to poweroff will permanently shut down the defect cluster node. This will expedite a takeover on AWS.

The default setting reboot will make the STONITH agent wait until a reboot will have been successfully completed. This will delay the reconfiguration of the SAP HANA database. Re-integrating a faulty cluster node into the cluster has to be performed manually since it'll take an investigation why the cluster node didn't operate as expected.

Restarting the second (faulty) cluster node automatically can be configured as well. It bears however the risk that the remaining node gets harmed trough an incorrect acting second (faulty) node. The reconfiguration of the second (faulty) node happens through the following steps:

1. Restart node through the AWS console
2. Investigate the node after reboot and fix a potential defect
3. Boot SAP HANA manually. Check the instance health. Fix a potential defect. Shut down SAP HANA.
4. Configure SAP HANA to be a secondary node to the new master node.
5. Start SAP HANA as secondary node
6. Restart the HAE cluster with the command "systemctl start pacemaker" as super user. This process can take several minutes.
7. Verify that all cluster services operate correctly. A takeover is now completed. The roles of the two cluster nodes have been flipped. The SAP HANA database is now protected against future failure events.

8.3.2 Updating the OS and Cluster

Update of SUSE Linux Enterprise Server for SAP Applications packages including cluster software you should follow the rolling update procedure defined in the product documentation of SUSE Linux Enterprise High Availability *Upgrading Your Cluster and Updating Software Packages*, ↑ High Availability Administration Guide.

8.3.3 Updating SAP HANA

For updating SAP HANA database systems in system replication you need to follow the defined SAP processes. This section describes the steps to be done before and after the update procedure to get the system replication automated again.

Procedure 7.1: Top level steps to updating SAP HANA in the cluster

1. Prepare the cluster not to react on the maintenance work to be done on the SAP HANA database systems. Set the master/slave resource to be unmanaged and the cluster nodes in maintenance mode.

- For the master/slave resource set the unmanage status: **crm resource unmanage** master-slave-resource
- For all cluster nodes set the ready status: **crm node maintenance** *node*.

2. Complete the SAP Update process for both SAP HANA database systems. This process is described by SAP.
3. After the SAP HANA update is complete on both sites, tell the cluster about the end of the maintenance process.
   - For all cluster nodes set the ready status: **crm node ready** *node*.
4. As we expect the primary/secondary roles to be exchanged after the maintenance, tell the cluster to forget about this states and to reprobe the updated SAP HANA database systems. **crm resource cleanup** *master-slave-resource*.
5. In the last step we tell the cluster to manage the SAP HANA database systems again. **crm resource manage** *master-slave-resource*.

8.3.4 Migrating a HANA Primary

In the following procedures we assume the primary to be running on node1 and the secondary on node2. The goal is to "exchange" the roles of the nodes, so finally the primary should run on node2 and the secondary should run on node1.

There are different methods to get the exchange of the roles done, the following procedure shows how to tell the cluster to "accept" a role change done with native HANA commands.

Procedure 7.2: Migrating a HANA primary with unmanaged master/slave resource

1. Set the master/slave resource to be unmanaged. This could be done on any cluster node.
   - **crm resource unmanage** *master-slave-resource-name*
2. Stop the primary SAP HANA database system. Enter the command in our example on node1 as user <*sid*>#adm.
   - **HDB stop**
3. Start the takeover process on the secondary SAP HANA database system. Enter the command in our example on node2 as user <*sid*>#adm.

- **hdbnsutil -sr_takeover**

4. Register the former primary to become the new secondary. Enter the command in our example on node1 as user <*sid*>#adm.

   - **hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --replicationMode=sync --name=WDF --operationMode=logreplay**

5. Start the new secondary SAP HANA database system. Enter the command in our example on node1 as user <*sid*>#adm.

   - **HDB start**

   ```
   Wait some time till SAPHanaSR-showAttr shows both SAP HANA database systems
   to be up again (field roles must start with the digit 4).
   ```

6. Tell the cluster to forget about the former master/slave roles and to re-monitor the failed master. The command could be submitted on any cluster node as user root.

   - **crm resource cleanup** *master-slave-resource-name*

7. Set the master/slave resource to the status managed again. The command could be submitted on any cluster node as user root.

   - **crm resource manage** *master-slave-resource-name*

Now we explain how to use the cluster to partially automate the migration.

Procedure 7.3: Migrating a HANA primary using cluster migration commands

1. Create a migrate away from this node rule.

- **crm resource migrate** *sapHanaResource* **force**
- Because of we used the migrate away rule the cluster will stop the current primary and run a promote on the secondary site, if the system replication was in sync before. You should not migrate the primary, if the status of the system replication is not in sync (SFAIL).
- Wait till the secondary has completely taken over to be the new primary.

2. If you have setup AUTOMATED_REGISTER = "true" you can skip this step. You now need to register the old primary in other cases. Enter the command in our example on node1 as user <*sid*>adm.
   - **hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --replicationMode=sync --name=WDF --operationMode=logreplay**
3. Unmigrate the resource to allow the cluster to start the new secondary.
   - **crm resource unmigrate** *sapHanaRsource*

Procedure 7.4: Migrating a HANA primary using cluster node status standby

1. Set the primary node to be standby.
   - **crm node standby** *suse01*
   - The cluster will stop the primary SAP HANA database and if the system replication was in sync process the takeover on the secondary site
   - Wait till the former secondary has completely taken over to be the new primary.
2. If you have setup AUTOMATED_REGISTER = "true" you can skip this step. You now need to register the old primary in other cases. Enter the command in our example on node1 as user <*sid*>adm.
   - **hdbnsutil -sr_register --remoteHost=suse02 --remoteInstance=10 --replicationMode=sync --name=WDF --operationMode=logreplay**
3. Set the standby node to be online again.

- **crm node online** *suse01*

8.4 Appendix: Troubleshooting

8.4.1 Verification and debugging of the aws-vpc-move-ip Cluster Agent

8.4.1.1 Appendix: Start the Overlay IP Address to be hosted on a given Node

As root user run the following command using the same parameters as in your cluster configuration:

```
# OCF_RESKEY_address=<virtual_IPv4_address> OCF_RESKEY_routing_table=<AWS_route_table>
 OCF_RESKEY_interface=eth0 OCF_RESKEY_profile=<AWS-profile> OCF_ROOT=/usr/lib/ocf /usr/
lib/ocf/resource.d/suse/aws-vpc-move-ip monitor
```

Check the console output (DEBUG keyword) for error messages

8.4.1.2 Appendix: Stop the Overlay IP Address to be hosted on a given Node

As root user run the following command using the same parameters as in your cluster configuration:

```
# OCF_RESKEY_address=<virtual_IPv4_address> OCF_RESKEY_routing_table=<AWS_route_table>
 OCF_RESKEY_interface=eth0 OCF_RESKEY_profile=<AWS-profile> OCF_ROOT=/usr/lib/ocf /usr/
lib/ocf/resource.d/suse/aws-vpc-move-ip stop
```

Check DEBUG output for errors and verify that the virtual IP address is NOT active on the current node with the command *ip address list dev eth0*. Start the overlay IP Address to be hosted on a given Node. As root user run the following command using the same parameters as in your cluster configuration:

```
# OCF_RESKEY_address=<virtual_IPv4_address> OCF_RESKEY_routing_table=<AWS_route_table>
 OCF_RESKEY_interface=eth0 OCF_RESKEY_profile=<AWS-profile> OCF_ROOT=/usr/lib/ocf /usr/
lib/ocf/resource.d/suse/aws-vpc-move-ip start
```

Check DEBUG output for error messages and verify that the virtual IP address is active on the current node with the command ip a.

8.4.2 Testing the AWS STONITH Agent

The STONITH agent will shutdown the other node if he thinks that this node isn't anymore reachable. The agent can be called manually as super user on a cluster node 1 to shut down cluster node 2. Use it with the same parameter as being used in the Stonith agent configuration:

```
# stonith -t external/ec2 profile=<AWS-profile> port=<cluster-node2>
 tag=<aws_tag_containing_hostname> -T off <cluster-node2>
```

This command will shutdown cluster node 2. Check the errors reported during execution of the command if it's not going to work as planned. Re-start cluster node 2 and test STONITH the other way around. The parameter used here are:

- AWS-profile : The profile which will be used by the AWS CLI. heck the file ~/.aws/config for the matching one. Using the AWS CLI command aws configure list will provide the same information cluster-node2:
- The name or IP address of the other cluster node
- aws_tag_containing_hostname: The name of the tag of the EC2 instances for the two cluster nodes. We used the name *pacemaker* in this documentation

9 Appendix: Useful Links, Manuals, and SAP Notes

9.1 Appendix: SUSE Best Practices and more

- Best Practices for SAP on SUSE Linux Enterprise: https://www.suse.com/products/sles-for-sap/resource-library/sap-best-practices.html
- Fail-Safe Operation of SAP HANA®: SUSE Extends Its High-Availability Solution: http://scn.sap.com/community/hana-in-memory/blog/2014/04/04/fail-safe-operation-of-sap-hana-suse-extends-its-high-availability-solution
- HOW TO SET UP SAPHanaSR IN THE COST OPTIMIZED SAP HANA SR SCENARIO http://scn.sap.com/docs/DOC-65899

9.2 Appendix: SUSE Product Documentation

The SUSE product manuals and documentation can be downloaded at www.suse.com/ documentation.

- Current online documentation of SLES for SAP https://www.suse.com/documentation/sles_for_sap/
- Current online documentation of SUSE Linux Enterprise High Availability Extension https://www.suse.com/documentation/sle_ha/
- Tuning guide for SUSE Linux Enterprise Server https://www.suse.com/documentation/sles11/book_sle_tuning/data/book_sle_tuning.html
- Storage admin guide for SUSE Linux Enterprise Server https://www.suse.com/documentation/sles11/stor_admin/data/bookinfo.html
- Release notes https://www.suse.com/releasenotes/
- TID Estimate correct multipath timeout http://www.suse.com/support/kb/doc.php?id=7008216
- TID How to load the correct watchdog kernel module http://www.suse.com/support/kb/doc.php?id=7016880
- TID Performance issues after upgrade from SLES11 SP1 to SP2 or SP3 http://www.suse.com/support/kb/doc.php?id=7011982
- TID Addressing file system performance issues on NUMA machines http://www.suse.com/support/kb/doc.php?id=7008919
- TID Low write performance on SLES 11/12 servers with large RAM https://www.suse.com/support/kb/doc.php?id=7010287
- TID Overcommit Memory in SLES https://www.suse.com/support/kb/doc.php?id=7002775
- SLES technical information https://www.suse.com/products/server/technical-information/
- XFS file system https://www.suse.com/communities/conversations/xfs-the-file-system-of-choice/

9.3 Appendix: SAP Product Documentation

- SAP HANA Installation and Update Guide http://help.sap.com/hana/SAP_HANA_Server_Installation_Guide_en.pdf
- SAP HANA Administration Guide http://help.sap.com/hana/SAP_HANA_Administration_Guide_en.pdf

9.4 Appendix: SAP Notes

- 1310037 SUSE LINUX Enterprise Server 11: Installation notes
- 1824819 SAP HANA DB: Recommended OS settings for SLES 11 / SLES for SAP Applications 11 SP4
- 1876398 Network configuration for System Replication in HANA SP6
- 611361 Hostnames of SAP servers
- 1275776 Preparing SLES for Sap Environments
- 1514967 SAP HANA: Central Note
- 1523337 SAP In-Memory Database 1.0: Central Note
- 1501701 Single Computing Unit Performance and Sizing
- 1944799 SAP HANA Guidelines for SLES Operating System Installation
- 1954788 SAP HANA DB: Recommended OS settings for SLES 11 / SLES for SAP Applications 11 SP3
- 1855805 Recommended SLES 11 packages for HANA support on OS level
- 1890444 Slow HANA system due to CPU power save mode
- 1867783 XFS Data Inconsistency Bug with SLES 11 SP2
- 1888072 SAP HANA DB: Indexserver crash in strcmp sse42
- 1846872 "No space left on device" error reported from HANA

9.5 Appendix: Examples

9.5.1 Appendix: Example Cluster Configuration

The following complete crm configuration is for a two-node cluster (suse01, suse02) and a SAP HANA database with SID HA1 and instance number 10. The virtual IP address in the example is 192.168.10.15 .

```
node suse01
node suse02

primitive rsc_SAPHanaTopology_HA1_HDB10 ocf:suse:SAPHanaTopology \
        operations $id="rsc_sap2_HA1_HDB10-operations" \
        op monitor interval="10" timeout="300" \
        op start interval="0" timeout="300" \
        op stop interval="0" timeout="300" \
        params SID="HA1" InstanceNumber="10"
primitive rsc_SAPHana_HA1_HDB10 ocf:suse:SAPHana \
        operations $id="rsc_sap_HA1_HDB10-operations" \
        op monitor interval="61" role="Slave" timeout="700" \
        op start interval="0" timeout="3600" \
        op stop interval="0" timeout="3600" \
        op promote interval="0" timeout="3600" \
        op monitor interval="60" role="Master" timeout="700" \
        params SID="HA1" InstanceNumber="10" PREFER_SITE_TAKEOVER="true"
DUPLICATE_PRIMARY_TIMEOUT="7200" AUTOMATED_REGISTER=“false“
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 \
        meta target-role=Started \
        params tag=pacemaker profile=cluster
primitive rsc_ip_HA1_HDB10 ocf:suse:aws-vpc-move-ip \
        params ip=192.168.10.15 routing_table=rtb-d29969be interface=eth0 profile=cluster
 \
        op start interval=0 timeout=180 \
        op stop interval=0 timeout=180 \
        op monitor interval=120 timeout=60
ms msl_SAPHana_HA1_HDB10 rsc_SAPHana_HA1_HDB10 \
        meta clone-max="2" clone-node-max="1" interleave="true"
clone cln_SAPHanaTopology_HA1_HDB10 rsc_SAPHanaTopology_HA1_HDB10 \
        meta clone-node-max="1" interleave="true"
colocation col_saphana_ip_HA1_HDB10 2000: \
        rsc_ip_HA1_HDB10:Started msl_SAPHana_HA1_HDB10:Master
```

```
order ord_SAPHana_HA1_HDB10 2000: \
        cln_SAPHanaTopology_HA1_HDB10 msl_SAPHana_HA1_HDB10
property cib-bootstrap-options: \
        have-watchdog=false \
        dc-version=1.1.15-21.1-e174ec8 \
        cluster-infrastructure=corosync \
        stonith-enabled=true \
        stonith-action=poweroff \
        stonith-timeout=150s \
        last-lrm-refresh=1518102942 \
        maintenance-mode=false
rsc_defaults $id="rsc_default-options" \
        resource-stickiness="1000" \
        migration-threshold="5000"
op_defaults $id="op_defaults-options" \
        timeout="600"
```

9.5.2 Appendix: Example for /etc/corosync/corosync.conf

The following file shows a typical corosync configuration with one ring. Please view SUSE product documentation about details and about additional rings.

```
# Please read the corosync.conf.5 manual page

totem {

  version: 2
  token: 5000
  consensus: 7500
  token_retransmits_before_loss_const: 6
  secauth: on
  crypto_hash: sha1
  crypto_cipher: aes256
  clear_node_high_bit: yes
  interface {
    ringnumber: 0
    bindnetaddr: 10.79.254.249
    mcastport: 5405
    ttl: 1
  }

  transport: udpu

}
```

```
nodelist {
  node {
  ring0_addr: 10.79.254.249
  nodeid: 1
  }

  node {
  ring0_addr: 10.79.9.213
  nodeid: 2
  }
}

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
        }
}


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
}
```

9.5.3 Appendix: Checklist AWS Installation

Check your AWS configuration upfront and gather the following AWS items before you start the installation:

Checklist AWS installation	
Item	**Status/Value**
SLES subscription and update status	

Checklist AWS installation	
All systems have a SLES for SAP subscription	
All systems have a public cloud channel	
All system have been updated to use the latest patch level	
AWS User Privileges for the installing person	
Creation of EC2 instances and EBS volumes	
Creation security groups	
Modification of AWS routing tables	
Creation policies and attach them to IAM roles	
Potentially needed :Creation of subnets and routing tables	
VPC and Network	
VPC Id	
CIDR range of VPC	
Subnet id A for systems in first AZ	
Subnet id B for systems in second AZ	
Routing table id for subnet A and B	
Is this routing table associated with both subnets?	
Alternative: Is it associated to VPC? Subnets do not have their own ones	
AWS Policies Creation	

Checklist AWS installation	
Name of data provider policy	
Name of STONITH policy	
Name of Move IP (Overlay IP) policy	
First cluster node (initially primary server)	
instance id	
ENI id	
IP address	
hostname	
instance is associated to subnet A?	
instance has all 3 policies attached?	
EC2 tag *pacemaker* set with host-name?	
AWS CLI profile *cluster* created and set to *text*?	
source/destination check disabled?	
Second cluster node (initially secondary server)	
instance id	
ENI id	
IP address	
hostname	
instance is associated to subnet B?	
instance has all 3 policies attached?	
EC2 tag *pacemaker* set with host-name?	

Checklist AWS installation	
AWS CLI profile *cluster* created and set to *text*?	
source/destination check disabled?	
Overlay IP address: database service	
IP address	
Has it been added to routing table?	
Does it point to the ENI of first node?	
Internet access	
All instance have Internet access ? Check routing tables	
Alternative: Add http proxies for data providers and cluster software	