Hewlett Packard Enterprise

Reference Architecture

HPE Reference Configuration for Linux based high availability for SQL Server 2017 on HPE Synergy

Contents

Executive summary

The demands of database implementations continue to escalate. Faster transaction processing speeds, scalable capacity, and increased flexibility are required to meet the needs of today's business. In the day-to-day management of SQL Server database environments, administrators need to be able to quickly deploy new servers, easily update existing systems, and upgrade processing capabilities for scale-out performance. With traditional infrastructure, these activities are often disruptive and time consuming.

HPE Synergy is an ideal platform for SQL Server databases, offering fluid resource pools which can be customized for specific database needs. SQL Server can be deployed rapidly through the software-defined intelligence embedded in the HPE Synergy Composer and HPE Synergy Image Streamer. An administrator can utilize HPE Synergy Image Streamer to develop a deployment plan to install and configure both the operating system and application software. A server profile defined in the HPE Synergy Composer can use that deployment plan to configure a new server in a matter of minutes, compared to hours or days utilizing traditional infrastructure.

This Reference Configuration (RC) is ideal for customers running older versions of SQL Server on older infrastructure and seeking better transaction performance, or those looking to migrate from an Oracle deployment to SQL Server on Linux, and/or those desiring a high availability solution for SQL Server on Linux.

It is demonstrated in this RC how to use the functionality provided by HPE Synergy Composer and HPE Synergy Image Streamer to quickly deploy two specific use cases:

1. Microsoft® SQL Server installation.
2. Microsoft SQL Server Always on availability groups with HPE Serviceguard Solutions for SQL Server on Linux, providing HA capabilities for Always on Availability Instance (AOAI) workload.

In addition, this RC shows the following benefits of utilizing HPE Synergy for SQL Server solutions:

- HPE Synergy Composer with embedded HPE OneView seamlessly manages the entire environment, including configuration of network resources required for SQL Server compute nodes, and deploying the OS and application software on the compute nodes.
- Testing shows that HPE Synergy Composer plus HPE Synergy Image Streamer allows administrators to configure a new system for SQL Server in less than three minutes, which is a significant reduction as compared to traditional deployment times of hours or days.
- HPE Serviceguard for Linux (SGLX) provides high availability to Microsoft SQL Server Always on Availability Group Instance. It reduces the planned downtime for maintenance by keeping the application running on other node.

Target audience: This Hewlett Packard Enterprise white paper is designed for IT professionals who use, program, manage, or administer large databases that require high availability and high performance. Specifically, this information is intended for those who evaluate, recommend, or design new IT high performance architectures.

Document purpose: The purpose of this document is to describe a Reference Configuration, highlighting the usage of HPE Synergy Image Streamer to deploy Microsoft SQL Server instance on Linux, and deploying SQL Server availability groups with HPE Serviceguard for Linux, to provide high availability capabilities.

Solution overview

HPE Synergy enables IT organizations to accelerate application and service delivery through a single interface that composes physical and virtual compute, storage, and fabric pools into any configuration for any application. Composable resources (compute, storage and fabric resources) are provisioned together with their state (determined by variables such as BIOS settings, firmware, drivers, and protocols) and their OS and application image using repeatable templates. This is ideal for applications such as SQL Server because it eliminates time-consuming provisioning processes.

The key components of this solution are the HPE Synergy Composer and HPE Synergy Image Streamer. The combination of these tools allows automating the customization of a Red Hat® Enterprise Linux® (RHEL) 7.4 OS image, configuration of Microsoft SQL Server and HPE Serviceguard software, to quickly deploy a new environment. In this solution for Microsoft SQL Server Always On availability groups deployment with HPE Serviceguard Always on Availability Instance (AOAI) workload, three cluster nodes with one quorum server has been used.

Figure 1 shows the components that were used for this solution.

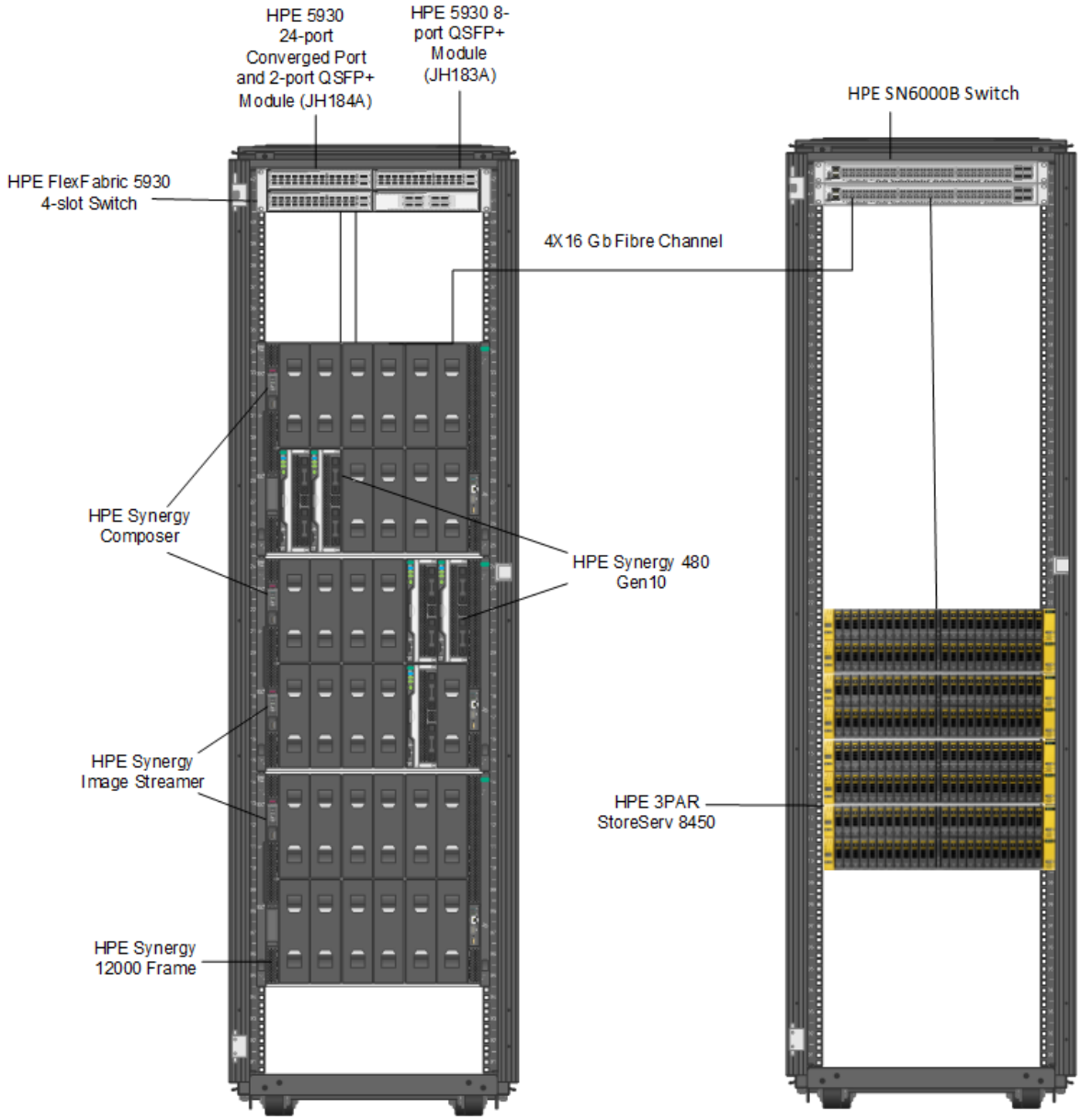

Figure 1. HPE Synergy Solution diagram

Solution components

The HPE Synergy components used in this solution included the following:

- Three HPE Synergy 12000 Frames
  - Two HPE Synergy Composers
  - Two HPE Synergy Image Streamers
  - Five HPE Synergy 480 Gen10 Compute Modules

- Network Interconnects
  - Two Virtual Connect SE 40 Gb F8 Modules
  - Four HPE Synergy 20Gb Interconnect Link Modules
  - Four Virtual Connect SE 16Gb FC Modules for Synergy provided FC connectivity
- External HPE 3PAR SAN storage

For this solution, HPE 3PAR StoreServ 8450 all-flash storage was used for the SQL database data and log files. The all-flash performance, as well as mission-critical resiliency of the HPE 3PAR StoreServ 8450, make it ideal for both SQL Server data and log files.

HPE Synergy 12000 Frame

The HPE Synergy 12000 Frame is the base infrastructure that ties together compute, storage, network fabric, and power into a scalable solution that easily addresses and scales with various customer workloads and infrastructure. The Synergy 12000 reduces complexity in the IT infrastructure by unifying all these resources into a common bus, and with the myriad of available network and storage interconnects, allows the frame to interoperate with any other IT environment. At a high level the HPE Synergy frame supports the following:

- 12 half-height or 6 full-height compute modules per frame. The HPE Synergy design allows for the inclusion of double-wide modules as well as support for internal storage with the HPE Synergy D3940 Storage Module
- Two Frame Link Modules for in-band and out-of-band management
- Up to six 2650 watt power supplies and ten fans
- Up to six interconnect modules for full redundancy of three fabrics

The HPE Synergy 12000 features a fully automated and managed composer module. HPE OneView handles all the setup, provisioning, and management both at the physical and logical level.

HPE Synergy Composer

HPE Synergy Composer is a hardware management appliance that is powered by HPE OneView. The HPE Synergy Composer provides a single interface for assembling and reassembling flexible compute, storage, and fabric resources to support business-critical applications and a variety of workloads, whether they are bare metal, virtualized, or containerized.

The HPE Synergy Composer provides lifecycle management to deploy, monitor, and update your infrastructure using a single interface or the Unified API. IT departments can rapidly deploy infrastructure for traditional, virtualized, and cloud environments in just a few minutes — sometimes in a single step. Resources can be updated, expanded, flexed, and redeployed without service interruptions. Key features of the composer are:

- Simplify deployment and configuration of resources in your environment
- Accelerate updates using templates
- Automate applications and workloads using the Unified API
- Designed for high availability using redundant physical appliances

HPE Synergy Image Streamer

HPE Synergy Image Streamer is a management appliance option for the HPE Synergy solution that is used to deploy stateless compute modules within the HPE Synergy environment. The HPE Synergy Image Streamer solution offers a stateless deployment experience for bare-metal compute modules by managing and maintaining the software state (operating system and settings) separate from the physical state (firmware, BIOS settings, etc.). Boot volumes for the compute modules are hosted and maintained on the HPE Synergy Image Streamer appliance as iSCSI boot volumes. Image Streamer uses scripts and build plans to generalize and personalize the OS boot volumes during capture and deployment.

HPE Synergy Image Streamer adds a powerful dimension to “infrastructure as code”—the ability to manage physical servers like virtual machines. In traditional environments, deploying an OS and applications or hypervisor is time-consuming because it requires building or copying the software image onto individual servers, possibly requiring multiple reboot cycles. In HPE Synergy, the tight integration of HPE Synergy Image Streamer with HPE Synergy Composer enhances server profiles with images and personalities for true stateless operation.

HPE Synergy Composer, powered by HPE OneView, captures the physical state of the server in the server profile. HPE Synergy Image Streamer enhances this server profile (and its desired configuration) by capturing your golden image as the "deployed software state" in the form of bootable image volumes. These enhanced server profiles and bootable OS images, plus application images are software structures (infrastructure as code)—no compute module hardware is required for these operations. The bootable images are stored on redundant HPE Synergy Image Streamer appliances, and they are available for fast implementation onto multiple compute modules at any time. This enables bare-metal compute modules to boot directly into a running OS with applications, and multiple compute modules to be quickly updated. HPE Image Streamer:

- Manages physical servers like virtual machines
- Enables true stateless operation by capturing software (OS and settings) state separate from the hardware (firmware, BIOS) state
- Deploys, updates, and rolls back compute images rapidly for multiple compute modules
- Enables automation via Unified API

Figure 2 shows how HPE Synergy Composer and HPE Synergy Image Streamer manage a compute module via a server profile.

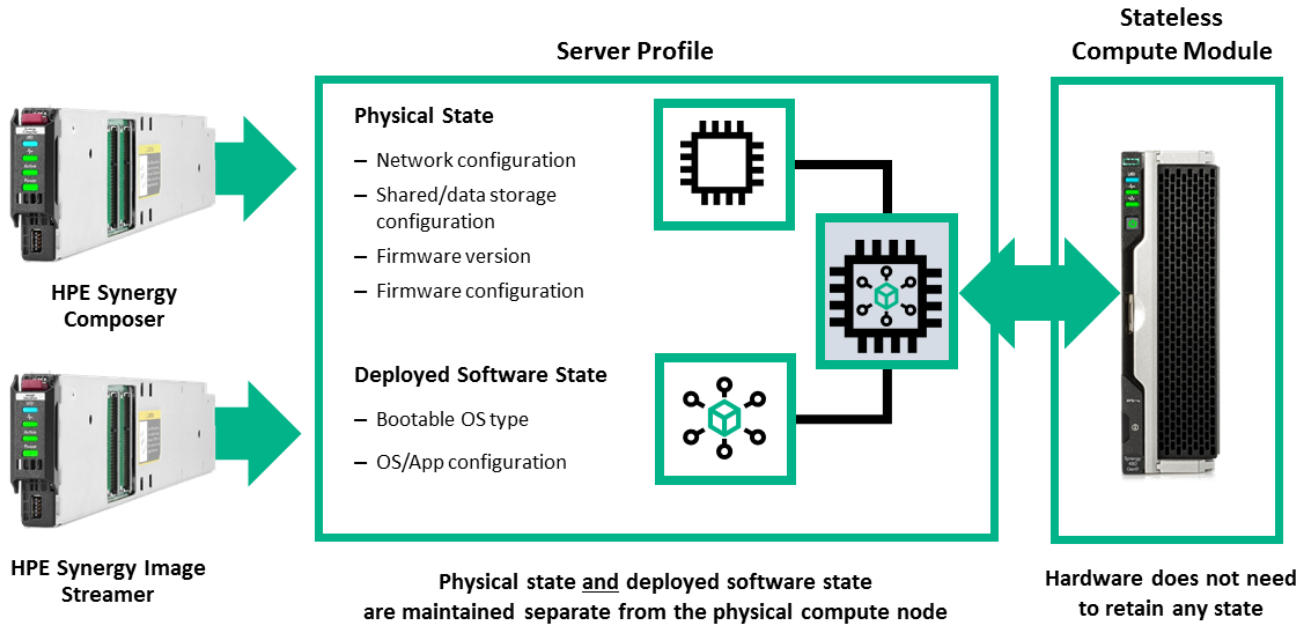

Figure 2. HPE Synergy Composer and HPE Synergy Image Streamer managing compute module with a server profile

HPE Synergy Image Streamer building blocks

HPE Synergy Image Streamer uses the following components for capture and deployment of images:

- **Plan script:** A guestfish script used by OS build plans to personalize OS volumes based on the values of custom attributes. There are three types of plan scripts. Capture type plan scripts are only for use in the capture build plans. Deploy type plan scripts are only for use in the deploy build plans. General type plan scripts are for use in both.
- **OS build plan:** Build plans provide the order of execution for customization of the OS volumes. That is they provide the order in which plan scripts are executed within a deployment plan. A build plan may contain many plan scripts or none at all.
- **Golden image:** A generic format of an application and operating system image that can be customized for multiple deployments.
- **Deployment plan:** A combination of an OS build plan and golden image that is used by a server profile for the deployment of a server.
- **Artifact bundles:** Artifacts included during bundle creation are compressed to a zip file and stored on the appliance. The zipped artifact bundle can then be downloaded for offline storage or uploaded to other Image Streamer appliances in the environment.

Note
Hewlett Packard Enterprise provides artifact bundles for certain supported operating systems. These artifact bundles and other HPE Synergy Image Streamer content can be downloaded from https://github.com/HewlettPackard.

Figure 3 shows the HPE Synergy Image Streamer Dashboard, which displays the resources available to create and modify OS images.

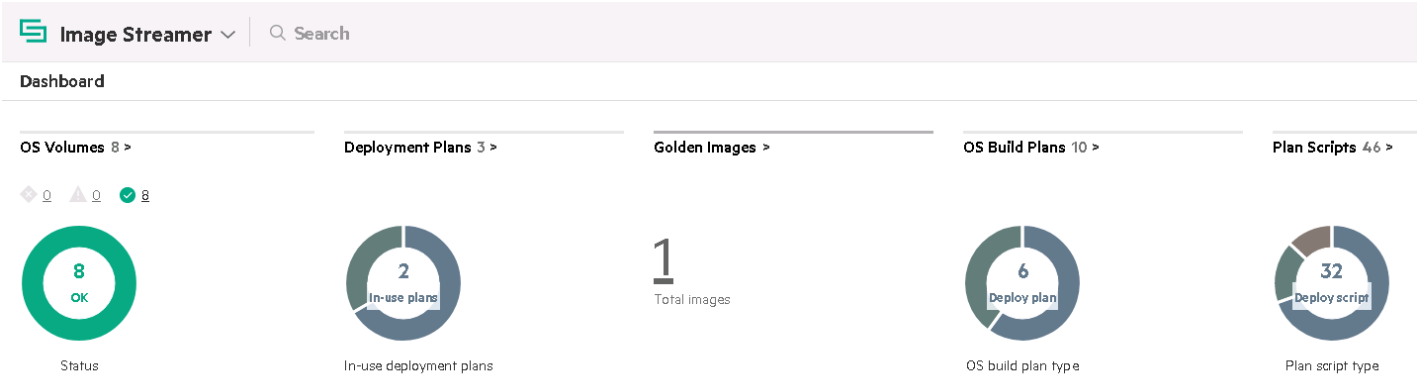

Figure 3. HPE Synergy Image Streamer Dashboard

HPE Synergy Image Streamer supports a variety of operations for flexibility in how you handle your images. For example, you can capture golden images for your use, import golden images from another location, or modify some of your "known good" images for re-use. This flexibility allows you to easily establish your desired images for use. A variety of images can be used on HPE Synergy Image Streamer. Reference implementations provide artifacts for recent versions of VMware® ESXi (5.0, 6.0, 6.5), and application images based on Red Hat Enterprise Linux (RHEL 7.X) and SUSE Linux Enterprise Server (SLES 12 SP1, SP2) using ext3 and ext4 file systems. You can also enable your own specific images and image types using the tools provided with HPE Synergy Image Streamer. In this solution, we have used HPE Synergy Image Streamer to deploy two specific use cases: (1) Microsoft SQL Server installation on RHEL7.4, and (2) Microsoft SQL Server Always On availability groups with HPE Serviceguard cluster.

Hardware

In this solution, the following configuration was used for testing, but the solution can be built with other larger and smaller configurations.

Three HPE Synergy 12000 Frames had the following components.

First frame

- One HPE Synergy Composer
- One HPE Virtual Connect SE 40 Gb F8 Module for Synergy
- One HPE Synergy 20Gb Interconnect Link Module
- 2 x HPE Virtual Connect SE 16Gb FC Module for Synergy
- 2 x HPE Synergy Frame Link Module (active and standby)
- 2 x HPE Synergy 480 Gen10 Compute Module.

Second frame

- One HPE Synergy Composer
- One HPE Synergy Image Streamer
- One HPE Virtual Connect SE 40 Gb F8 Module for Synergy
- One HPE Synergy 20Gb Interconnect Link Module

- 2 x HPE Virtual Connect SE 16Gb FC Module for Synergy
- 2 x HPE Synergy Frame Link Module (active and standby)
- 3 x HPE Synergy 480 Gen10 Compute Module.

Third frame

- One HPE Synergy Image Streamer
- 2 x HPE Synergy 20Gb Interconnect Link Module
- 2 x HPE Synergy Frame Link Module (active and standby)

As a best practice it is recommended to have the three compute nodes being carved out from different synergy frames. Or in general not more than half of the compute nodes part of Availability Group should be carved out of one Single Synergy Frame. So as to a loss of single synergy frame will not bring down the complete workload.

Storage configuration

The HPE 3PAR StoreServ 8450 was configured as follows:

- 1 x HPE 3PAR StoreServ 8450 all-flash array
  - Four controller nodes
  - 384GiB of cache
  - 8 x drive enclosures
  - 64 x 480GB MLC SSD
  - 20 x 16Gb Fibre Channel ports

For the SQL database, the HPE 3PAR StoreServ 8450 provided the backend storage. When provisioning storage from the HPE 3PAR array, two different storage concepts are presented:

- **Common Provisioning Group (CPG):** A CPG is a pool of drives from which virtual or logical storage volumes can be allocated. CPGs also dictate which RAID level will be used when virtual storage volumes are created from this pool. Fully provisioned virtual volumes (FPVVs), thinly provisioned virtual volumes (TPVV), thinly deduped virtual volumes (TDVVs), and compressed VVs can be created that draw space from a CPG's logical disks. Full provisioned virtual volumes allocate volume space at the time of creation, where as thin provisioned virtual volumes allocate volume space on demand. It is important to note that if no volumes of any type have been created in a CPG, it consumes no space.
- **Virtual Volume (VV):** Virtual volumes are the storage LUNs that are presented to the host. While creating virtual volumes, a CPG must be specified. All data residing on the virtual volume will be spread across the drives residing in the CPG.

Figure 4 below provides an overview of the HPE 3PAR storage model.

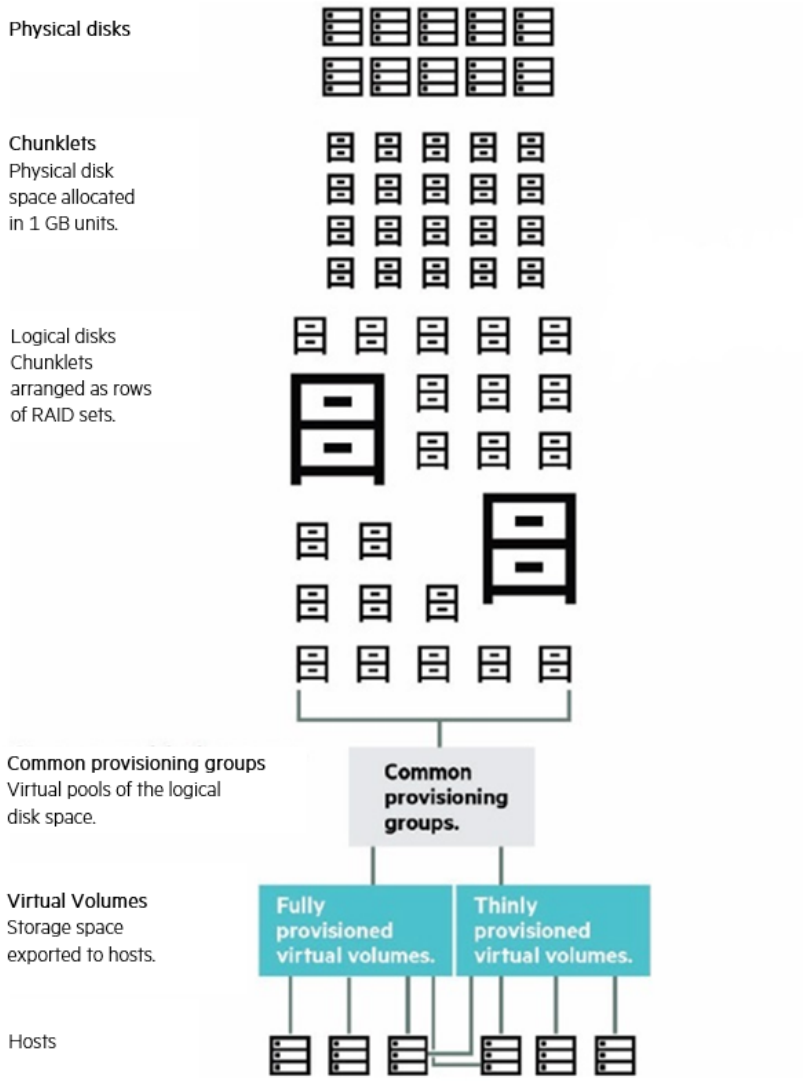

Figure 4. HPE 3PAR storage hierarchy

Best practices were implemented for RAID usage, SAN zoning, and connection redundancy from the HPE 3PAR StoreServ storage best practices guide at http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA4-4524ENW.

Thin provisioning was used for all virtual volumes. Table 1 provides details about the CPG configuration for the HPE 3PAR array. A RAID 5 CPG was created for data volumes and RAID 1 for log volumes.

Table 1. HPE 3PAR CPG layout

	RAID Type	Media	Content
SSD_RAID5	5	SSD	Data Virtual Volumes
SSD_RAID1	1	SSD	Log Virtual Volumes

As a best practice it is recommended to have the SQL Database and Logs reside on different CPG for each compute node. Or in general not more than half of the compute nodes part of Availability Group should have their SQL Database Data and Log files residing on one single CPG. So a loss or problem with one CPG will not bring down the complete workload.

HPE Serviceguard for Linux (SGLX)

HPE Serviceguard for Linux (SGLX) is a high availability solution that provides business continuity and disaster recovery capabilities to customers' x86/Linux environment. SGLX intelligently monitors applications so that when a failure is detected, it automatically and transparently recovers them in seconds without compromising data integrity and performance.

Using Serviceguard Manager graphical user interface, you can configure and manage the Serviceguard cluster and the Microsoft SQL Server 2017 workloads easily.

SQL Server 2017 supports the following high availability and disaster recovery deployment models:

- Failover Cluster Instance (FCI). Based on shared storage architecture, a FCI contains two or more cluster nodes with only one node active at a time and secondary node(s) are available but passive, ready to take an active role during failover.
- Always On availability groups (AG). In this architecture, primary and standby databases are configured with SQL Server Always On availability groups. The databases can be located on the same premises or in geographically dispersed data centers and data can be replicated by SQL Server.

In order to achieve high availability and disaster recovery, both of these architectures would require a cluster manager to perform various critical tasks like monitoring the health of databases, server, network, storages, virtualization layer, virtual machine guests, and OS. Then take automatic actions to minimize application downtimes. HPE Serviceguard for Linux (SGLX) introduces the support of SQL Server to provide mission critical class of robustness to SQL Server deployments and ensures robust monitoring, reliable actions, and protection against data loss. This market-leading high availability and disaster recovery clustering solution protects your applications from a multitude of infrastructure and application faults across physical or virtual environments over any distance. It reduces the impact of unplanned downtime without compromising data integrity and performance, and helps you achieve near zero planned downtime for maintenance.

When SGLX detects a failure in case of Failover Cluster Instance deployments, it follows shutdown procedures, and recovers the database by restarting SQL Server on the adoptive cluster node. In case of (AG) deployments, SGLX will also monitor and administer the replication between primary and standby databases. In case of failures SGLX will perform automatic role management to recover from failures by promotion of the standby database instance to primary. With Availability Groups support, SGLX provides database level protection with much faster recovery times. In this reference architecture we have used Serviceguard Always On Availability Instance (AOAI) for high availability.

HPE Serviceguard AOAI workload

Serviceguard for AOAI workload provides HA and DR protection for an availability group.

Using HPE Serviceguard for Linux along with SQL Server Always on Availability groups ensures that failures at just about any level can be automatically mitigated within seconds.

Following are some of the salient features of protection by Serviceguard for Microsoft SQL Server Availability Groups.:

- Monitoring of diagnostic data and health information of the SQL Server, and various other components as required by the Microsoft SQL Server.
- Out of the box integration.

- Provides Availability Group (Database) Level Protection for failover and failback.
- Automatically identifies and manages the role(s) of different Availability Group replica(s).
- Automatic failure detection and role promotion at the Availability Group Level.
- Support for RPO based failover using REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT.
- Support for Multiple Availability Groups in same cluster.
- Robust mechanism to promote a replica as primary that is most up to date and Health in the AG
- Easy deployment and management from Serviceguard Manager Graphical User Interface.

An AOAI workload consists of three packages:

- **SQL Server Instance Package (SVR-MNP):** This package is a Serviceguard multi node package and monitors the SQL Server processes and the server instance health. You can create only one SVR-MNP package for the entire cluster.
- **Always on Availability Instance Package (AOAI-MNP):** This package is a Serviceguard multi node package and it monitors the availability group specified in the package configuration. One AOAI-MNP package is required for each availability group.
- **Write Intent Package (AOAI-WI):** This package is a Serviceguard failover package configured for an availability group. This package runs on the Serviceguard node which is the current primary replica. This package also signifies which Serviceguard node has write access to the availability databases that are configured in the availability group.

All three packages together monitor the Microsoft SQL availability group and provide HA and DR protection for a single availability group. A new workload has to be configured for each availability group.

Quorum Server

For Serviceguard AOAI workload for Microsoft SQL Server availability groups Deployment we have used Quorum Server.

The Serviceguard Quorum Server provides arbitration services for Serviceguard clusters when a cluster partition is discovered: when equal-sized groups of nodes become separated from each other, the Quorum Server allows one group to achieve quorum and form the cluster, while the other group is denied quorum and cannot start a cluster. The Quorum Server runs on a Linux system outside of the cluster for which it is providing quorum services. Quorum server should be preinstalled for Serviceguard AOAI workload. See Resources and additional links for more information about HPE Serviceguard for Linux.

Software

The following Operating system was used for the solution.

- Red Hat Enterprise Linux version 7.4

Application software

The following management and application software was used for the solution.

- Microsoft SQL Server 2017 (RTM-CU7)
- HPE Serviceguard for Linux A.12.20.00
- HPE OneView 4.10
- HPE 3PAR StoreServ Management Console 3.1.0.22913

Best practices and configuration guidance for the solution

The high level steps for creating 1) a deployment plan, and 2) a server profile for a Microsoft SQL Server environment are listed below. The details for implementing each of these steps are provided in the following sections.

Steps to develop an HPE Synergy Image Streamer deployment plan

1. Create golden image
2. Create plan scripts to customize golden image
3. Create OS build plan using plan scripts
4. Create deployment plan using OS build plan and golden image

Steps to deploy a compute node using the HPE Synergy Image Streamer deployment plan

1. Create required networks
2. Create HPE 3PAR volumes required for SQL Server data and logs
3. Create server profile utilizing networks, storage and OS deployment plan

Develop an HPE Synergy Image Streamer deployment plan

Step 1: Create golden image

Use Case 1: Prepare RHEL 7 golden image for SQL Server instance deployment

The following steps were required to create a RHEL 7 golden image for deploying Microsoft SQL Server instance.

Note

An OS deployment plan must be in place to create an empty OS volume for step one below.

---

1. Create an HPE OneView server profile with an OS deployment plan that creates an empty OS volume of size 40 GB and assign the profile to a compute node.
2. Install RHEL 7.4 on the empty volume, adding the ip=ibft parameter as an install kernel option.
3. Create /etc/resolv.conf with address of DNS server.
4. Install Microsoft SQL Server package on Linux following Microsoft guidelines. After the package installation finishes, don't run mssql-conf setup. mssql-conf setup will run during the OS boot for the newly deployed node.

After the OS has been customized according to the steps listed above, the Image Streamer "Create Golden Image" interface is used to create an image which is stored on the Image Streamer appliance, as shown in Figure 5. Do the following to create the golden image:

1. Shut down the OS.
2. Find the OS volume number in the HPE OneView server profile created in step one above. It is listed under the OS Deployment section of the profile.
3. On the Image Streamer Golden Image screen, select "Create golden image" and specify a name ("RHEL 7.4 for SQL"), description, OS volume, and Capture OS build plan. Note that the Capture OS build plan, "RHEL-generalize-2018-02-09" is chosen in this case.

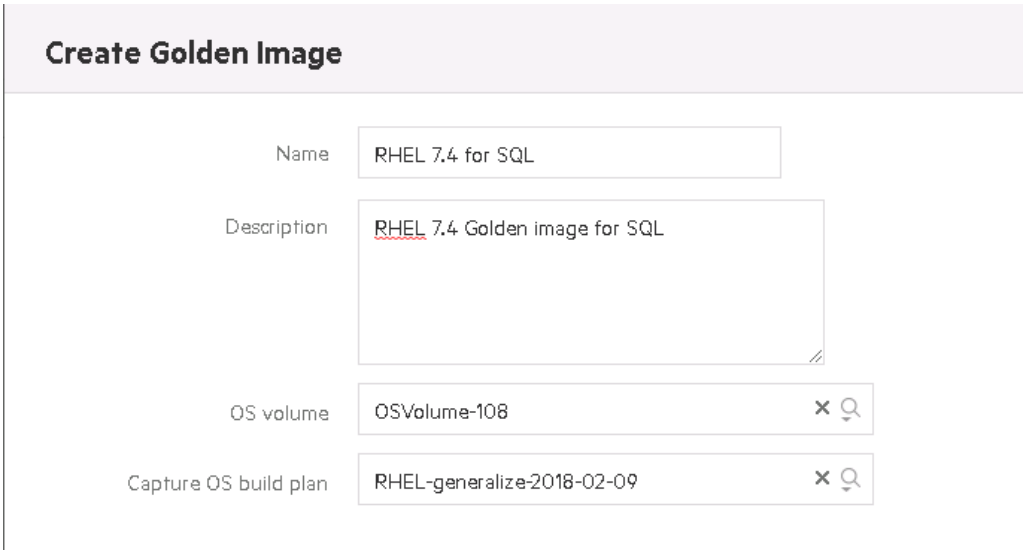

Figure 5. Create golden image

Use Case 2: Prepare RHEL 7 golden image for SQL Server with Serviceguard

In addition to the steps listed above for the SQL Server instance golden image, the following steps are required when creating a golden image for a Serviceguard Always on Availability Instance workload for SQL Server availability group:

Note

HPE Serviceguard for Linux can be installed by using cmeasyinstall or by the traditional way. We have installed Serviceguard using traditional way. For prerequisite and installing service guard refer: https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00039038en_us&docLocale=en_US

---

1. Obtain and install packages for Serviceguard. Mount the Serviceguard for Linux DVD or ISO image. DVD directory structure for Serviceguard <DVD-mount-path>/RedHat/RedHat7/Serviceguard/x86_64/<*.rpm>. If you get a dependency error, install the dependencies and run the Serviceguard installation again.
2. Install Serviceguard Manager RPM. Before installing the RPM, ensure that the user sgmgr exists in the system. Export the SGMGR_ENV environment and run the RPM command.

   export SGMGR_ENV=<password>;rpm -ivh <serviceguard-manager>.rpm
3. Install Serviceguard extension for SQL Server. Directory structure for this package is

   <dist>/<distro_version>/Toolkit/noarch/\serviceguard-extension-for-mssql-A.12.20.00-0.<dist>.noarch.rpm

<dist> is the value that can be either RedHat or SLES based on the operating system. <distro_version> is the value that can be either RedHat7.3 +, or SLES12 SP2+ based on the operating system. <os_version> is the value that can be either rhel7, or sles12 based on the operating system.

4. Install sshpass tool for non-interactive SSH login between the cluster node during the cluster setup.
5. Disable the libvirtd.service.

After the OS has been customized according to the steps listed above, the Image Streamer "Create Golden Image" interface is used to create an image named "RHEL 7.4 Serviceguard for SQL", as shown in figure 6.

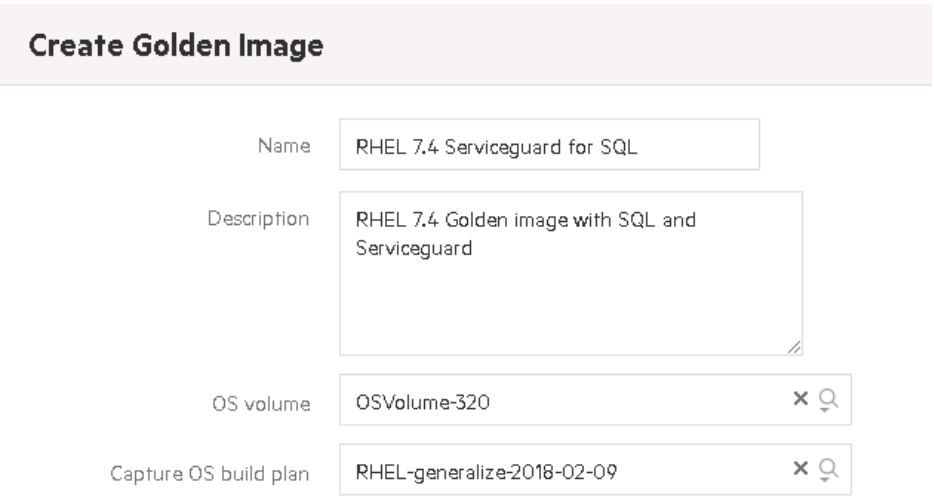

Figure 6. Create golden image for RHEL7.4 with SQL server and Serviceguard

Steps 2 and 3: Create HPE Synergy Image Streamer plan scripts and OS build plans

Use Case 1: Create plan scripts and OS build plan for SQL Server instance deployment

Table 2 lists the HPE Synergy Image Streamer plan scripts that were developed to create an OS build plan named "RHEL 7.4 SQL Standalone" to customize a RHEL 7.4 golden image for an SQL Server instance. The contents of the plan scripts are included in Appendix A.

Table 2. Plan scripts for RHEL 7 OS deployment for SQL Server instance

Plan script name	Purpose
RHEL-mount-and-validate-2017-12-12	Mount root filesystem
RHEL Single NIC Teaming	Configure the single NIC teaming
RHEL-configure-hostname-2017-12-12	Set hostname and add to /etc/hosts
SQLConfig	Configure SQL server setup using mssql-conf setup, enable SQL server agent and configure SQL configuration settings.
RHEL-unmount-2017-12-12	Unmount root file system

Use Case 2: Create plan scripts and OS build plan for SQL server with Serviceguard

Table 3 lists the HPE Synergy Image Streamer plan scripts that were developed to create an OS build plan named "RHEL7.4_SQL_Serviceguard cluster node build Plan". This build plan creates the Serviceguard cluster of three nodes. The contents of the plan scripts are included in Appendix B. Serviceguard provides an easy deployment feature that provides a quick and simple way to create a cluster. Easy deployment automates cluster configuration using the 'cmdeploycl' command which internally calls 'cmpreparecl', cmquerycl','cmapplyconf '. For this setup, after running the 'cmdeploycl' command, we are updating the cluster configuration file to exclude the Image Streamer network and applying the updated configuration with 'cmapplyconf'. For running this script all the three cluster nodes, a quorum server should be up and running. Since the

cluster configuration script must be run after all three nodes are up and running, the plan script 'RHEL – Configure sshkey and deploy cluster' runs from the third node at the first time that the server is booted. This plan script configures sshkey on all three servers to allow all three nodes to access each other, and configure the Serviceguard cluster to complete the cluster configuration.

Table 3. Plan scripts for RHEL 7 OS deployment that configure the Serviceguard cluster for SQL Server

Plan script name	Purpose
RHEL-mount-and-validate-2017-12-12	Mount root filesystem
NIC Teaming for HeartBeat and Stationary_IP for Serviceguard	Configure the two NIC teaming (For Public and heartbeat network)
RHEL-configure-hostname-2017-12-12	Set hostname and add to /etc/hosts
SQLConfig	Configure SQL server setup using mssql-conf setup, enable SQL server agent and configure SQL configuration settings.
RHEL – Configure sshkey and deploy cluster	Create script to configure ssh keys for root account and configure the Serviceguard cluster
RHEL-unmount-2017-12-12	Unmount root file system

Step 4: Create HPE Synergy Image Streamer deployment plans

An Image Streamer deployment plan is created by specifying an OS build plan and a golden image, using the "Create Deployment Plan" dialog box shown in figure 7. Note that the name of the OS build plan has not yet been filled in, because once it is entered, the list of custom attributes will be displayed, making it difficult to view the entire dialog box. The list of custom attributes and their default settings for the deployment plan is included in Appendix C.

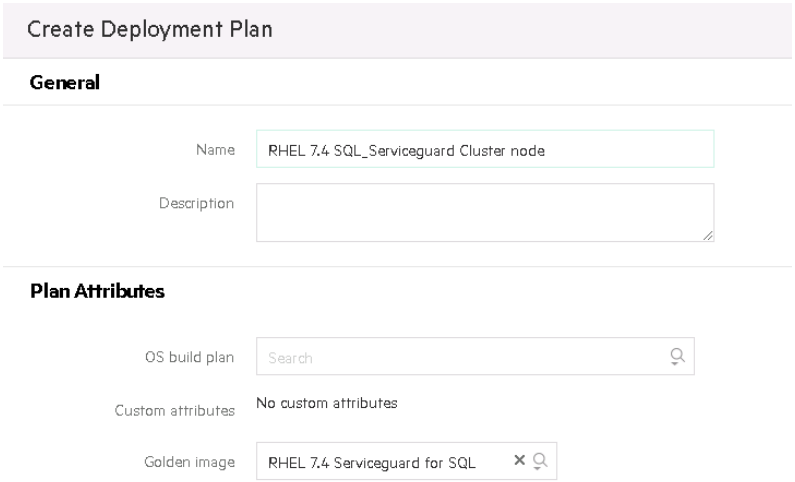

Figure 7. Create Deployment Plan

Deploy a compute node using HPE Synergy Image Streamer deployment plan

This section describes the steps required for deploying a Synergy compute node with an Image Streamer deployment plan. Prior to deploying a compute node, the Synergy environment must be configured with the networks and storage volumes required for SQL server database.

Step 1: Create required networks

The following networks are required in the Synergy environment, for usage in the server profiles that deploy SQL Server instance with HPE Serviceguard cluster:

- Public network for Stationary IP
- Private network for heartbeat

- FC network to connect to external storage (HPE 3PAR)
- Image Streamer deployment network

All networks were created with a preferred bandwidth of 2.5 Gb/second and a maximum bandwidth of 20 Gb/second since they all shared a single Virtual Connect SE 40 Gb F8 Module. Note that the private network for the Serviceguard heartbeat is not assigned a VLAN or uplink set, since communications are only needed between the Synergy compute modules.

Figure 8 shows the Connections section of the server profile for Serviceguard cluster node for Microsoft SQL server. The server profile for a Microsoft SQL server instance without Serviceguard cluster is similar, but does not use the heartbeat private network. While creating the connection create the private heartbeat connections before public stationary IP connections. We have created the teaming for private and public network during the Os deployment. Note that the Image Streamer deployment network is automatically added to the server profile when an OS deployment plan is selected. The iSCSI boot configuration is also automatically added to the profile.

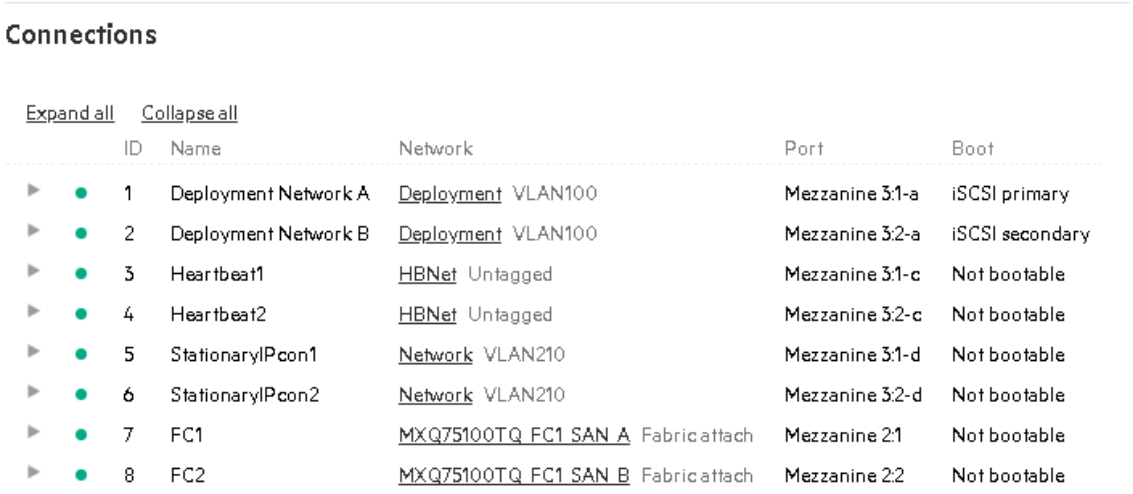

Figure 8. Connections section of server profile for Serviceguard cluster node

Step 2: Storage configuration

The HPE OneView Create Volume screen was used to configure HPE 3PAR storage for the SQL data and log disks. For the SQL server availability group database, we created a local volume for each node with the sharing attribute set to Private, as shown in figure 9.

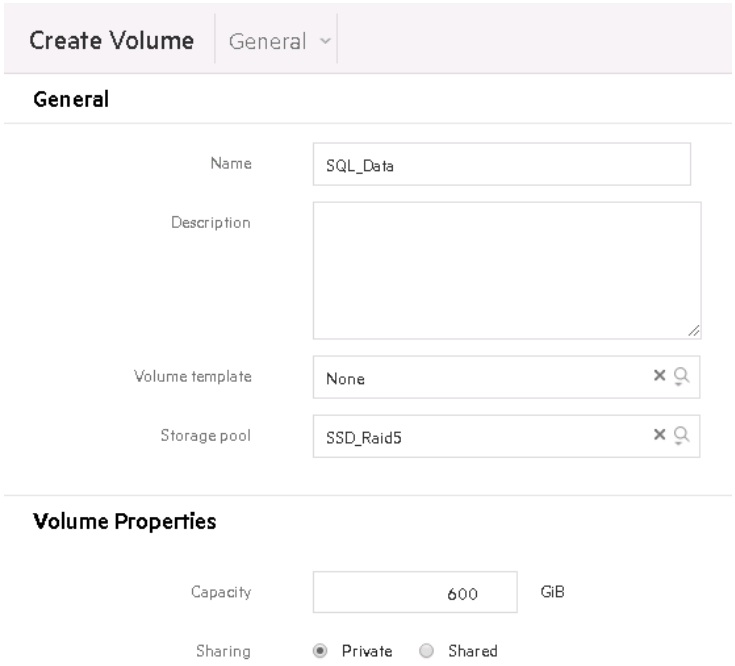

Figure 9. Create local volume for Serviceguard node for SQL AOAI workload

When creating the server profile for the node, all that was needed was to add the existing volumes to the server profile. Figure 10 shows the SAN Storage portion of the server profile for the node.

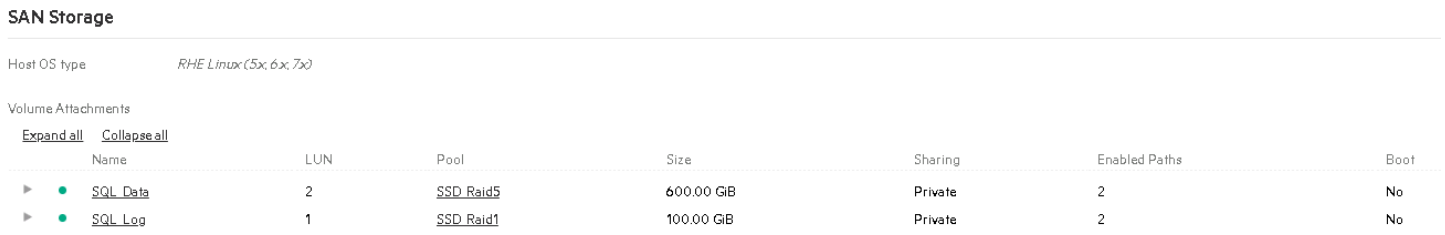

Figure 10. SAN Storage section of server profile for Serviceguard node for SQL AOAI workload

Step 3: Create a server profile

After the required networks have been configured, a server profile can be created that utilizes these components along with an OS deployment plan that will install SQL Server and configure the Serviceguard cluster with three nodes.

Note

Deploying a Serviceguard cluster node also requires that the DNS server for the environment be updated to include the hostname and all IP addresses required for the cluster node.

Figure 11 shows the Create Server Profile dialog box, with the OS Deployment section specifying the OS deployment plan “RHEL 7.4 SQL_Serviceguard Cluster node”. At this point, the user may also modify the deployment settings listed for the deployment plan. This includes settings such as hostnames, domain name, NIC team name, and IP address. For a cluster node, this includes IP addresses for the public network, and the private network for the Serviceguard cluster heartbeat. Also we have SQL server SA password and version parameter, which is used to complete the SQL server setup. The full set of attributes that can be customized for this deployment plan are listed in Appendix C. When the server profile is created, the plan scripts specified in the OS build plan are used to customize the new OS volume. After the profile creation has completed, the compute node can be powered on. Create the server profile for node one and node two and power on. The third node will be last to power on and the steps required to create the three node Serviceguard cluster are run at the first boot.

Note

The creation of a server profile also allows the user to specify firmware updates, boot settings, and BIOS settings to be applied to the server. These steps are not shown here as the focus is on OS deployment.

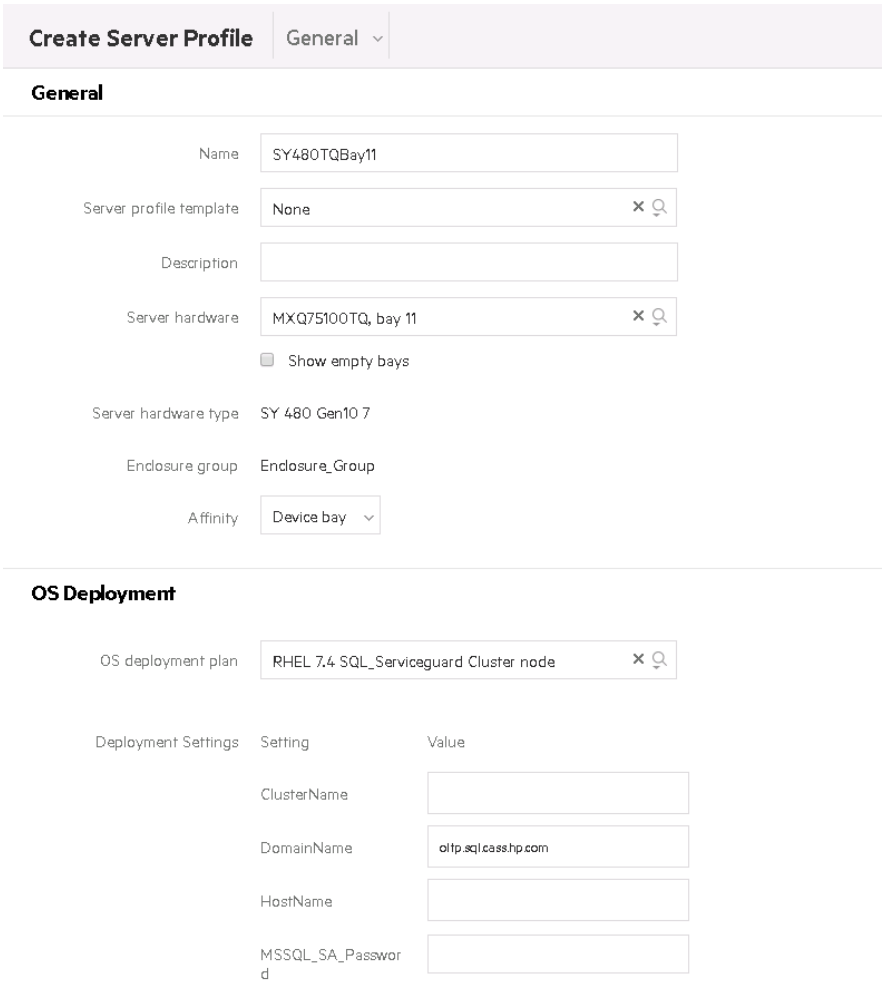

Figure 11. Create server profile

Steps to create SQL Server Always On Availability group with database

After the server profile is applied and the server is powered on, a 3-node Serviceguard cluster is up with preinstalled SQL Server. We have to configure the availability group for SQL Server.

Configure SQL Server Always On Availability Group

1. Create the certificate on the primary SQL Server instance. Copy certificate and private key files to the same location on all servers that will host availability replica. On each target server, give permission to the mssql user to access the certificate.
2. Create the certificate on secondary servers from the files we backup from the primary SQL server in previous step.
3. Create the database mirroring endpoints on all replicas.
4. Create the Availability Group on primary replica server. For this solution we have created an availability group with three synchronous replicas.
5. Join secondary replicas to the availability group.
6. Add a database to availability group. Verify that the database is created on the secondary servers.

Steps to create Serviceguard SQL Server Workloads for Availability groups from Serviceguard Manager

Once the Availability group is created, the Availability Group and all the AG Databases can be made Highly Available via Serviceguard. This can be done by configuring them as a Workload in Serviceguard.

You can deploy the AOAI workload either from the Serviceguard Manager or manually from the CLI. Serviceguard Manager provides for easy deployment of the AOAI workload. We have used Serviceguard Manager to deploy AOAI workload for SQL server. Alternatively one can also deploy the workload via Serviceguard REST Interfaces.

Access the Serviceguard Manager with URL format: https://<fully_qualified_domain>:5522. Login with the root user and password to deploy the workload. You can also use the sgmgr user that provides you with a single pane of glass to manager multiple clusters.

1. Select Workloads under the RESOURCES tab. The page will navigate to Workloads page.
2. In the Workload page, click on Deploy Workload button. Select Microsoft SQL Server from Workload Type dropdown.
3. Enter a name for the workload in the Workload Name textbox.
4. In the Workload Parameters section, provide Microsoft SQL server credentials. Clicking on Query Availability Group lists all the availability groups in the cluster.
5. Under the Workload Parameters:
    a. Select the availability group.
    b. Select subnet and enter the Read-Write IP address.
    c. Optionally you can enter an email address for notification purposes.
    d. Click on Deploy.

Figure 12 shows the workload parameters for the Microsoft SQL Server AOAI workload in Serviceguard Manager. As part of the deployment, three packages, SVR-MNP, AOAI-MNP, and AOAI-WI, are deployed.

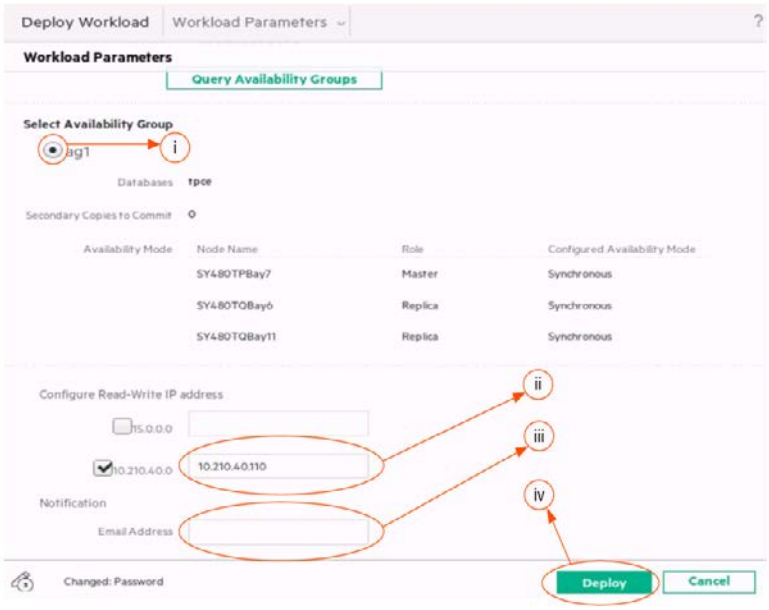

Figure 12. Workload parameters for Microsoft SQL Server AOAI workload in Serviceguard Manager

Capacity and sizing

Using HPE OneView server profiles with HPE Synergy Image Streamer deployment plans allows users to very quickly deploy a server for Microsoft SQL Server instance alone or Serviceguard cluster environment with SQL Server instance. The total time to complete the deployment of compute module for a SQL server instance was 2 minutes and 55 seconds. This includes the time to customize the OS volume, using the plan scripts, at 20 seconds, configuring the connections and the time to apply the profile.

Note

The creation of a server profile also allows the user to specify firmware updates, boot settings, and BIOS settings to be applied to the compute module. These steps can add a significant and highly variable amount of time to the deployment. They are not included here as the focus is on OS deployment using Image Streamer. Firmware updates may be conducted when the compute node is added to the environment, avoiding the need to do this at OS deployment time. The iSCSI boot configuration for the HPE Synergy Image Streamer OS volume is set automatically in the server profile and is included in the timings shown here.

Figure 13 shows the Activity section of a server profile for deploying a compute module for a SQL Server instance.

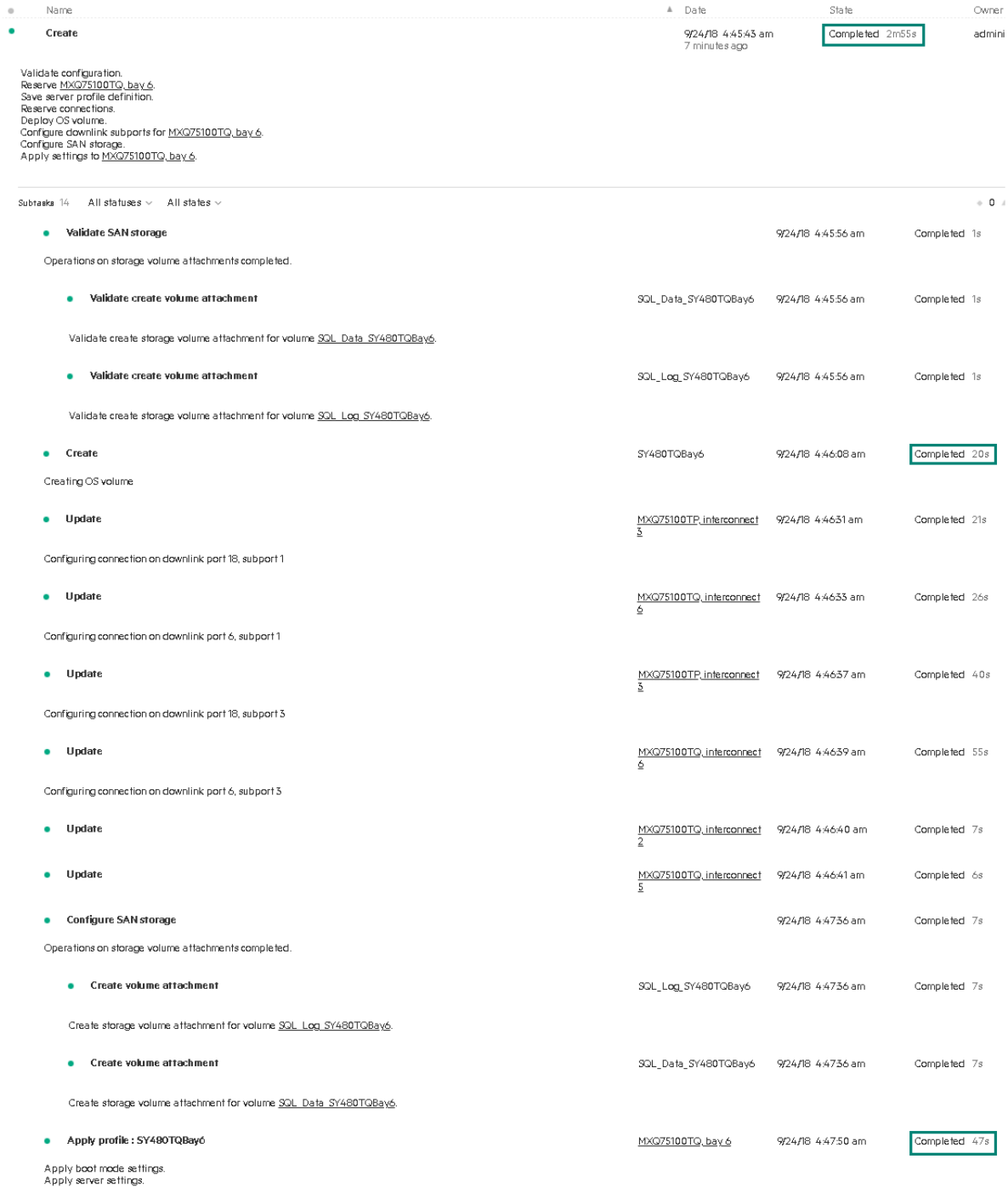

Figure 13. Activity log for deploying compute node for SQL Server instance

Figure 14 shows the Activity section of a server profile for deploying a new node of Serviceguard cluster AOAI workload for SQL Server. The total time to complete the deployment was 3 minutes and 42 seconds. The time required to customize the OS volume (i.e. apply the plan scripts) was a mere 19 seconds. Applying the profile to the server, including the boot mode settings and server settings took 51 seconds.

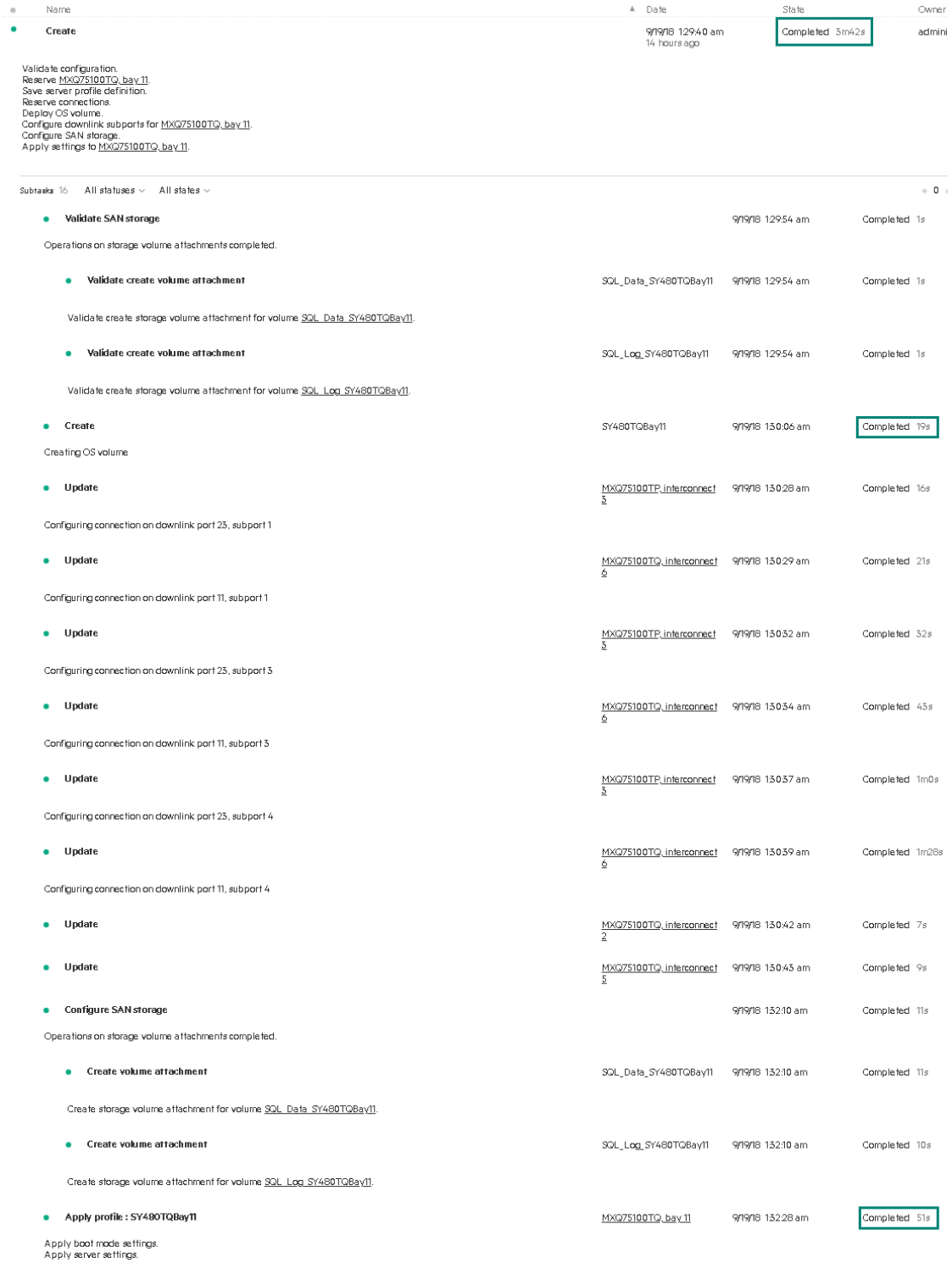

Figure 14. Activity log for deploying new Serviceguard cluster node

Table 4 list the server profile creation and OS boot time for SQL server instance deployment and Serviceguard cluster node deployment with SQL server instance.

Table 4. Server profile creation and OS boot time

Task	Profile Creation	Server Boot	Comment
Deploying compute node for SQL server instance	2 min 55 sec	2 min 10 secs	OS boots in 2 min 10 secs and SQL Server instance get configured.
Deploying new Serviceguard cluster node with SQL server instance	3 min 42 sec	5 min 25 secs	Serviceguard cluster third node boots in 5 min 25 sec and configure the 3 node cluster with other two nodes.

Summary

This Reference Configuration demonstrates how HPE Synergy enables administrators to accelerate Microsoft SQL Server database deployment and easily update their environments. HPE Synergy Composer and HPE Synergy Image Streamer can be utilized to create deployment plans to install and configure a SQL Server instance as well as Serviceguard cluster with SQL Server. The fluid resource pools and software-defined intelligence of HPE Synergy allow administrators to rapidly compose any configuration required, reducing deployment time from hours or days down to minutes. More specifically, this Reference Configuration shows the following benefits of utilizing HPE Synergy for Microsoft SQL Server solutions.

- HPE Synergy Composer with embedded HPE OneView seamlessly manages the entire environment, including configuration of network resources required for SQL Server compute nodes, deploying the OS and application software on the compute nodes.
- Testing shows that HPE Synergy Composer plus HPE Synergy Image Streamer allows administrators to configure a new system for SQL Server in less than three minutes, which is a significant reduction as compared to traditional deployment times of hours or days.

HPE Serviceguard for Linux (SGLX) provides high availability & disaster recovery capabilities to Microsoft SQL Server Always On availability groups. It reduces the planned downtime for maintenance by keeping the application running on other node.

This Reference Configuration describes solution testing performed in August 2018.

Implementing a proof-of-concept

As a matter of best practice for all deployments, HPE recommends implementing a proof-of-concept using a test environment that matches as closely as possible the planned production environment. In this way, appropriate performance and scalability characterizations can be obtained. For help with a proof-of-concept, contact an HPE Services representative (hpe.com/us/en/services/consulting.html) or your HPE partner.

Appendix A: HPE Synergy Image Streamer plan scripts for SQL Server deployment

Note

The scripts in Appendix A and Appendix B are provided as is by HPE. HPE has no obligation to maintain or support this software.

The following plan scripts were used for the build plan "RHEL 7.4 SQL Standalone" which was used in OS deployment plan "RHEL 7.4 and SQL standalone instance". Note that custom attributes (i.e. variables) in the scripts are enclosed by the @ character. These custom attributes are set at deployment time in the deployment plan.

RHEL-mount-and-validate-2017-12-12

```
# List the Volume Group and lvols:
vgs
lvs
list-filesystems

# Mount the root partition (Assume that the partition type is LVM type)
```

```
-mount /dev/rhel/root /
# Mount the root partition (Assume that the partition type is Standard)
-mount /dev/sda3 /

#Create temporary directory
mkdir-p /temp/ImageStreamer


echo "Check Image Streamer capture details"
-download /ImageStreamerCapture ./ImageStreamerCapture

upload -<<END /temp/ImageStreamer/check_capture
#!/bin/bash
if [ -f ./ImageStreamerCapture ]; then
    echo "Capture details:"
    cat ./ImageStreamerCapture
else
    echo
    echo "WARNING: RHEL 7.X Golden Image was not captured by Image Streamer."
    echo "Golden Image may not be prepared for correct personalization."
    echo "Recommend deploying Golden Image as is and capturing a new"
    echo "Golden Image using Image Streamer via correct capture Build Plan"
    echo
fi
echo
END
download /temp/ImageStreamer/check_capture ./check_capture
!source ./check_capture

-rm-f /ImageStreamerCapture

echo "Copy out rc.local"
download /etc/rc.d/rc.local rc.local
#Take a backup
echo "Take a backup of rc.local"
download /etc/rc.d/rc.local rc.local.backup

#Setup rc.local file to run scripts at the boot time.
!echo "#The following commands has been added from the Image Streamer" >> rc.local
```

RHEL Single NIC Teaming

```
#Child script for NIC Teaming.
upload -<<END /temp/NIC-teaming
#! /bin/bash

team_name=$1
net1_mac=$2
net2_mac=$3
net1_ip=$4
net1_gw=$5
domain=$6
dns1=$7
dns2=$8
netmask=$9

mask2cidr() {
    nbits=0
    IFS=.
```

```
    for dec in $1 ; do
        case $dec in
            255) let nbits+=8;;
            254) let nbits+=7;;
            252) let nbits+=6;;
            248) let nbits+=5;;
            240) let nbits+=4;;
            224) let nbits+=3;;
            192) let nbits+=2;;
            128) let nbits+=1;;
            0);;
            *) echo "Error: $dec is not recognised"; exit 1
        esac
    done
    echo "$nbits"
}


#Check the maching interface names for mac addresses.
if [ -f /temp/int-back ]
then
 nic1=$(cat /temp/int-back | grep -i $net1_mac | awk '{ print $1 }')
 nic2=$(cat /temp/int-back | grep -i $net2_mac | awk '{ print $1 }')
else
 interfaces=`ls /sys/class/net/`
 for iface in $interfaces
 do
  mac=`cat /sys/class/net/$iface/address`
  echo "$iface  $mac" >> /temp/int-back
  if [ ${mac,,} == ${net1_mac,,} ]
  then
   nic1=$iface
  elif [ ${mac,,} == ${net2_mac,,} ]
  then
   nic2=$iface
  fi
 done
fi

#Add $team_name connection.

port1=$(nmcli con | grep $nic1 |  awk '{ print $1 }')
port2=$(nmcli con | grep $nic2 |  awk '{ print $1 }')
if [ -z $port1 ] && [ -z $port2 ]
then

  nmcli connection add type team con-name $team_name ifname $team_name ipv4.addresses $net1_ip/$(mask2cidr
$netmask) ipv4.gateway $net1_gw ipv4.method manual config '{"runner": {"name": "activebackup"}, "link_watch":
{"name": "ethtool"}}'
  nmcli connection add type team-slave con-name $team_name-port1 ifname $nic1 master $team_name
  nmcli connection add type team-slave con-name $team_name-port2 ifname $nic2 master $team_name

fi

cat <<CONF>/etc/resolv.conf
# Generated by Image Streamer
search $domain
nameserver $dns1
nameserver $dns2
```

```
CONF

nmcli connection up $team_name

exit 0
END


#Parent script for NIC Teaming.
upload -<<END /temp/ImageStreamer/teaming-configuration
#! /bin/bash

team0=@NICTeam0Name:team0@

echo "echo $team0 > /temp/interface_names" >> ./rc.local
               if [[ "@TeamONIC1.ipaddress@" =~ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]* ]]; then

                       echo "sh /temp/NIC-teaming $team0 @TeamONIC1.mac@ @TeamONIC2.mac@ @TeamONIC1.ipaddress@
@TeamONIC1.gateway@ @TeamONIC1.domain@ @TeamONIC1.dns1@ @TeamONIC1.dns2@ @TeamONIC1.netmask@ " >> ./rc.local

               fi

echo "rm -rf /temp/NIC-teaming" >> ./rc.local
echo "rm -rf /temp/int-back" >> ./rc.local


END

download /temp/ImageStreamer/teaming-configuration ./teaming-configuration
!source ./teaming-configuration
```

RHEL-configure-hostname-2017-12-12

```
# Edit network file to set hostname
upload -<<END /etc/sysconfig/network
#This file has been altered by Image Streamer.
NETWORKING=yes
HOSTNAME=@HostName@
END

# Add /etc/hostname entry for hostname
upload -<<END /etc/hostname
@HostName@
END

# Configure /etc/hosts by running with the master customize service
upload -<<END /temp/configure_hosts
#! /bin/bash
sleep 10s
ifaces=`cat /temp/interface_names`
rm -f /temp/interface_names
read -r -a array <<< $ifaces
iface=${array[0]}
ip_address=`ip a | grep $iface | grep 'inet' | cut -d: -f2 | awk '{ print $2}' | cut -d/ -f1`
fqdn=@HostName@.@DomainName@
alias=`echo "$fqdn" | awk -F'[.]' '{print $1}'`
sed -i "/127.0.0.1/a\\$ip_address $fqdn $alias" /etc/hosts
sed -i "/127.0.0.1/a\\#This file has been altered by Image Streamer." /etc/hosts
unlink $0
```

```
END

echo "-------- new /etc/sysconfig/network ----------"
cat /etc/sysconfig/network

echo "-------- new /etc/hostname ----------"
cat /etc/hostname

#Add the generated script to rc.local to run at boot time.
!echo "sh /temp/configure_hosts" >> rc.local
```

SQLConfig

```
# name:
#   SQLConfig.sh
#
# Purpose:
#   Complete the SQL Server setup using mssql-conf. Enable the SQL Server agent. Configure firewall to allow TCP
port 1433
# Setthe trace flag for deadlock tracing.

mkdir /root/SQLConfig

#
# Creating SQLConfig.sh
#
upload -<<END /root/SQLConfig/SQLConfig.sh


#!/bin/bash -e

# Use the following variables to control your install:

# Password for the SA user (required)
#MSSQL_SA_PASSWORD=@MSSQL_SA_Password@

# Product ID of the version of SQL server you're installing
# Must be evaluation, developer, express, web, standard, enterprise, or your 25 digit product key
# Defaults to developer
MSSQL_PID=@MSSQL_Version@


echo Running mssql-conf setup...
        MSSQL_SA_PASSWORD=@MSSQL_SA_Password@ \
     MSSQL_PID=$MSSQL_PID \
     /opt/mssql/bin/mssql-conf -n setup accept-eula

# SQL Server Agent enable:

echo Enabling SQL Server Agent...
/opt/mssql/bin/mssql-conf set sqlagent.enabled true

# Set trace flags 1204 and 1222 for deadlock tracing :
echo Setting trace flags...
/opt/mssql/bin/mssql-conf traceflag  1204 1222 on

# Restart SQL Server after making configuration changes:
echo Restarting SQL Server...
systemctl restart mssql-server
```

```
# Configure firewall to allow TCP port 1433:
echo Configuring firewall to allow traffic on port 1433...
 firewall-cmd --zone=public --add-port=1433/tcp --permanent
 firewall-cmd --reload


echo Done!

END

chmod 754 /root/SQLConfig/SQLConfig.sh

cat /root/SQLConfig/SQLConfig.sh

# add this file to rc.local so that it will be executed at first boot
#
#!echo "sh /root/SQLConfig/SQLConfig.sh" >> ./rc.local
!echo "su - root -c /root/SQLConfig/SQLConfig.sh" >> ./rc.local
```

RHEL-unmount-2017-12-12

```
!echo "mv -f /etc/rc.d/rc.local.backup /etc/rc.d/rc.local" >> ./rc.local

#upload rc.local
upload ./rc.local /etc/rc.d/rc.local
chmod 755 /etc/rc.d/rc.local
#Upload backup file.
upload ./rc.local.backup /etc/rc.d/rc.local.backup

echo "####"
cat /etc/rc.d/rc.local
echo "####"

#Remove the temporary directpory created
rm-rf /temp/ImageStreamer

#unmount the root partition

unmount /
```

Appendix B: HPE Synergy Image Streamer plan scripts for Serviceguard cluster deployment

The following plan scripts were used for the build plan "RHEL 7.4_SQL_Serviceguard cluster node build Plan" which was used in OS deployment plan "RHEL 7.4 SQL_Serviceguard Cluster node". Note that custom attributes (i.e. variables) in the scripts are enclosed by the @ character. These custom attributes are set at deployment time in the deployment plan.

RHEL-mount-and-validate-2017-12-12

Same as Appendix A

NIC Teaming for HeartBeat and Stationary_IP for Serviceguard

```
#Child script for NIC Teaming.
upload -<<END /temp/NIC-teaming
#! /bin/bash
ip_type=$1
team_name=$2
net1_mac=$3
net2_mac=$4
```

```
net1_ip=$5
domain=$6
dns1=$7
dns2=$8
netmask=$9
net1_gw=${10}


mask2cidr() {
    nbits=0
    IFS=.
    for dec in $1 ; do
        case $dec in
            255) let nbits+=8;;
            254) let nbits+=7;;
            252) let nbits+=6;;
            248) let nbits+=5;;
            240) let nbits+=4;;
            224) let nbits+=3;;
            192) let nbits+=2;;
            128) let nbits+=1;;
            0);;
            *) echo "Error: $dec is not recognised"; exit 1
        esac
    done
    echo "$nbits"
}


#Check the maching interface names for mac addresses.
if [ -f /temp/int-back ]
then
 nic1=$(cat /temp/int-back | grep -i $net1_mac | awk '{ print $1 }')
 nic2=$(cat /temp/int-back | grep -i $net2_mac | awk '{ print $1 }')
else
 interfaces=`ls /sys/class/net/`
 for iface in $interfaces
 do
  mac=`cat /sys/class/net/$iface/address`
  echo "$iface  $mac" >> /temp/int-back
  if [ ${mac,,} == ${net1_mac,,} ]
  then
   nic1=$iface
  elif [ ${mac,,} == ${net2_mac,,} ]
  then
   nic2=$iface
  fi
 done
fi

#Add $team_name connection.

port1=$(nmcli con | grep $nic1 |  awk '{ print $1 }')
port2=$(nmcli con | grep $nic2 |  awk '{ print $1 }')
if [ -z $port1 ] && [ -z $port2 ]
then

 #Teaming for Heartbeat
 if [ $ip_type -eq 1 ];then
```

```
  nmcli connection add type team con-name $team_name ifname $team_name ipv4.addresses $net1_ip/${mask2cidr
$netmask} ipv4.method manual config '{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
  nmcli connection add type team-slave con-name $team_name-port1 ifname $nic1 master $team_name
  nmcli connection add type team-slave con-name $team_name-port2 ifname $nic2 master $team_name

  #Teaming for Stationary IP
  elif [ $ip_type -eq 2 ];then

  nmcli connection add type team con-name $team_name ifname $team_name ipv4.addresses $net1_ip/${mask2cidr
$netmask} ipv4.gateway $net1_gw ipv4.method manual config '{"runner": {"name": "activebackup"}, "link_watch":
{"name": "ethtool"}}'
  nmcli connection add type team-slave con-name $team_name-port1 ifname $nic1 master $team_name
  nmcli connection add type team-slave con-name $team_name-port2 ifname $nic2 master $team_name

 fi

fi

cat <<CONF>/etc/resolv.conf
# Generated by Image Streamer
search $domain
nameserver $dns1
nameserver $dns2
CONF

nmcli connection up $team_name

exit 0
END


#Parent script for NIC Teaming.
upload -<<END /temp/ImageStreamer/teaming-configuration
#! /bin/bash
team_count=2
TeamHB=@NICTeamHBName:TeamHB@
TeamS=@NICTeamSName:TeamS@

for i in `seq 1 $team_count`
do
    if [ $i -eq 1 ];then
                echo "echo $TeamHB > /temp/interface_names" >> ./rc.local
                if [[ "@TeamHBNIC1.ipaddress@" =~ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]* ]]; then
                        echo "sh /temp/NIC-teaming  $i $TeamHB @TeamHBNIC1.mac@ @TeamHBNIC2.mac@
@TeamHBNIC1.ipaddress@ @TeamHBNIC1.domain@ @TeamHBNIC1.dns1@ @TeamHBNIC1.dns2@ @TeamHBNIC1.netmask@ " >>
./rc.local

                fi
    elif [ $i -eq 2 ];then
                echo "echo $TeamS > /temp/interface_names" >> ./rc.local
                if [[ "@TeamSNIC1.ipaddress@" =~ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]* ]]; then
                        echo "sh /temp/NIC-teaming  $i $TeamS @TeamSNIC1.mac@ @TeamSNIC2.mac@
@TeamSNIC1.ipaddress@ @TeamSNIC1.domain@ @TeamSNIC1.dns1@ @TeamSNIC1.dns2@ @TeamSNIC1.netmask@
@TeamSNIC1.gateway@ " >> ./rc.local
                fi
        fi

done
```

```
echo "rm -rf /temp/NIC-teaming" >> ./rc.local
echo "rm -rf /temp/int-back" >> ./rc.local


END

download /temp/ImageStreamer/teaming-configuration ./teaming-configuration
!source ./teaming-configuration
```

RHEL-configure-hostname-2017-12-12

Same as Appendix A

SQLConfig

Same as Appendix A

RHEL – Configure sshkey and deploy cluster

```
# Creating sshkeySGCludep.sh and it distributes SSH keys across node and #configure the Serviceguard cluster,
when executed from third Serviceguard #cluster node for first time boot.
#
#

mkdir /root/bootstrap

#
# Creating sshkeySGCludep.sh
#
upload -<<END /root/bootstrap/sshkeySGCludep.sh

#! /bin/bash
#
#
# Check if all three cluster nodes are up
#
ping -c1 @Node1Hostname@
node1pingret=${echo $?}
ping -c1 @Node2Hostname@
node2pingret=${echo $?}
ping -c1 @Node3Hostname@
node3pingret=${echo $?}
ping -c1 @QuorumServer@
Quorumpingret=${echo $?}

if [ "$node1pingret" -eq 0 ] && [ "$node2pingret" -eq 0 ] && [ "$node3pingret" -eq 0 ]; then

        # Creating ssh keys in the local machine (Node 3)

        ssh-keygen -b 1024 -f /root/.ssh/id_rsa -N "" ;

        # Node 3 and Node 1 - copying ssh keys

        sshpass -p @RootUserPassword@ ssh-copy-id  -o StrictHostKeyChecking=no root@@Node1Hostname@ ;

        sleep 5; ssh-add; ssh root@@Node1Hostname@ "{ssh-keygen -N \"\" -b 1024 -f /root/.ssh/id_rsa; \
        sshpass -p @RootUserPassword@ ssh-copy-id  -o StrictHostKeyChecking=no root@@Node3Hostname@ ;}"
```

```
        # Node 3 and Node 2 - copying ssh keys

        sshpass -p @RootUserPassword@ ssh-copy-id  -o StrictHostKeyChecking=no root@@Node2Hostname@ ;
        sleep 5; ssh-add; ssh root@@Node2Hostname@ "(ssh-keygen -N \"\" -b 1024 -f /root/.ssh/id_rsa; \
        sshpass -p @RootUserPassword@ ssh-copy-id  -o StrictHostKeyChecking=no root@@Node3Hostname@ ;)"

        # Node 1 and Node 2 - copying ssh keys

        # logging into Node 1

    ssh root@@@Node1Hostname@ "(sshpass -p @RootUserPassword@ ssh-copy-id  -o StrictHostKeyChecking=no
root@@Node2Hostname@;)"

        sleep 5; ssh-add; ssh root@@@Node2Hostname@ "(sshpass -p @RootUserPassword@ ssh-copy-id  -o
StrictHostKeyChecking=no root@@Node1Hostname@;)"

        # Configuring the Seviceguard 3 node cluster.

    if [ "$Quorumpingret" -eq 0 ]; then

        echo "StrictHostKeyChecking no" >> /etc/ssh/ssh_config
        service sshd restart

        cmdeploycl -n @Node1Hostname@ -n @Node2Hostname@ -n @Node3Hostname@ -c @ClusterName@ -q @QuorumServer@
        sed -i -e 's/StrictHostKeyChecking/#StrictHostKeyChecking/g' /etc/ssh/ssh_config
    service sshd restart

        # Modify the Serviceguard cluster configuration file.

        cp /usr/local/cmcluster/conf/sgeasy/cluster.ascii /usr/local/cmcluster/conf/sgeasy/cluster.ascii_ORG
    for i in `cat /usr/local/cmcluster/conf/sgeasy/cluster.ascii | grep -A 1 ibft | grep _IP | awk '{print $2}'`
    do
        lineoutput=`ip route | grep $i`
        if [[ -n $$lineoutput ]]; then
        subnet=`echo $lineoutput | awk -F/ '{print $1}'`
        line_number=`cat /usr/local/cmcluster/conf/sgeasy/cluster.ascii | grep -n $subnet | awk -F: '{print $1}'`

                if [[ -n "$line_number" ]] && [[ "$line_number" -gt 0 ]]; then
                                sed -e $line_number'd' /usr/local/cmcluster/conf/sgeasy/cluster.ascii >
/usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
                                cp /usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
/usr/local/cmcluster/conf/sgeasy/cluster.ascii

                                sed -e $line_number'd' /usr/local/cmcluster/conf/sgeasy/cluster.ascii >
/usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
                                cp /usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
/usr/local/cmcluster/conf/sgeasy/cluster.ascii

                        fi
                fi

    grep -v $i /usr/local/cmcluster/conf/sgeasy/cluster.ascii >
/usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
    cp /usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii  /usr/local/cmcluster/conf/sgeasy/cluster.ascii
    done
    grep -v ibft /usr/local/cmcluster/conf/sgeasy/cluster.ascii >
/usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii
    cp /usr/local/cmcluster/conf/sgeasy/cluster_mod.ascii  /usr/local/cmcluster/conf/sgeasy/cluster.ascii
```

```
        /usr/local/cmcluster/bin/cmapplyconf -v -C /usr/local/cmcluster/conf/sgeasy/cluster.ascii

    cmruncl
        fi

fi
exit 0
END

#
#
chmod 754 /root/bootstrap/sshkeySGCludep.sh

echo "==== cat /root/bootstrap/sshkeySGCludep.sh ===="
cat /root/bootstrap/sshkeySGCludep.sh


# add this file to rc.local so that it will be executed at first boot

#

#!echo "sh /root/bootstrap/sshkeySGCludep.sh" >> ./rc.local

!echo "su - root -c /root/bootstrap/sshkeySGCludep.sh" >> ./rc.local
```

RHEL-unmount-2017-12-12

Same as Appendix A

Appendix C: Custom attributes for deployment plan

Table 4. Custom attributes for deployment plan "RHEL 7.4 SQL_Serviceguard Cluster node"

Custom attribute name	Default value	Comment
ClusterName		Serviceguard cluster name
DomainName		Domain name
Hostname		Hostname for node
MSSQL_SA_Password		SQL server SA password
MSSQL_Version	Evaluation	SQL Server version option. (Drop down list containing SQL Server Versions: Evaluation, Developer, Express, Web, Standard, Enterprise, Enterprise Core, if you have product key enter the product key as list option.)
NICTeamHBName	TeamHB	NIC team name for Heartbeat
NICTeamSName	TeamS	NIC team name for Stationary IP
Node1Hostname		Hostname for first Serviceguard cluster node
Node2Hostname		Hostname for second Serviceguard cluster node
Node3Hostname		Hostname for third Serviceguard cluster node
QuorumServer		Quorum server hostname
RootUserpassword		Root user password (Same as for golden image root user password)

Custom attribute name	Default value	Comment
TeamHBNIC1		First Heartbeat NIC connection (Private network)
IPv4 address		Heartbeat IP
Netmask		Network mask for heartbeat network
DNS1		First DNS nameserver for heartbeat IP
DNS2		Second DNS nameserver for heartbeat IP
Domain		Domain for heartbeat network
MAC address		MAC address for first NIC for heartbeat network (from server profile)
TeamHBNIC2		Second Heartbeat NIC connection
MAC address		MAC address for second NIC for heartbeat network (from server profile)
TeamSNIC1		First Stationary IP NIC connection
IPv4 address		Stationary IP (Public network)
Netmask		Network mask for Stationary IP network
Gateway		Gateway IP address for Stationary IP network
DNS1		First DNS nameserver for Stationary IP
DNS2		Second DNS nameserver for Stationary IP
Domain		Domain for Stationary IP
MAC address		MAC address for first NIC for Stationary IP network (from server profile)
TeamSNIC2		Second Stationary IP NIC connection
MAC address		MAC address for second NIC for Stationary IP network (from server profile)

Resources and additional links

HPE Reference Architectures, hpe.com/info/ra

HPE Synergy, hpe.com/synergy

HPE Serviceguard Product Catalog, http://hpe.com/servers/sglx

HPE Serviceguard for Linux Documents, https://support.hpe.com/hpesc/public/home/documentHome?sort_by=relevance&sp4ts.oid=1008734158

HPE Serviceguard Solutions for Microsoft SQL Server for Linux User Guide, https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-a00039040en_us&docLocale=en_US

HPE Serviceguard Rest API Guide, https://support.hpe.com/hpsc/doc/public/display?docId=a00039049en_us

Microsoft SQL Server on Linux, https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview

HPE 3PAR StoreServ Storage reference and best practices guide, http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA4-4524ENW

HPE Storage, hpe.com/storage

HPE networking, hpe.com/networking

HPE Technology Consulting Services, hpe.com/us/en/services/consulting.html

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

Sign up for updates

a00058003enw, October 2018