



NetApp SAP Landscape Management Integration using Ansible

NetApp Solutions SAP

NetApp
March 11, 2024

Table of Contents

- NetApp SAP Landscape Management Integration using Ansible 1
 - TR-4953: NetApp SAP Landscape Management Integration using Ansible 1
 - SAP system clone, copy, and refresh scenarios 1
 - Use cases for system refresh, copy, and cloning 2
 - NetApp SAP LaMa integration using Ansible 5
 - Example implementation 5
 - SAP LaMa provisioning workflow - clone system 10
 - SAP LaMa deprovisioning workflow - system destroy 18
 - SAP LaMa provisioning workflow - copy system 21
 - SAP LaMa provisioning workflow - system refresh 25
 - Provider script configuration and Ansible playbooks 27
 - Conclusion 40

NetApp SAP Landscape Management Integration using Ansible

TR-4953: NetApp SAP Landscape Management Integration using Ansible

Michael Schlosser, Nils Bauer, NetApp

SAP Landscape Management (LaMa) enables SAP system administrators to automate SAP system operations, including end-to-end SAP system clone, copy, and refresh operations.

NetApp offers a rich set of Ansible modules that allows SAP LaMa to access technologies such as NetApp Snapshot and FlexClone through SAP LaMa Automation Studio. These technologies help to simplify and accelerate SAP system clone, copy, and refresh operations.

The integration can be used by customers who run NetApp storage solutions on-premises or by customers using NetApp storage services at public cloud providers such as Amazon Web Services, Microsoft Azure, or Google Cloud Platform.

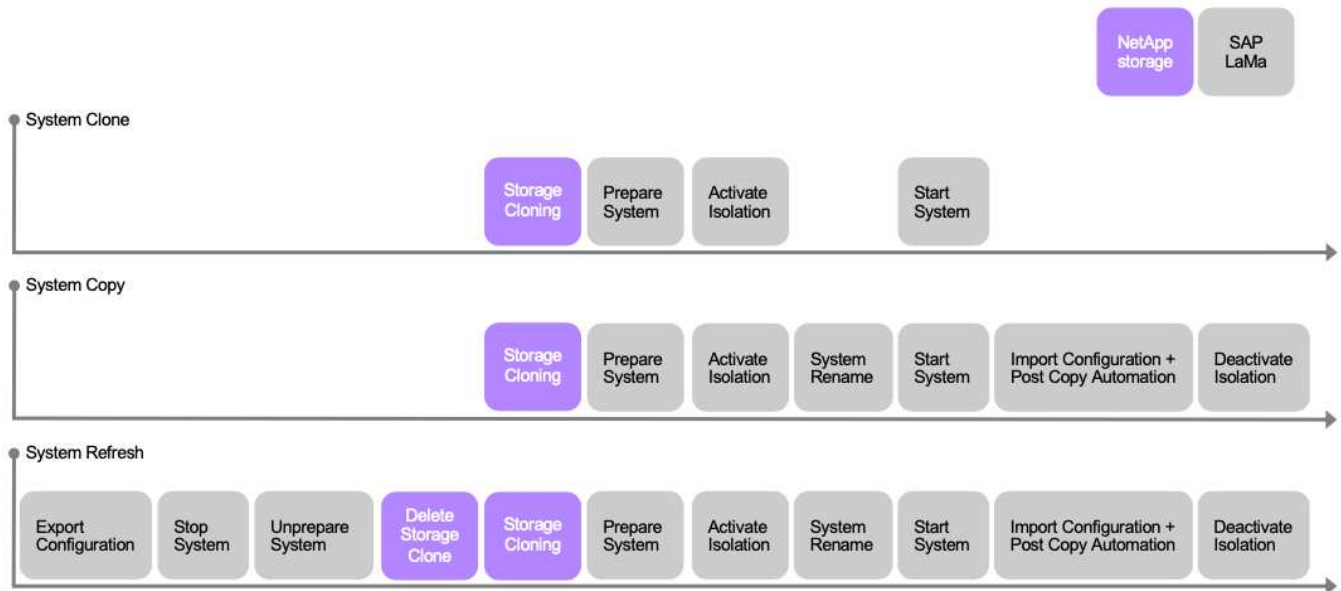
This document describes the configuration of SAP LaMa with NetApp storage features for SAP system copy, clone, and refresh operations using Ansible automation.

SAP system clone, copy, and refresh scenarios

The term SAP system copy is often used as a synonym for three different processes: SAP system clone, SAP system copy, or SAP system refresh. It is important to distinguish between the different operations because the workflows and use cases differ for each one.

- **SAP system clone.** An SAP system clone is an identical clone of a source SAP system. SAP system clones are typically used to address logical corruption or to test disaster recovery scenarios. With a system clone operation, the hostname, instance number, and SID remain the same. It is therefore important to establish proper network fencing for the target system to make sure that there is no communication with the production environment.
- **SAP system copy.** An SAP system copy is a setup of a new target SAP system with data from a source SAP system. The new target system could be, for example, an additional test system with data from the production system. The hostname, instance number, and SID are different for the source and target systems.
- **SAP system refresh.** An SAP system refresh is a refresh of an existing target SAP system with data from a source SAP system. The target system is typically part of an SAP transport landscape, for example a quality assurance system, that is refreshed with data from the production system. The hostname, instance number, and SID are different for the source and target systems.

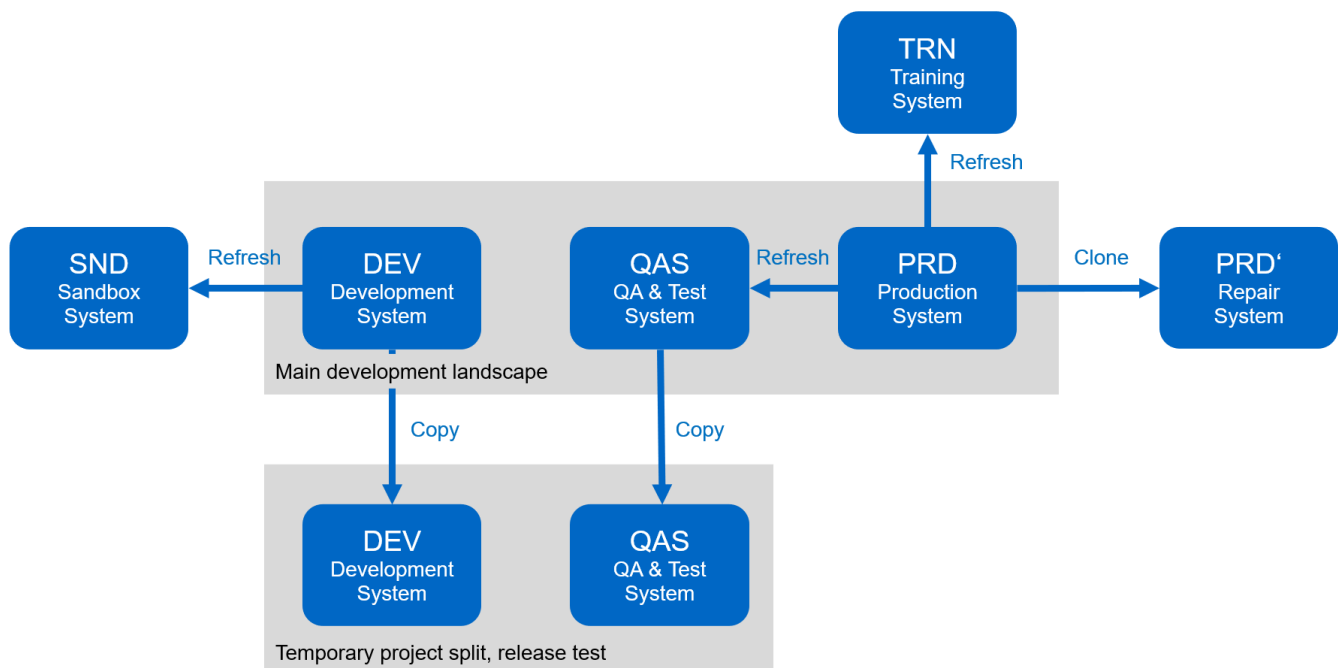
The following figure illustrates the main steps that must be performed during a system clone, system copy, or system refresh operation. The purple boxes indicate steps where NetApp storage features can be integrated. All three operations can be fully automated by using SAP LaMa.



Use cases for system refresh, copy, and cloning

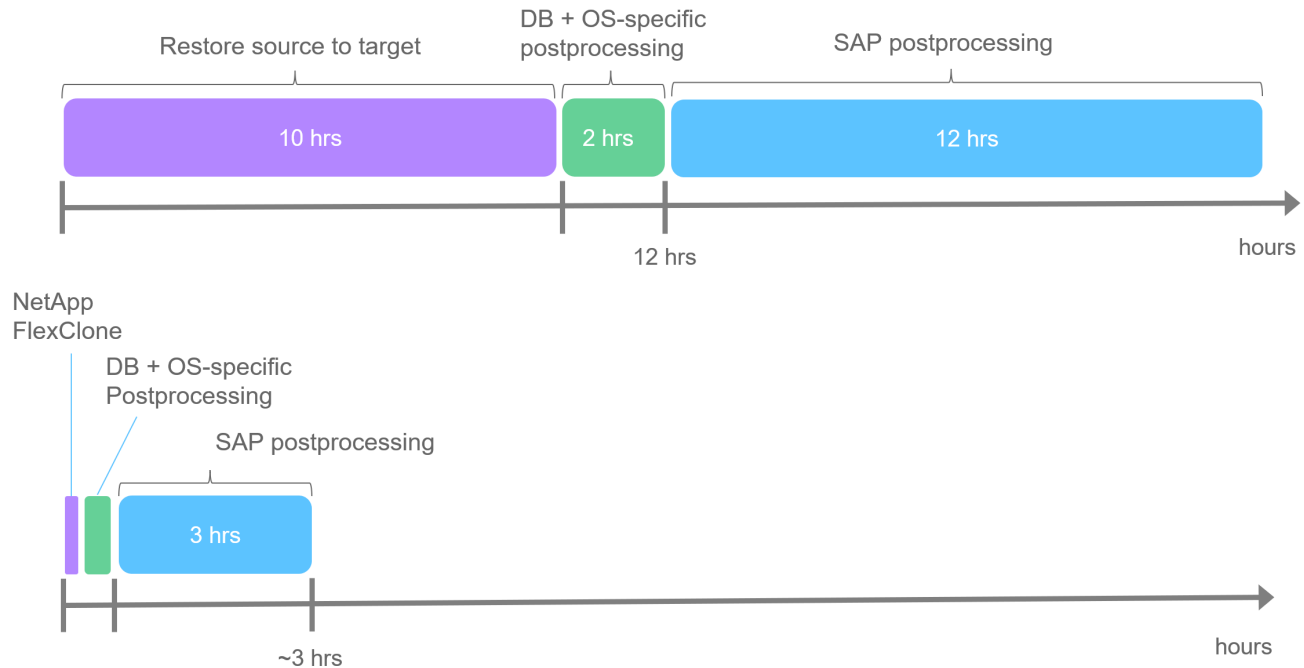
There are multiple scenarios in which data from a source system must be made available to a target system for testing or training purposes. These test and training systems must be updated with data from the source system on a regular basis to make sure that testing and training is performed with the current data set.

These system refresh operations consist of multiple tasks on the infrastructure, database, and application layers, and they can take multiple days depending on the level of automation.



SAP LaMa and NetApp cloning workflows can be used to accelerate and automate the required tasks at the

infrastructure and database layers. Instead of restoring a backup from the source system to the target system, SAP LaMa uses NetApp Snapshot copy and NetApp FlexClone technology so that required tasks up to a started HANA database can be performed in minutes instead of hours as shown in the following figure. The time needed for the cloning process is independent from the size of the database; therefore even very large systems can be created in a couple of minutes. Further reduction of the runtime is accomplished by automating tasks on the operating system and database layer as well as on the SAP post processing side.



Address logical corruption

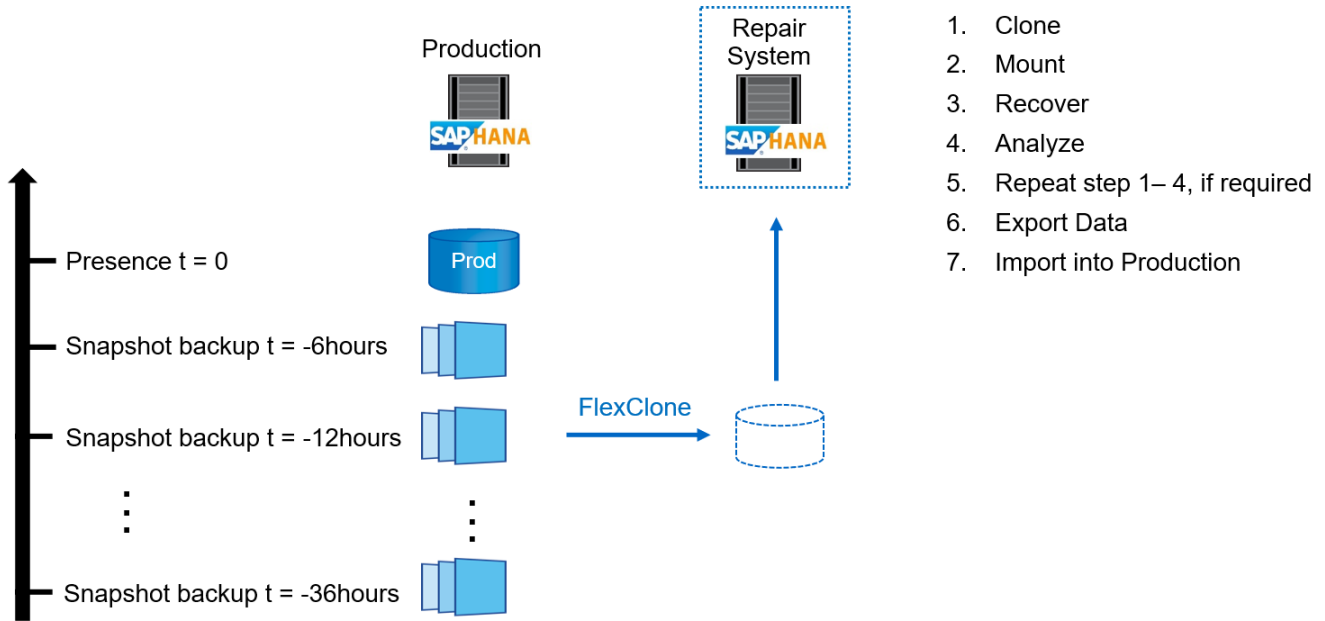
Logical corruption can be caused by software errors, human errors, or sabotage. Unfortunately, logical corruption often cannot be addressed with standard high-availability and disaster recovery solutions. As a result, depending on the layer, application, file system, or storage where the logical corruption occurred, minimal downtime and acceptable data loss requirements can sometimes not be fulfilled.

The worst case is logical corruption in an SAP application. SAP applications often operate in a landscape in which different applications communicate with each other and exchange data. Therefore, restoring and recovering an SAP system in which a logical corruption has occurred is not the recommended approach. Restoring the system to a point in time before the corruption occurred results in data loss. Also, the SAP landscape would no longer be in sync and would require additional postprocessing.

Instead of restoring the SAP system, the better approach is to try to fix the logical error within the system by analyzing the problem in a separate repair system. Root cause analysis requires the involvement of the business process and application owner. For this scenario, you create a repair system (a clone of the production system) based on data stored before the logical corruption occurred. Within the repair system, the required data can be exported and imported into the production system. With this approach, the production system does not need to be stopped, and, in the best-case scenario, no data or only a small fraction of data is lost.

When setting up the repair system, flexibility and speed are crucial. With NetApp storage-based Snapshot backups, multiple consistent database images are available to create a clone of the production system by using NetApp FlexClone technology. FlexClone volumes can be created in a matter of seconds rather than

multiple hours if a redirected restore from a file-based backup is used to set up the repair system.

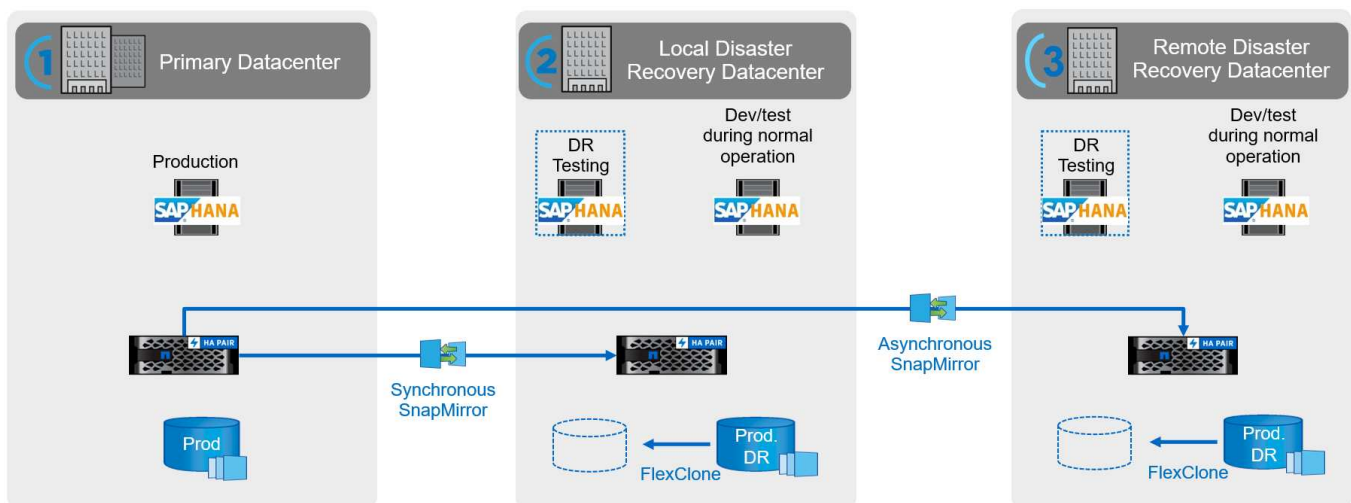


Disaster recovery testing

An effective disaster recovery strategy requires testing the required workflow. Testing demonstrates whether the strategy works and whether the internal documentation is sufficient. It also allows administrators to train on the required procedures.

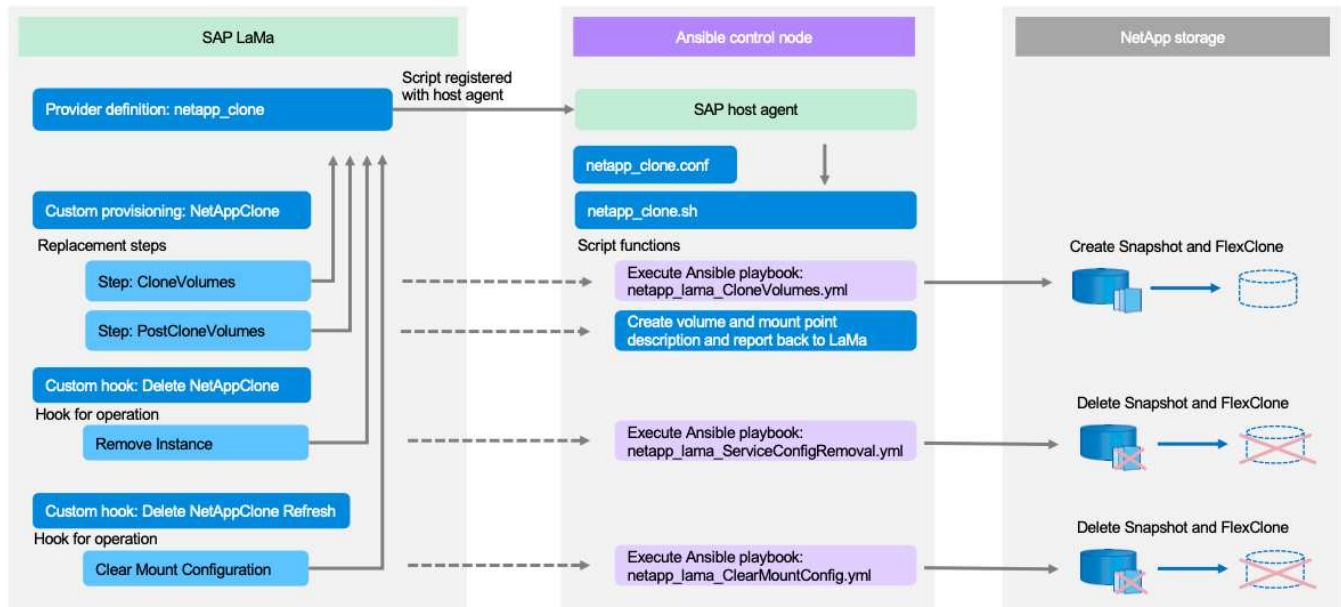
Storage replication with SnapMirror makes it possible to execute disaster recovery testing without putting RTO and RPO at risk. Disaster recovery testing can be performed without interrupting data replication. Disaster recovery testing for both asynchronous and synchronous SnapMirror uses Snapshot backups and FlexClone volumes at the disaster recovery target.

SAP LaMa can be used to orchestrate the entire testing procedure, and it also takes care of network fencing, target host maintenance, and so on.



NetApp SAP LaMa integration using Ansible

The integration approach uses SAP LaMa custom provisioning and operation hooks combined with Ansible playbooks for NetApp storage management. The following figure shows a high-level overview of the configuration on the LaMa side as well as the corresponding components of the example implementation.



A central host acting as an Ansible control node is used to execute the requests from SAP LaMa and to trigger the NetApp storage operations using Ansible playbooks. The SAP host agent components must be installed on this host so that the host can be used as a communication gateway to SAP LaMa.

Within LaMa Automation Studio, a provider is defined that is registered at the Ansible host's SAP host agent. A host agent configuration file points to a shell script that is called by SAP LaMa with a set of command line parameters, depending on the requested operation.

Within LaMa Automation Studio, custom provisioning and a custom hook is defined to execute storage cloning operations during provisioning and also during clean-up operations when the system is deprovisioned. The shell script on the Ansible control node then executes the corresponding Ansible playbooks, which trigger the Snapshot and FlexClone operations as well as the deletion of the clones with the deprovisioning workflow.

More information on NetApp Ansible modules and the LaMa provider definitions can be found at:

- [NetApp Ansible modules](#)
- [SAP LaMa documentation – provider definitions](#)

Example implementation

Due to the large number of options available for system and storage setups, the example implementation should be used as a template your individual system setup and configuration requirements.



The example scripts are provided as is and are not supported by NetApp. You can request the current version of the scripts via email to ng-sapcc@netapp.com.

Validated configurations and limitations

The following principles were applied to the example implementation and might need to be adapted to meet customer needs:

- Managed SAP systems used NFS to access NetApp storage volumes and were set up based on the adaptive design principle.
- You can use all ONTAP releases supported by NetApp Ansible modules (ZAPI and REST API).
- Credentials for a single NetApp cluster and SVM were hard coded as variables in the provider script.
- Storage cloning was performed on the same storage system that was used by the source SAP system.
- Storage volumes for the target SAP system had the same names as the source with an appendix.
- No cloning at secondary storage (SV/SM) was implemented.
- FlexClone split was not implemented.
- Instance numbers were identical for the source and target SAP systems.

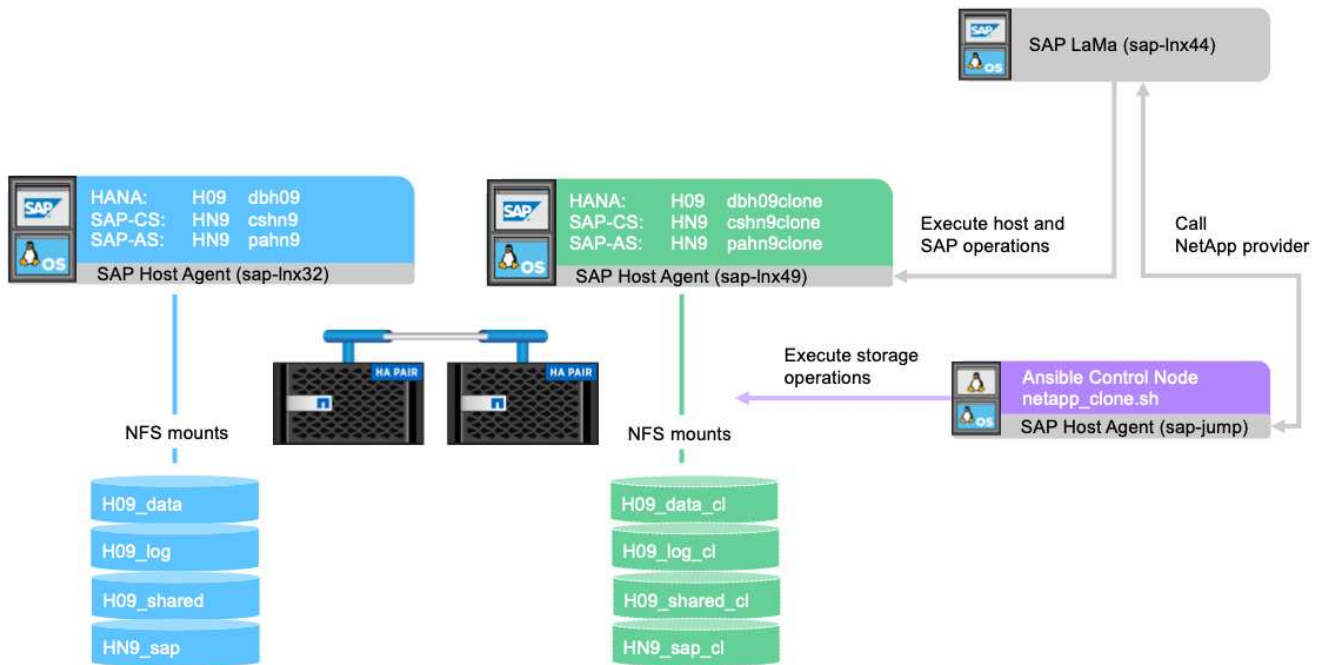
Lab setup

The following figure shows the lab setup we used. The source SAP system HN9 used for the system clone operation consisted of the database H09, the SAP CS, and the SAP AS services running on the same host (sap-lnx32) with installed [adaptive design](#) enabled. An Ansible control node was prepared according to the [Ansible Playbooks for NetApp ONTAP](#) documentation.

The SAP host agent was installed on this host as well. The NetApp provider script as well as the Ansible playbooks were configured on the Ansible control node as described in the "[Appendix: Provider Script Configuration](#)."

The host `sap-lnx49` was used as the target for the SAP LaMa cloning operations, and the isolation-ready feature was configured there.

Different SAP systems (HNA as source and HN2 as target) were used for system copy and refresh operations, because Post Copy Automation (PCA) was enabled there.



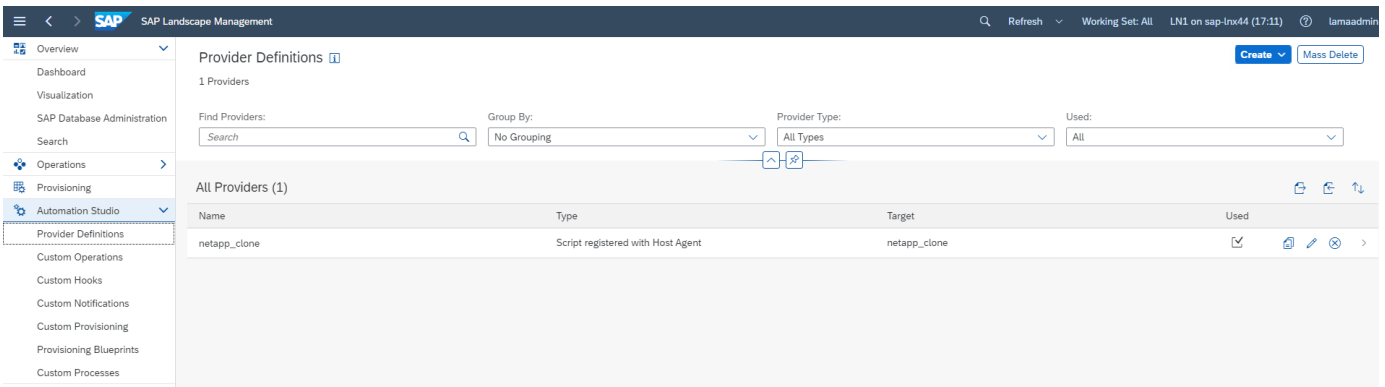
The following software releases were used in the lab setup:

- SAP LaMa Enterprise Edition 3.00 SP23_2
- SAP HANA 2.00.052.00.1599235305
- SAP 7.77 Patch 27 (S/4 HANA 1909)
- SAP Host Agent 7.22 Patch 56
- SAPACEXT 7.22 Patch 69
- Linux SLES 15 SP2
- Ansible 2.13.7
- NetApp ONTAP 9.8P8

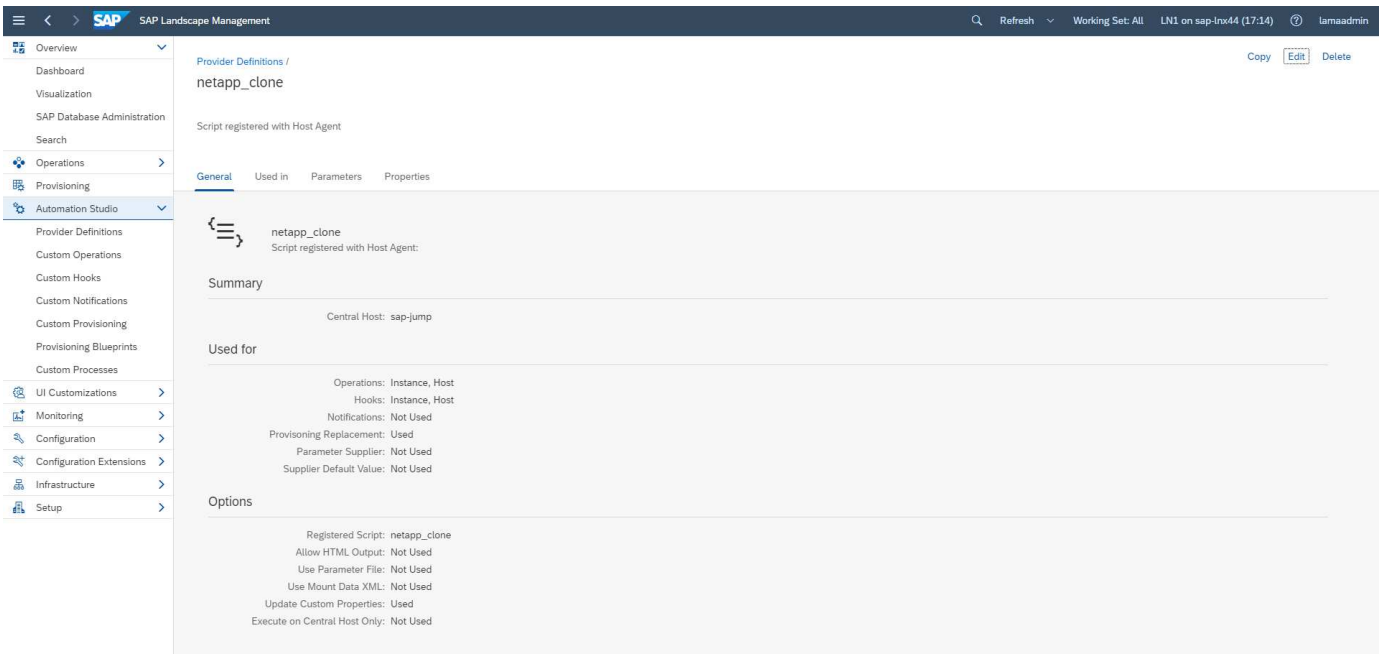
SAP LaMa configuration

SAP LaMa provider definition

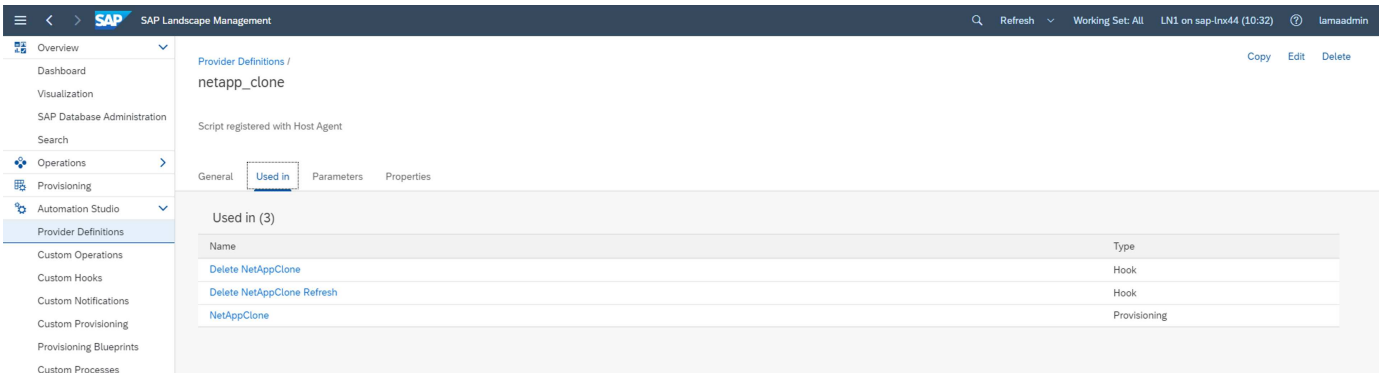
The provider definition is performed within Automation Studio of SAP LaMa as shown in the following screenshot. The example implementation uses a single provider definition that is used for different custom provisioning steps and operation hooks as explained before.



The provider `netapp_clone` is defined as the script `netapp_clone.sh` registered at the SAP host agent. The SAP host agent runs on the central host `sap-jump`, which also acts as the Ansible control node.



The **Used in** tab shows which custom operations the provider is used for. The configuration for the custom provisioning **NetAppClone** and the custom hooks **Delete NetAppClone** and **Delete NetAppClone Refresh** are shown in the next chapters.



The parameters **ClonePostFix** and **SnapPostFix** are requested during the execution of the provisioning workflow and are used for the Snapshot and FlexClone volume names.

The screenshot shows the 'netapp_clone' provider definition in SAP Landscape Management. The 'Parameters' tab is active, displaying a table of parameters:

Name	Label	Type	Value	Mandatory	Secure	Multivalue	
ClonePostFix	ClonePostFix	String		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
SnapPostFix	SnapPostFix	String		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

SAP LaMa custom provisioning

In the SAP LaMa custom provisioning configuration, the customer provider described before is used to replace the provisioning workflow steps **Clone Volumes** and **PostCloneVolumes**.

The screenshot shows the 'Custom Provisioning' section in SAP Landscape Management. It displays a table of custom provisioning processes:

Name	Provider Parameters	Instance Type	
CloneVolumes			
Clone Volumes	netapp_clone	Default (all unused instance types)	
FinalizeCloneVolumes			
Modify Mountpoints and add Custom Properties	netapp_clone	Default (all unused instance types)	

SAP LaMa custom hook

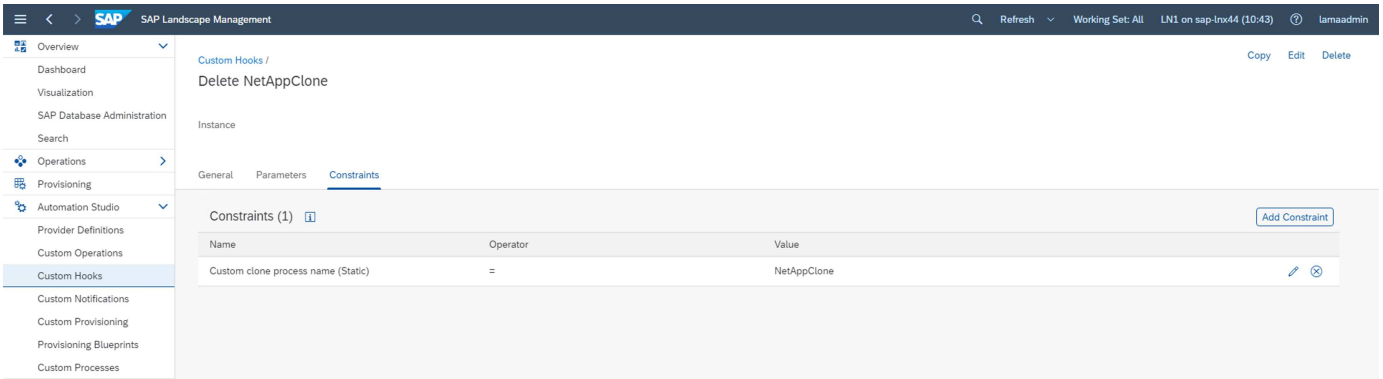
If a system is deleted with the system destroy workflow, the hook **Delete NetAppClone** is used to call the provider definition `netapp_clone`. The **Delete NetApp Clone Refresh** hook is used during the system refresh workflow because the instance is preserved during the execution.

The screenshot shows the 'Custom Hooks' section in SAP Landscape Management. It displays a table of custom hooks:

Name	Entity Type	Provider	Type	
Delete NetAppClone Refresh	Instance	netapp_clone	Pre hook for 'Clear Mount Configuration'	
Delete NetAppClone	Instance	netapp_clone	Pre hook for 'Remove Instance'	

It is important to configure **Use Mount Data XML** for the custom hook, so that SAP LaMa provides the information of the mount point configuration to the provider.

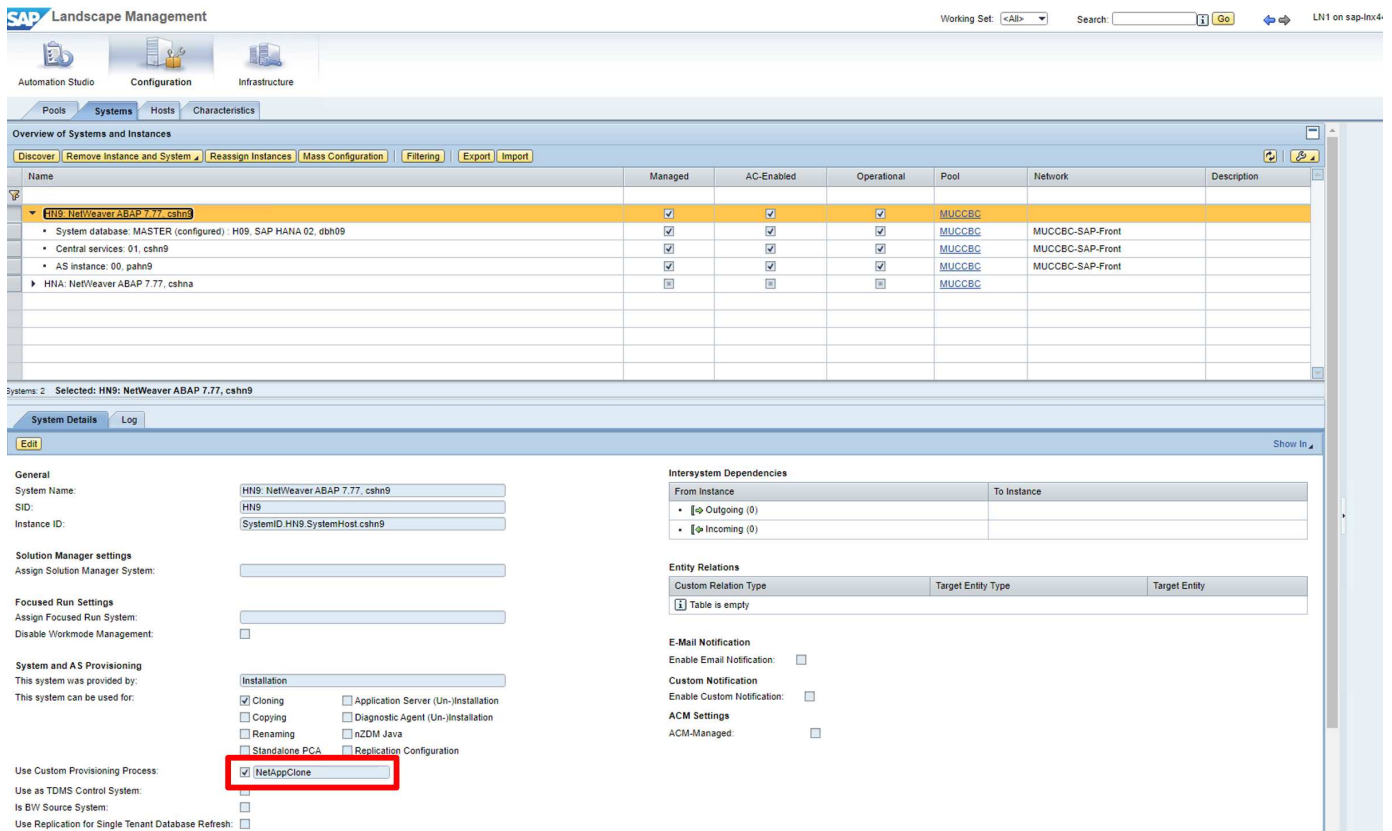
To ensure that the custom hook is only used and executed when the system was created with a custom provisioning workflow, the following constraint is added to it.



More information about the use of custom hooks can be found in the [SAP LaMa Documentation](#).

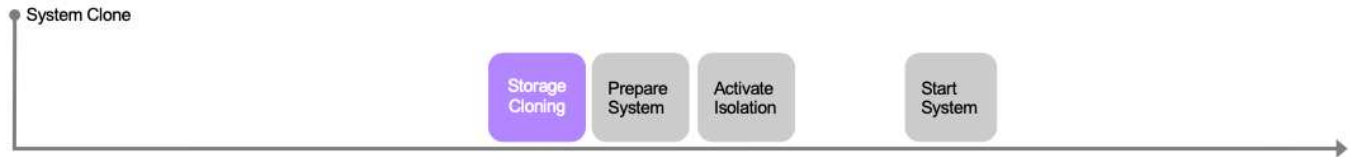
Enable custom provisioning workflow for SAP source system

To enable the custom provisioning workflow for the source system, it must be adapted in the configuration. The **Use Custom Provisioning Process** checkbox with the corresponding custom provisioning definition must be selected.

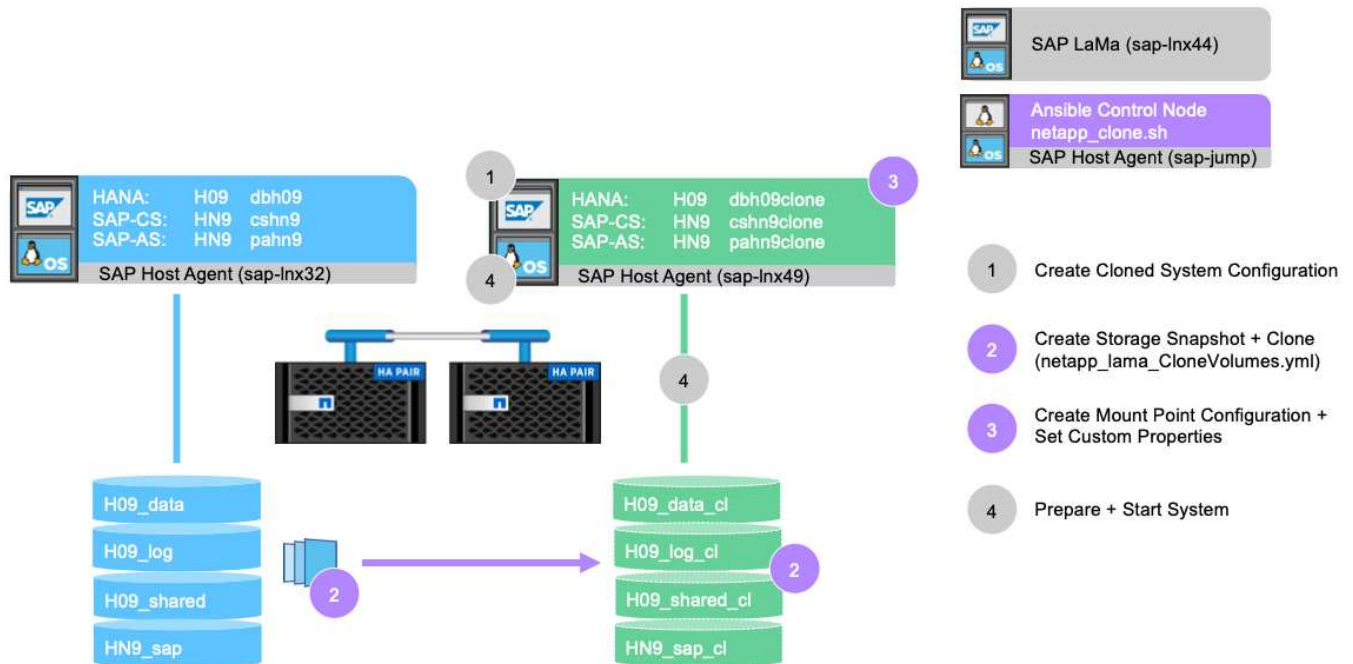


SAP LaMa provisioning workflow - clone system

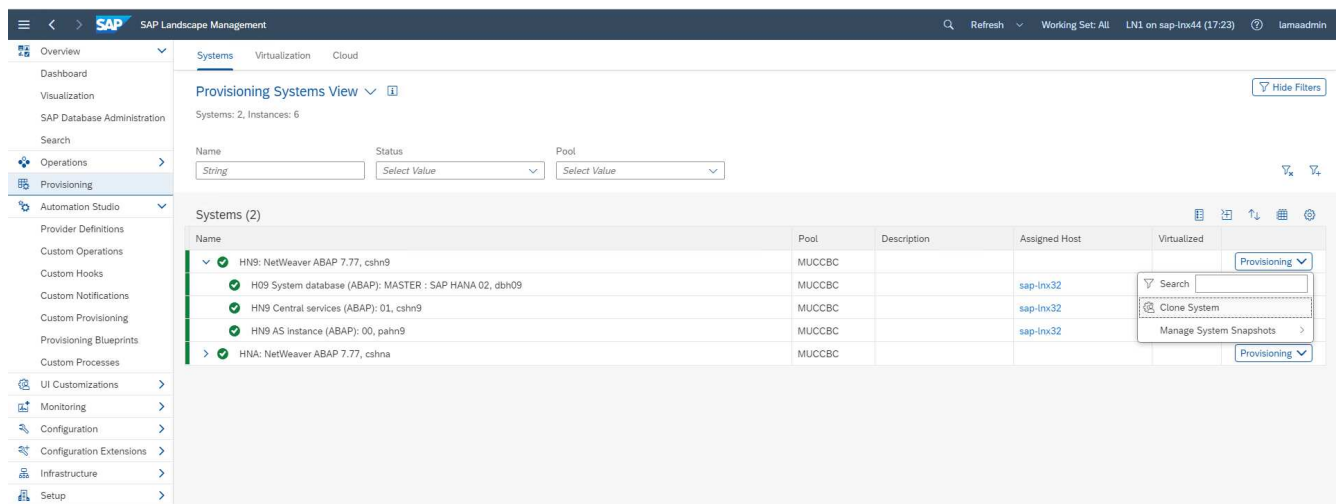
The following figure highlights the main steps executed with the system clone workflow.



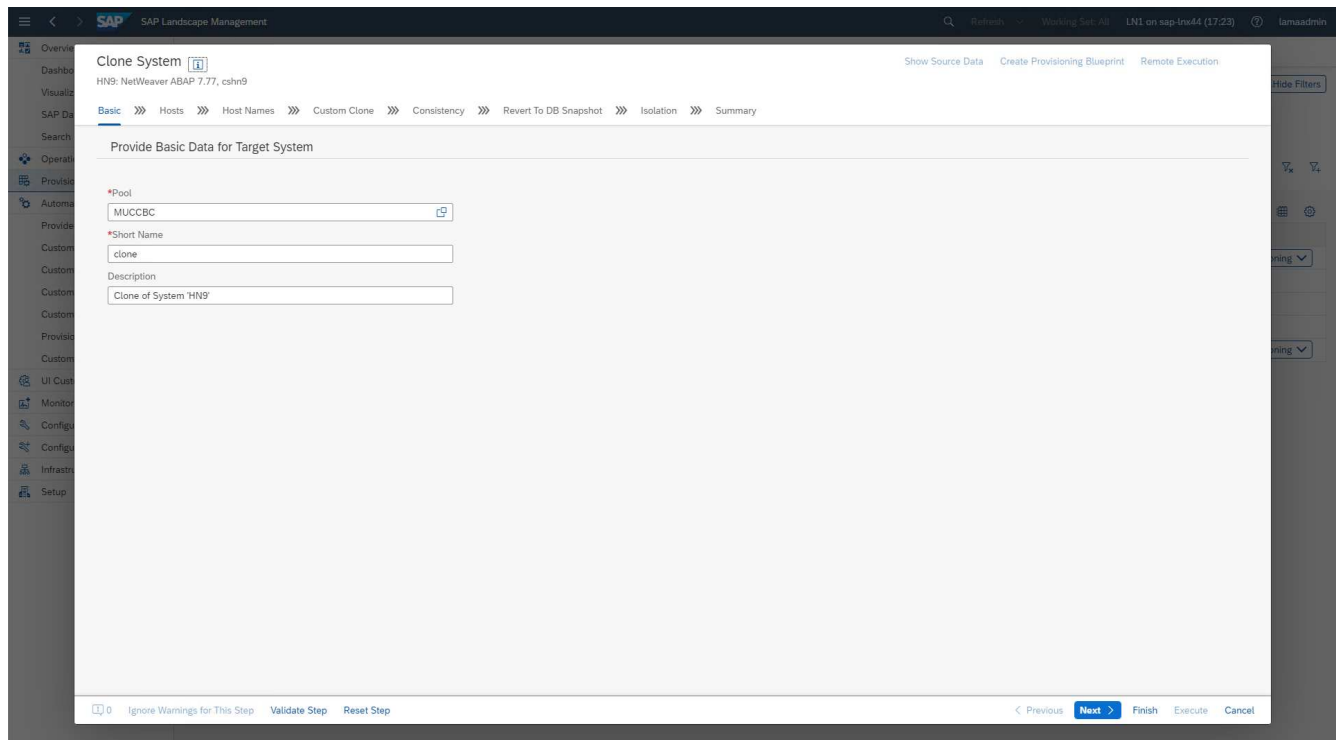
In this section, we go through the complete SAP LaMa system cloning workflow based on the source SAP system HN9 with HANA database H09. The following picture gives an overview of the steps executed during the workflow.



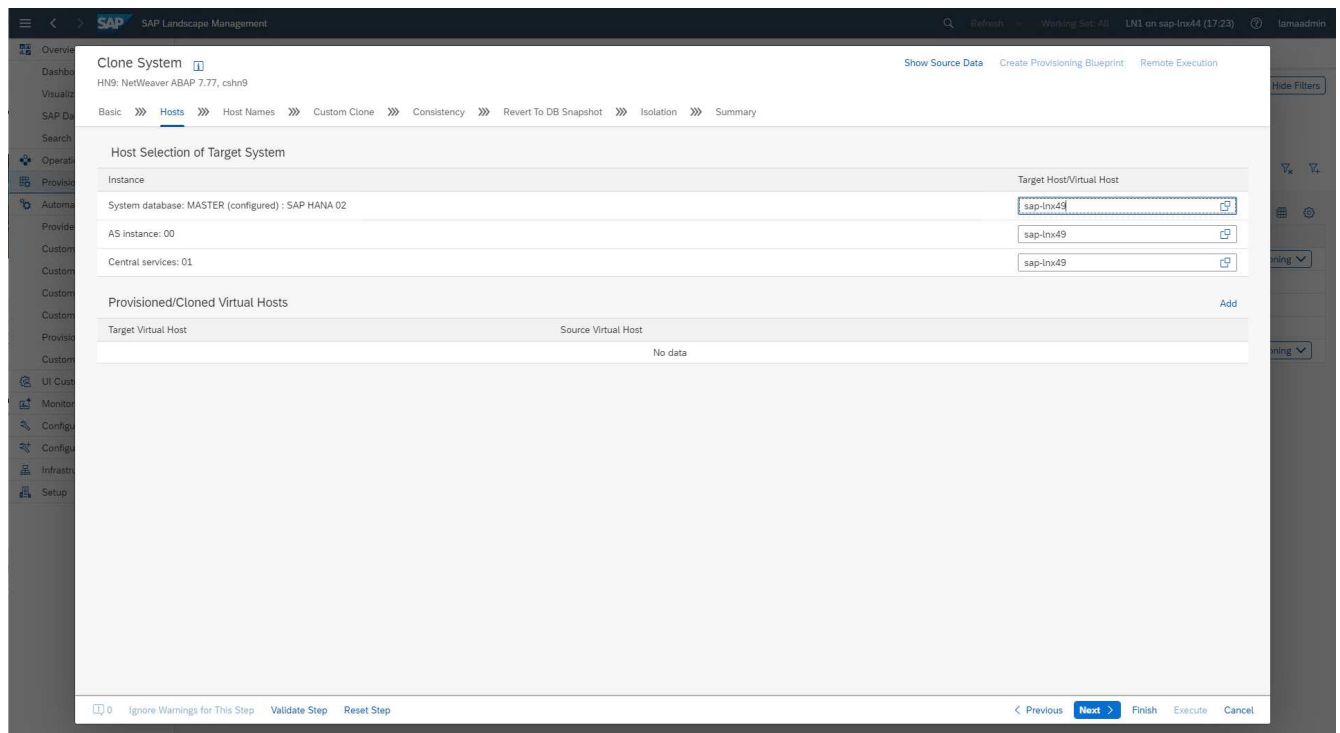
1. To start the cloning workflow, open **Provisioning** in the menu tree and select the source system (in our example HN9). Then start the **Clone System** wizard.



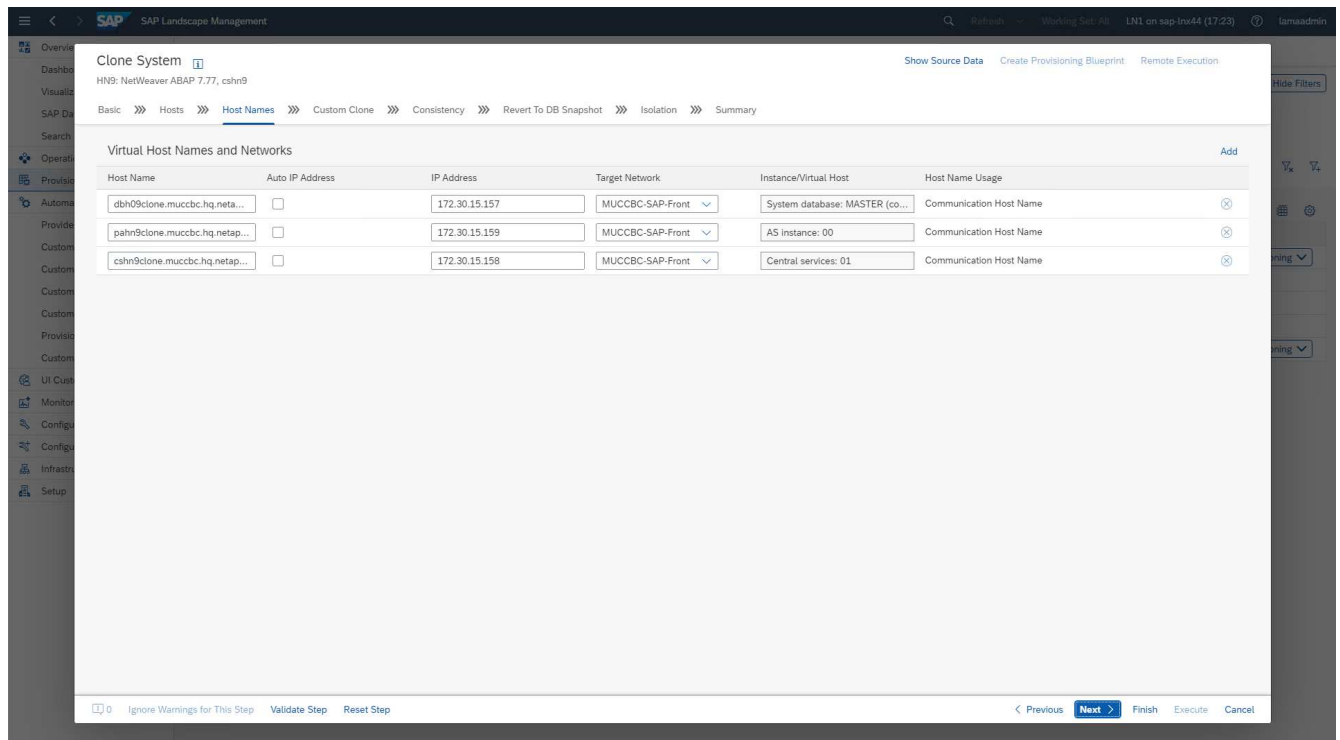
2. Enter the requested values. Screen 1 of the wizard asks for the pool name for the cloned system. This step specifies the instances (virtual or physical) on which the cloned system will be started. The default is to clone the system into the same pool as the target system.



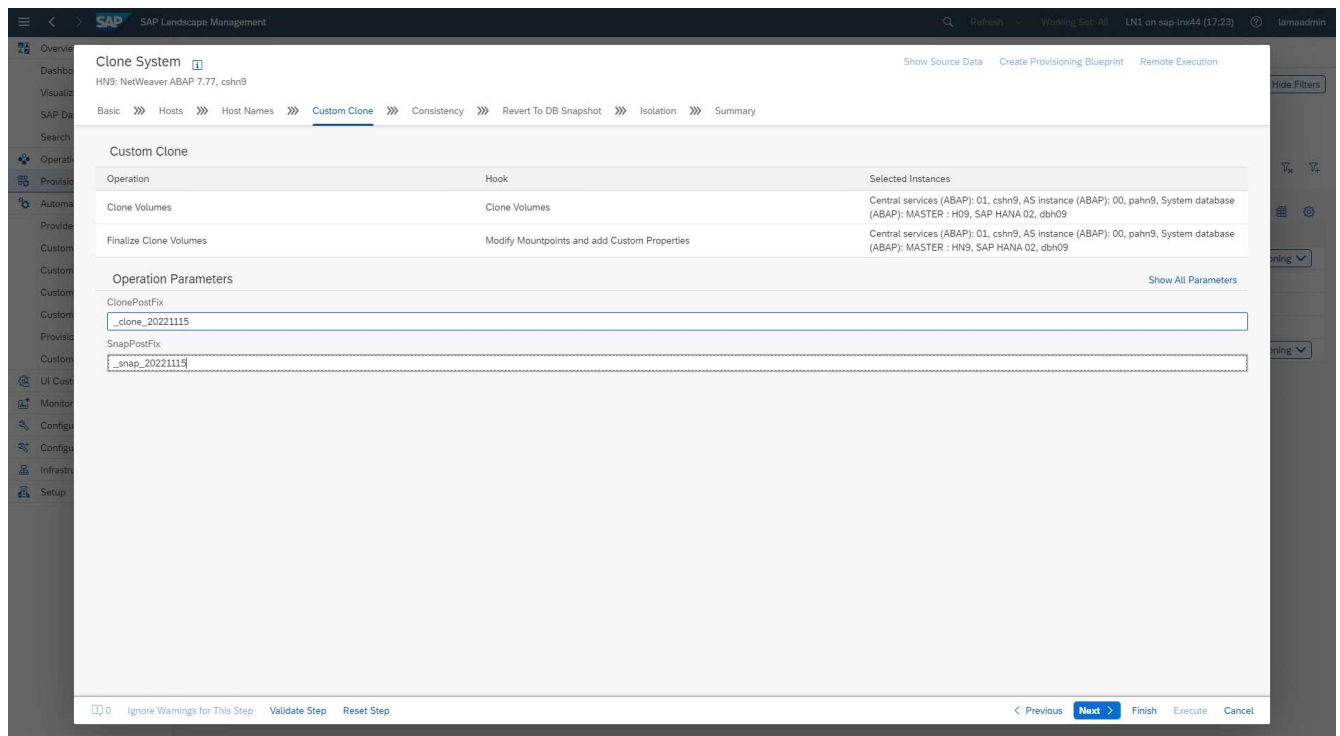
- Screen 2 of the wizard asks for the target hosts that the new SAP instances are started on. The target hosts for this instance(s) can be selected out of the host pool specified in the previous screen. Each instance or service can be started on a different host. In our example, all three services run on the same host.



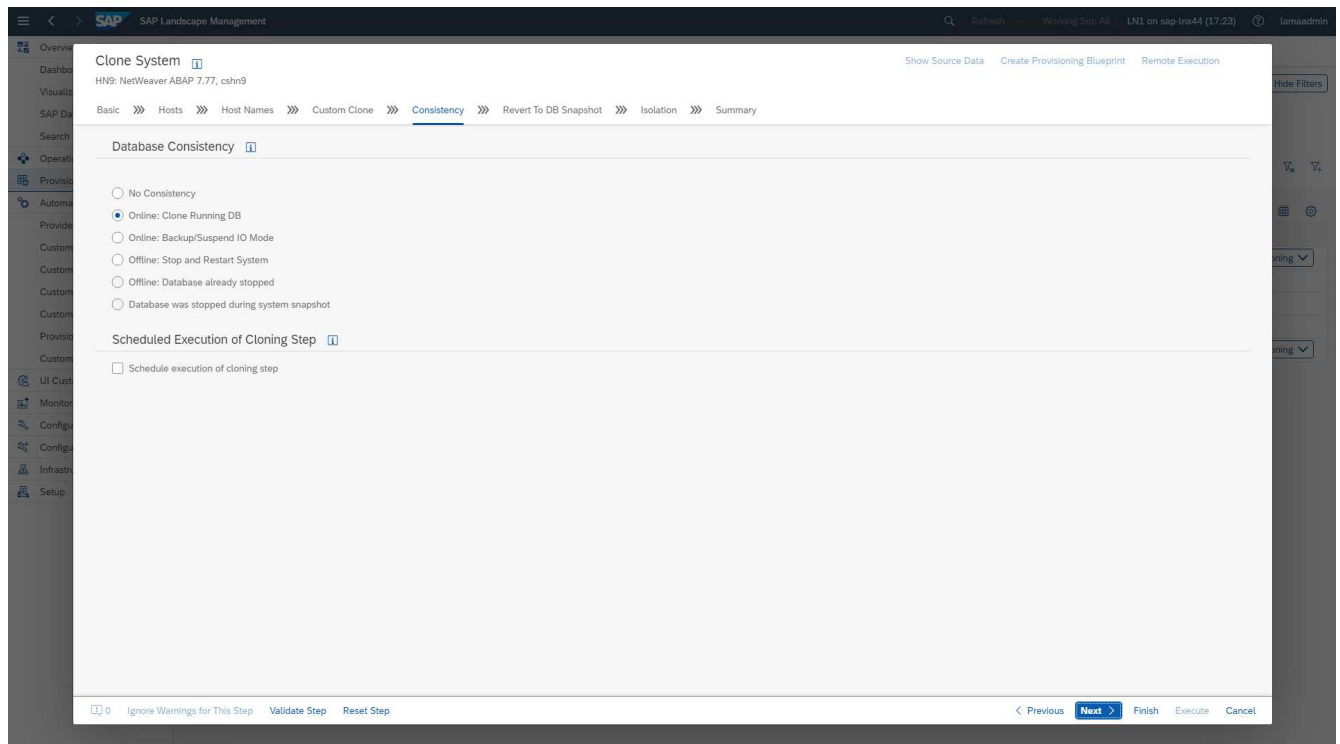
- Provide the information requested in screen 3, which asks for virtual host names and networks. Typically, the host names are maintained in DNS, so the IP addresses are prepopulated accordingly.



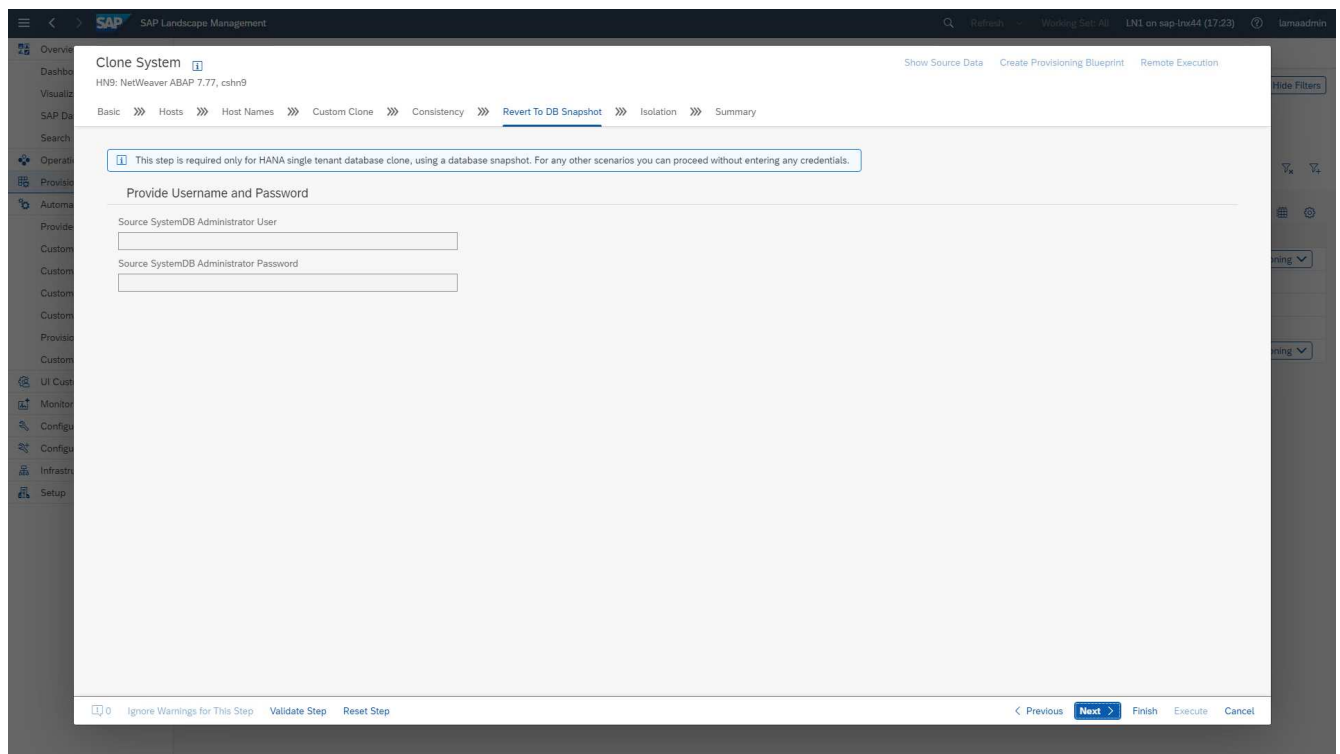
5. In screen 4, the custom clone operations are listed. A clone and a **SnapPostfix** name are provided, which are used during the storage clone operation for the FlexClone volume and Snapshot name, respectively. If you leave these fields empty, the default value configured in the variable section of the provider script `netapp_clone.sh` is used.



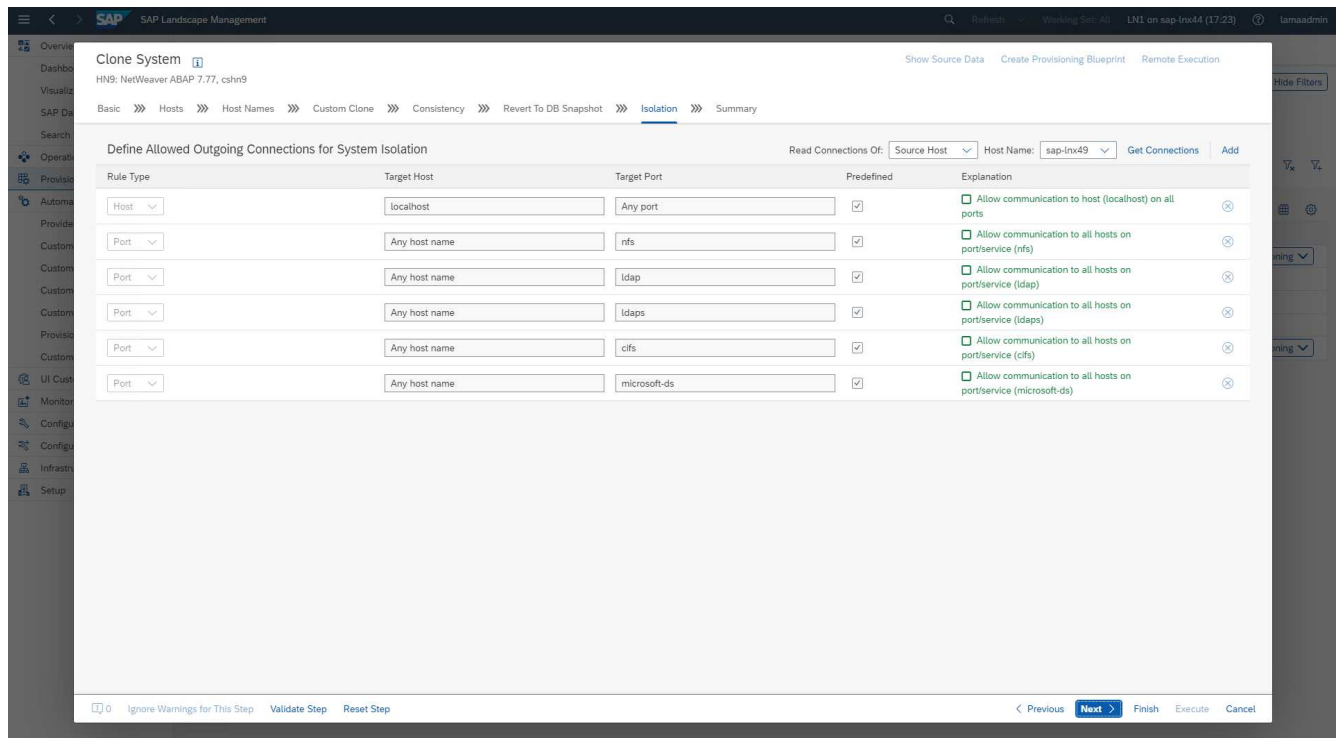
6. In screen 5, the database consistency option is selected. In our example, we selected **Online: Clone running DB**.



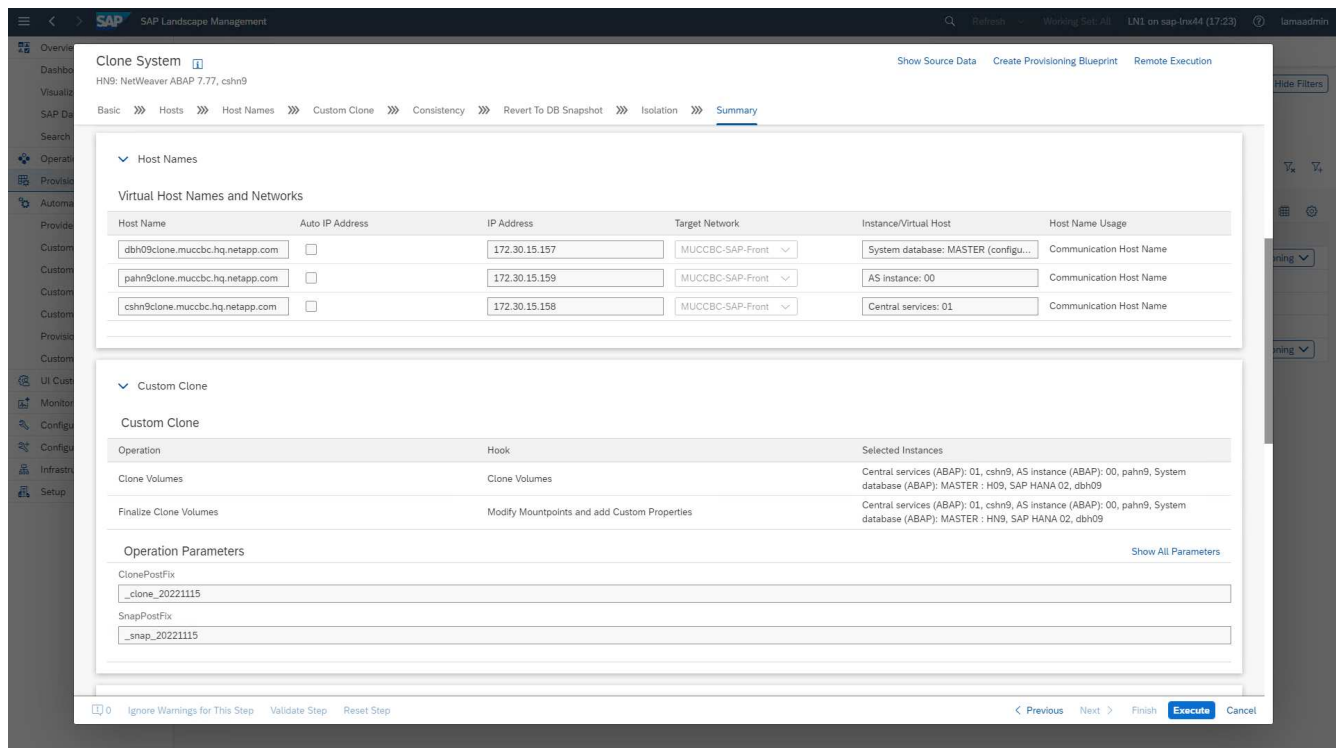
7. In screen 6, input is only required if you perform a tenant clone.



8. In screen 7, system isolation can be configured.



9. In screen 8, a summary page contains all the settings for final confirmation before the workflow is started. Click **Execute** to start the workflow.



SAP LaMa now performs all the actions indicated in the configuration. These actions include creating the storage volume clones and exports, mounting them to the target host, adding the firewall rules for isolation, and starting the HANA database and SAP services.

10. You can monitor the progress of the clone workflow under the **Monitoring** menu.

Within the detailed log, the operations **Clone Volume** and **Modify Mountpoints and add Custom Properties** are executed at the Ansible node, the `sap-jump` host. These steps are executed for each service, the HANA database, the SAP central services, and the SAP AS service.

- By selecting the **Clone Volumes** task the detailed log for that step is displayed and the execution of the Ansible Playbook is shown here. You can see, that the Ansible playbook `netapp_lama_CloneVolumes.yml` is executed for each HANA database volume, data, log, and shared.

The screenshot displays the SAP Landscape Management interface. The 'Clone Volumes' step is selected, showing a list of messages. A red box highlights the following messages:

- DEBUG** ID: 39 | Message Code: NetApp Clone for Custom Provis
Time: 2022-11-15 17:29:40 | Entry Time: 0:17
Running ansible playbook netapp_jama_CloneVolumes.yml on Volume H09_shared
- DEBUG** ID: 31 | Message Code: NetApp Clone for Custom Provis
Time: 2022-11-15 17:29:40 | Entry Time: 0:17
Running ansible playbook netapp_jama_CloneVolumes.yml on Volume H09_log
- DEBUG** ID: 23 | Message Code: NetApp Clone for Custom Provis
Time: 2022-11-15 17:29:40 | Entry Time: 0:17
Running ansible playbook netapp_jama_CloneVolumes.yml on Volume H09_data

12. In the details view of the step **Modify Mountpoints and add Custom Properties**, you can find information about the mount points and the custom properties handed over by the execution script.

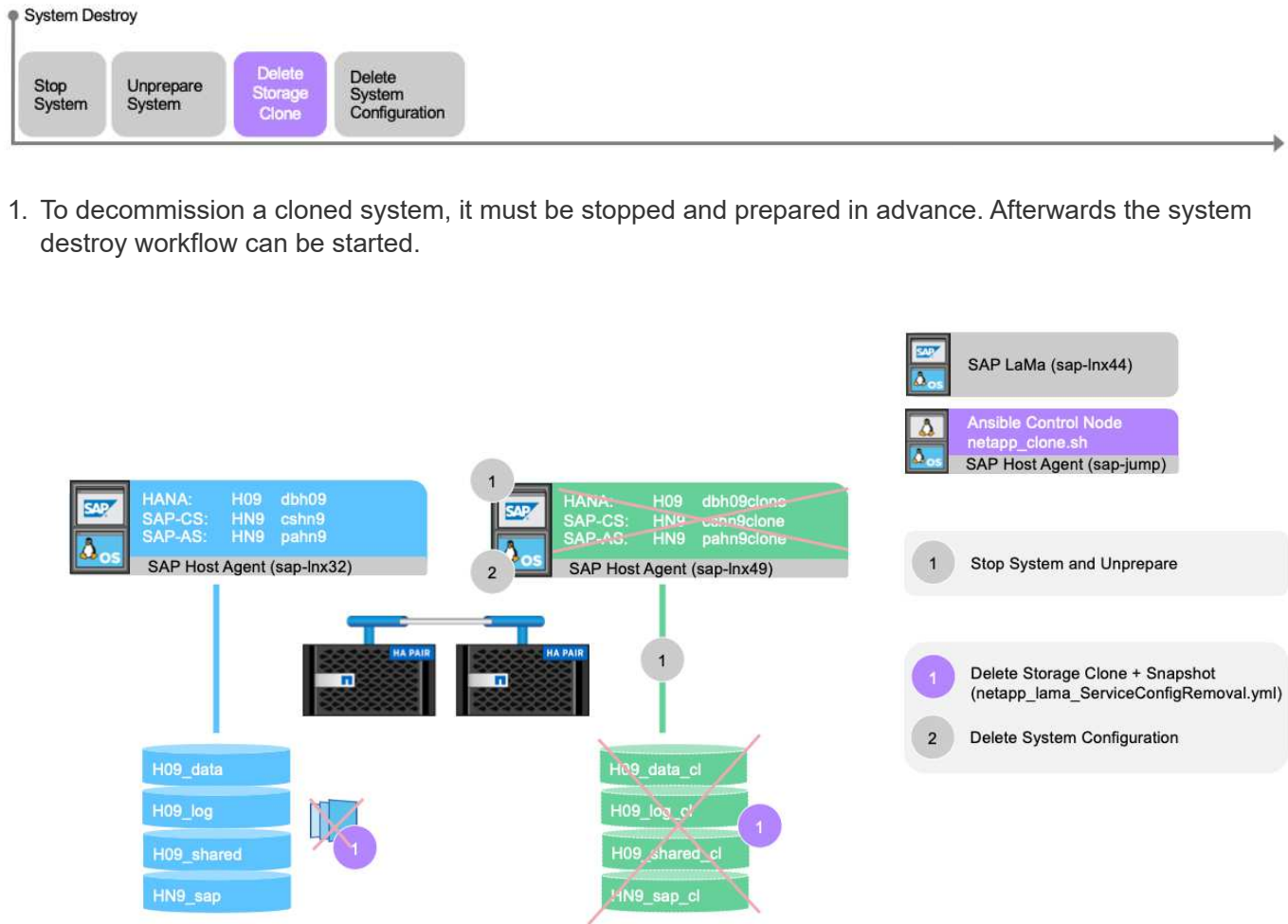
The screenshot displays the SAP Landscape Management interface. The 'Modify Mountpoints and add Custom Properties' step is selected, showing a list of messages. A red box highlights the following messages:

- RESULT** ID: 24 | Message Code: NetApp Clone for Custom Provis
Time: 2022-11-15 17:30:20 | Entry Time: 0:18
Got new property SnapPostFix_snap_20221115
- RESULT** ID: 23 | Message Code: NetApp Clone for Custom Provis
Time: 2022-11-15 17:30:20 | Entry Time: 0:18
Got new property ClonePostFix_clone_20221115

After the workflow has been completed, the cloned SAP system is prepared, started, and ready for use.

SAP LaMa deprovisioning workflow - system destroy

The following figure highlights the main steps executed with the system destroy workflow.



2. In this example, we run the system destroy workflow for the system created before. We select the system in the **System View** screen and start the system destroy workflow under **Destroy Processes**.

3. All the mount points maintained during the provisioning phase are shown here and are deleted during the system destroy workflow process.

SAP Landscape Management

Destroy System HN9: NetWeaver ABAP 7.77, dbh09clone.muccbc.hq.netapp.com

Delete Storage Volumes »»» Delete Host Names »»» Summary

Storage Volumes

Delete	Volume	Storage Manager	Storage System	Storage Pool	Volume Group	Latest Monitoring Time
No data						

Mount Data Without Corresponding Storage Volume

Instance	Storage Type	Export Path	Mount Point	Mount Options
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/hn9a...	/home/hn9adm	rw,noatime,vers=3,rsize=65536,wsiz...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sapmnt	/sapmnt/HN9	rw,noatime,vers=3,rsize=65536,wsiz...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/HN9	/usr/sap/HN9	rw,noatime,vers=3,rsize=65536,wsiz...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/ccms	/usr/sap/ccms/HN9_00	rw,noatime,vers=3,rsize=65536,wsiz...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/saptr...	/usr/sap/trans	rw,noatime,vers=3,rsize=65536,wsiz...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_data_clone_20221115/data	/hana/data/H09	rw,noatime,vers=3,rsize=65536,wsiz...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_log_clone_20221115/log	/hana/log/H09	rw,noatime,vers=3,rsize=65536,wsiz...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_shared_clone_20221115/sh...	/hana/shared/H09	rw,noatime,vers=3,rsize=65536,wsiz...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/hn9a...	/home/hn9adm	rw,noatime,vers=3,rsize=65536,wsiz...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sapmnt	/sapmnt/HN9	rw,noatime,vers=3,rsize=65536,wsiz...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/HN9	/usr/sap/HN9	rw,noatime,vers=3,rsize=65536,wsiz...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/ccms	/usr/sap/ccms/HN9_00	rw,noatime,vers=3,rsize=65536,wsiz...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/saptr...	/usr/sap/trans	rw,noatime,vers=3,rsize=65536,wsiz...

Monitoring Time: [Monitoring Data](#)

[Ignore Warnings for This Step](#) [Validate Step](#) [Reset Step](#) [Previous](#) [Next](#) [Finish](#) [Execute](#) [Cancel](#)

No virtual hostnames are deleted because they are maintained through DNS and have been assigned automatically.

SAP Landscape Management

Destroy System HN9: NetWeaver ABAP 7.77, dbh09clone.muccbc.hq.netapp.com

Delete Storage Volumes »»» Delete Host Names »»» Summary

Host Names

Delete	DNS Server	Host Name	IP Address
No data			

[Ignore Warnings for This Step](#) [Validate Step](#) [Reset Step](#) [Previous](#) [Next](#) [Finish](#) [Execute](#) [Cancel](#)

4. The operation is started by clicking the execute button.

Destroy System 🔍

HN9: NetWeaver ABAP 7.77, dbh09clone.muccbc.hq.netapp.com

Show Source Data Create Provisioning Blueprint Remote Execution

Delete Storage Volumes >> Delete Host Names >>> **Summary**

🔍 SAP advises that it is the customer's responsibility to ensure that no data is lost when the selected volumes/virtual hosts are deleted by SAP Landscape Management.

▼ Delete Storage Volumes

Storage Volumes

Delete	Volume	Storage Manager	Storage System	Storage Pool	Volume Group	Latest Monitoring Time
No data						

Mount Data Without Corresponding Storage Volume

Instance	Storage Type	Export Path	Mount Point	Mount Options
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/hn9...	/home/hn9adm	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sap...	/sapmnt/HN9	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/HN9	/usr/sap/HN9	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/ccms	/usr/sap/ccms/HN9_00	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
AS instance: 00	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sapt...	/usr/sap/trans	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_data_clone_20221115/data	/hana/data/H09	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_log_clone_20221115/log	/hana/log/H09	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
System database: MASTER : H09, SAP HANA 02	NETFS	192.168.10.14:/H09_shared_clone_20221115/s...	/hana/shared/H09	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/hn9...	/home/hn9adm	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sap...	/sapmnt/HN9	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/HN9	/usr/sap/HN9	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/ccms	/usr/sap/ccms/HN9_00	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...
Central services: 01	NETFS	192.168.10.14:/HN9_sap_clone_20221115/sapt...	/usr/sap/trans	rw,noatime,vers=3,rsize=65536,wsiz=65536,n...

Monitoring Time: Monitoring Data

0 Ignore Warnings for This Step Validate Step Reset Step

< Previous Next > Finish **Execute** Cancel

SAP LaMa now performs the deletion of the volume clones and deletes the configuration of the cloned system.

5. You can monitor the progress of the clone workflow under the **Monitoring** menu.

SAP Landscape Management

Overview Dashboard Visualization SAP Database Administration Search

Operations Provisioning Automation Studio

Provider Definitions Custom Operations Custom Hooks Custom Notifications Custom Provisioning Provisioning Blueprints Custom Processes

UI Customizations

Monitoring Activities Logs Performance Configuration Configuration Extensions Infrastructure Setup

New view * 🔍 Mass Actions

Latest Server Time: 2022-11-15 17:52:54 (CET)

Name:

Status:

Activity Number:

Activities (1)

System destroy

Activity Number: 1861

Progress: 0%

Note:

Start Time: 2022-11-15 17:55:03

System destroy

Activity | Activity Number 1861

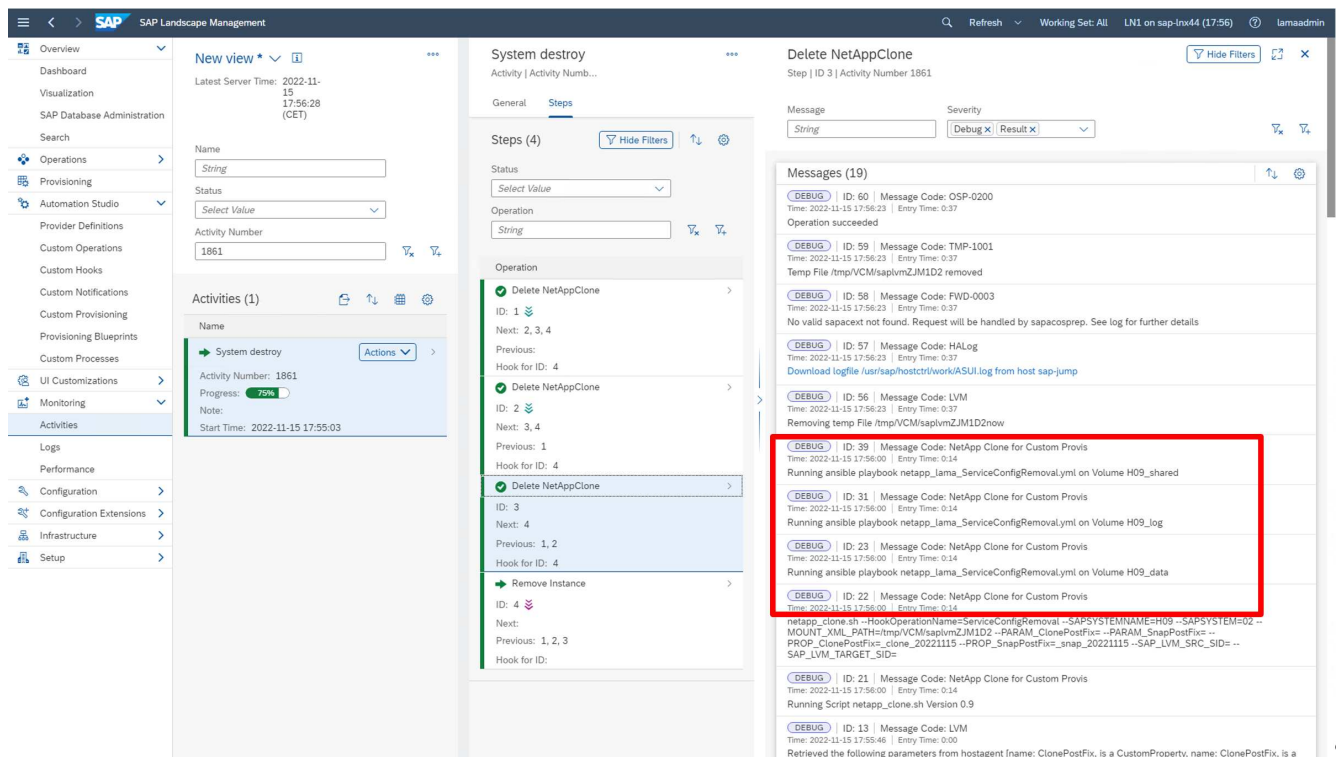
General Steps

Steps (4)

Status: Operation:

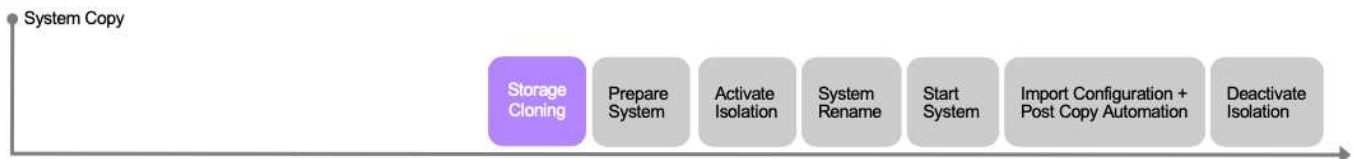
Operation	ID	Next	Previous	Hook for ID	Instance/Virtual Element	Host/Parent Virtual Element	Step Time	Duration
Delete NetAppClone	1	2, 3, 4		4	HN9 Central services (ABAP): 01, cshn9clone.muccbc.hq.netapp.com	sap-jump	0:00	0:11
Delete NetAppClone	2	3, 4	1	4	HN9 AS Instance (ABAP): 00, pah9clone.muccbc.hq.netapp.com	sap-jump		
Delete NetAppClone	3	4	1, 2	4	H09 System database (ABAP): MASTER : SAP HANA 02, dbh09clone.muccbc.hq.netapp.com	sap-jump		
Remove Instance	4		1, 2, 3		HN9: NetWeaver ABAP 7.77, dbh09clone.muccbc.hq.netapp.com			

6. By selecting the **Delete NetAppClone** task, the detailed log for that step is displayed. The execution of the Ansible Playbook is shown here. As you can see, the Ansible playbook `netapp_lama_ServiceConfigRemoval.yml` is executed for each HANA database volume, data, log, and shared.

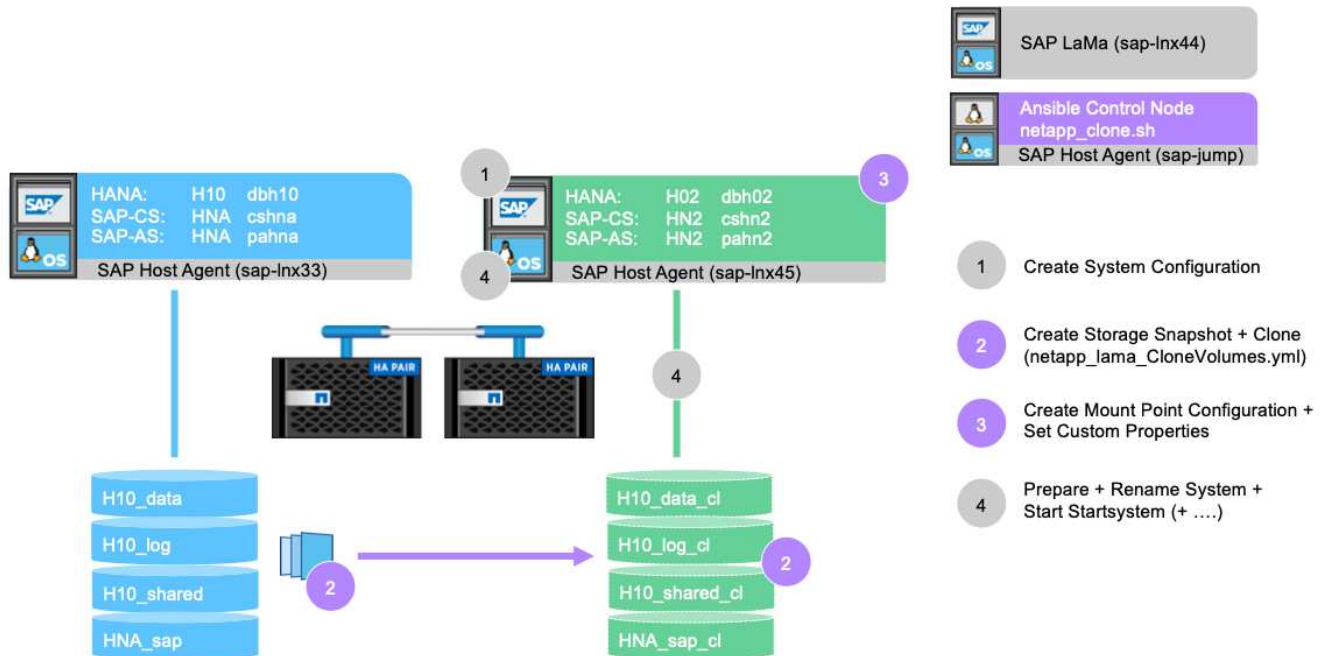


SAP LaMa provisioning workflow - copy system

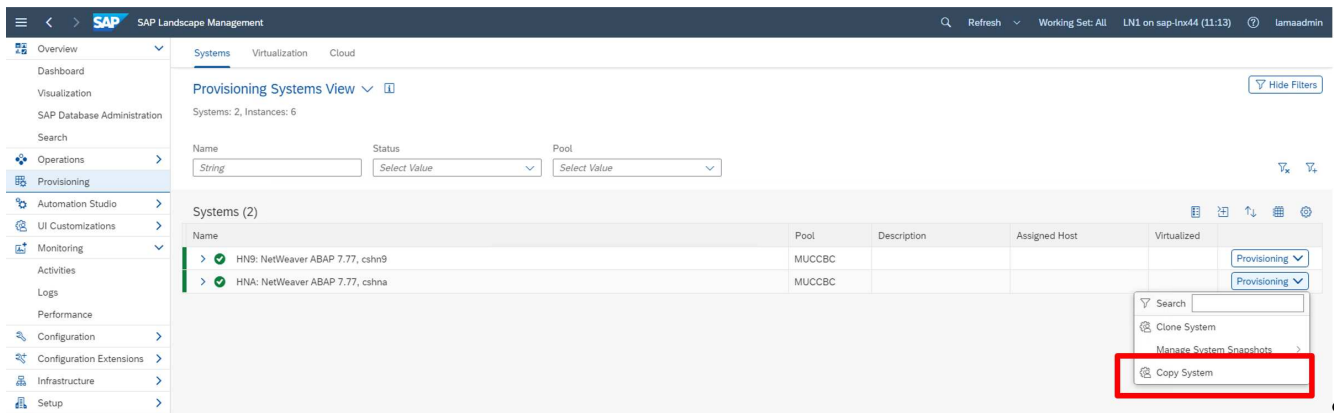
The following figure highlights the primary steps executed with the system copy workflow.



In this chapter, we briefly discuss the differences for the system clone workflow and input screens. As you can see in the following image, nothing changes in the storage workflow.



1. The system copy workflow can be started when the system is prepared accordingly. This is not a specific task for this configuration, and we do not explain it in detail. If you need further information, review the SAP LaMa documentation.



2. During the copy workflow, the system is renamed, as must be specified in the first screen.

Copy System HN1 on sap-tnv44 (11:16)

HN1: NetWeaver ABAP 7.77, cshna

Basic » Hosts » Host Names » **Instance Number** » Custom Clone » Consistency » Users » Rename » Isolation » ABAP PCA » Summary

Provide Basic Data for Target System

*System ID: HN2

☒ Use different Database Name

*HANA SID: H02

*Pool: MUCCBC

Description: Copy of System 'HNA'

Set Master Password for OS and DB Users

*Password: *****

*Confirm Password: *****

Ignore Warnings for This Step Validate Step Reset Step

< Previous **Next** > Finish Execute Cancel

3. During the workflow, you can change the instance numbers.

Copy System HN1 on sap-tnv44 (11:16)

HN1: NetWeaver ABAP 7.77, cshna

Basic » Hosts » Host Names » **Instance Number** » Custom Clone » Consistency » Users » Rename » Isolation » ABAP PCA » Summary

SAP Instance Numbers

*System database: MASTER (configured) : SAP HANA 02

02

*AS instance: 00

00

*Central services: 01

01

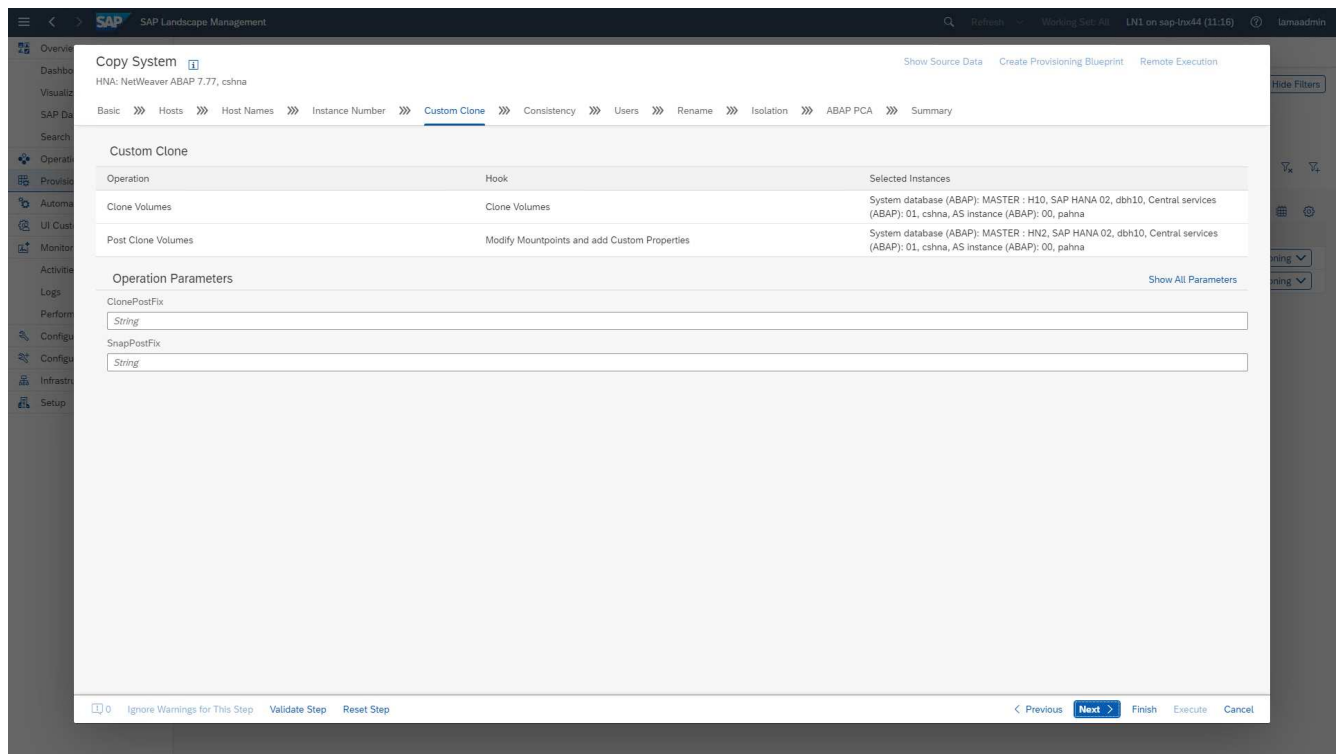
Ignore Warnings for This Step Validate Step Reset Step

< Previous **Next** > Finish Execute Cancel

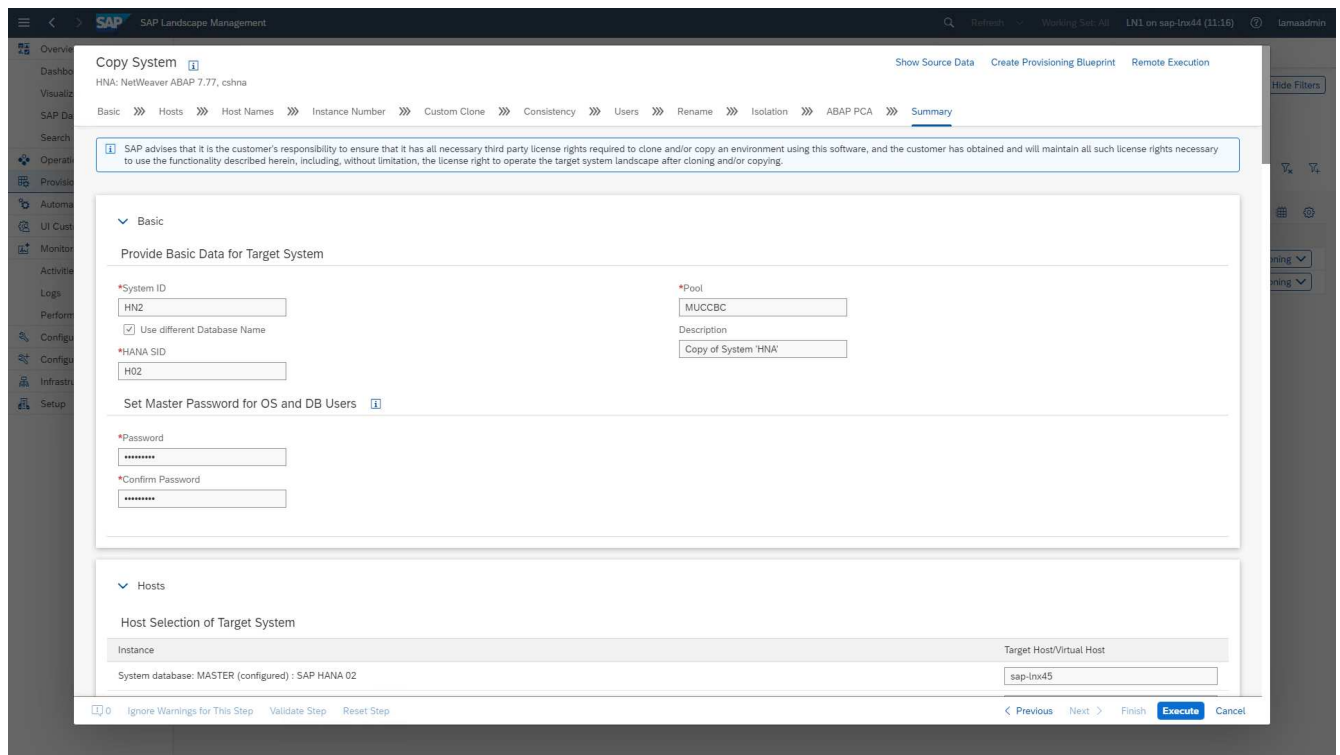


Changing instance numbers has not been tested and might require changes in the provider script.

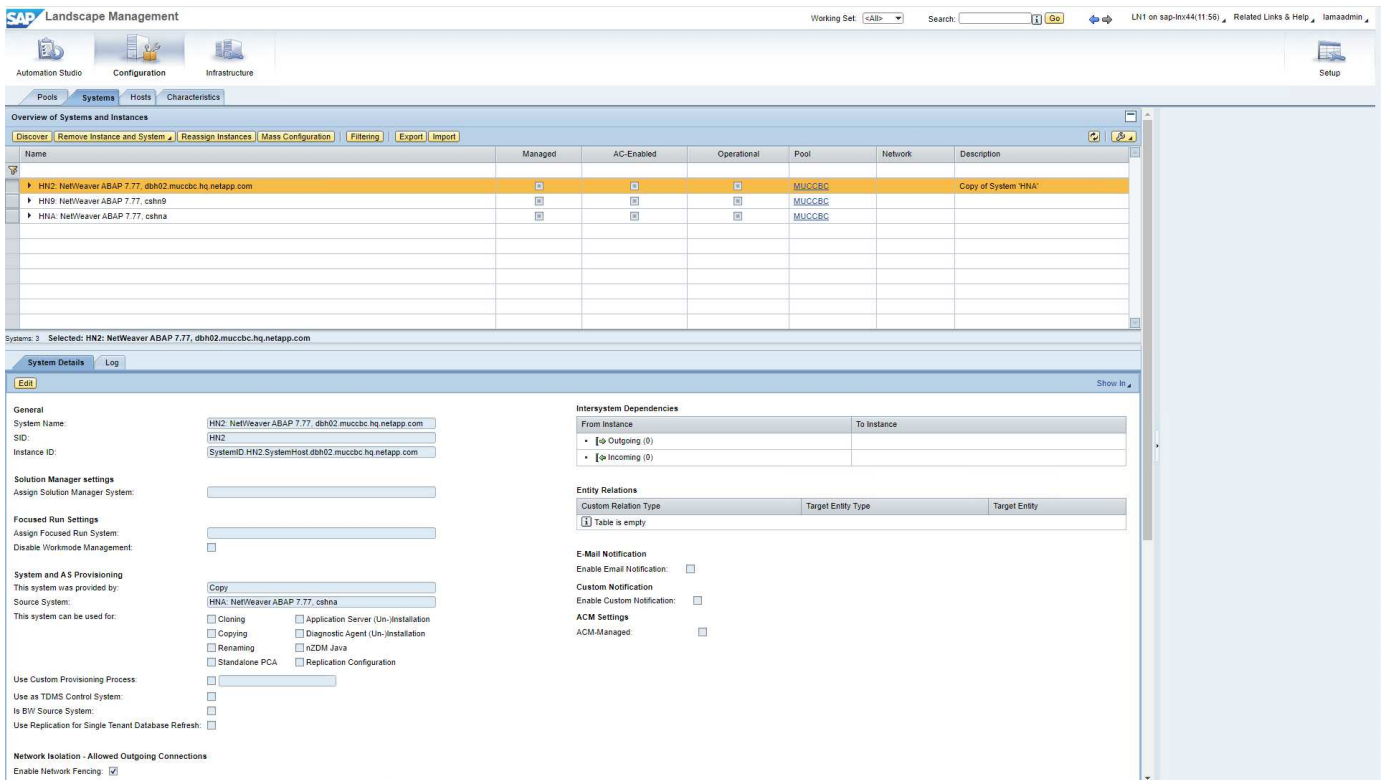
4. As described, the **Custom Clone** screen does not differ from the cloning workflow, as is shown here.



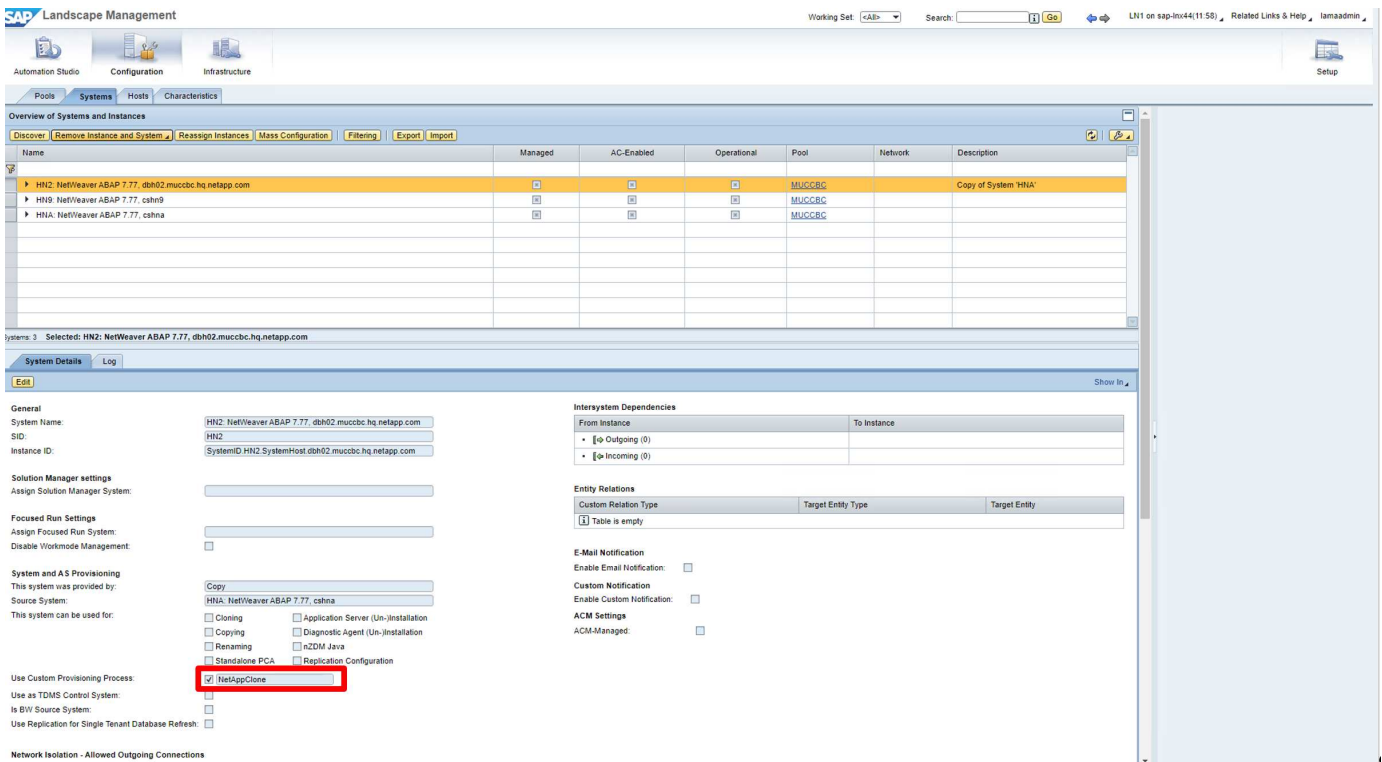
5. As we already described, the remaining input masks do not deviate from the standard, and we do not go into them any further here. The final screen shows a summary, and execution can now be started.



After the copy process, the target instance is not enabled for the custom cloning process.

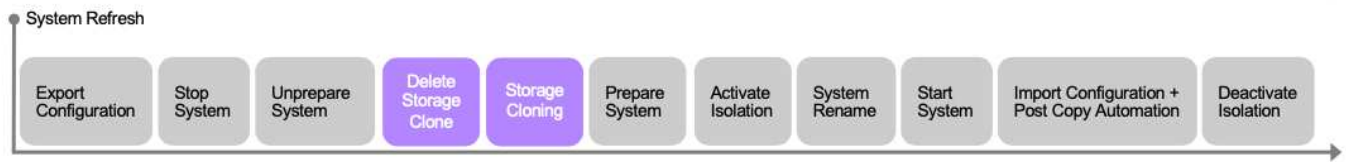


It must be adopted manually to run the pre-hook step during the system destroy process because a constraint is set and would prevent execution.

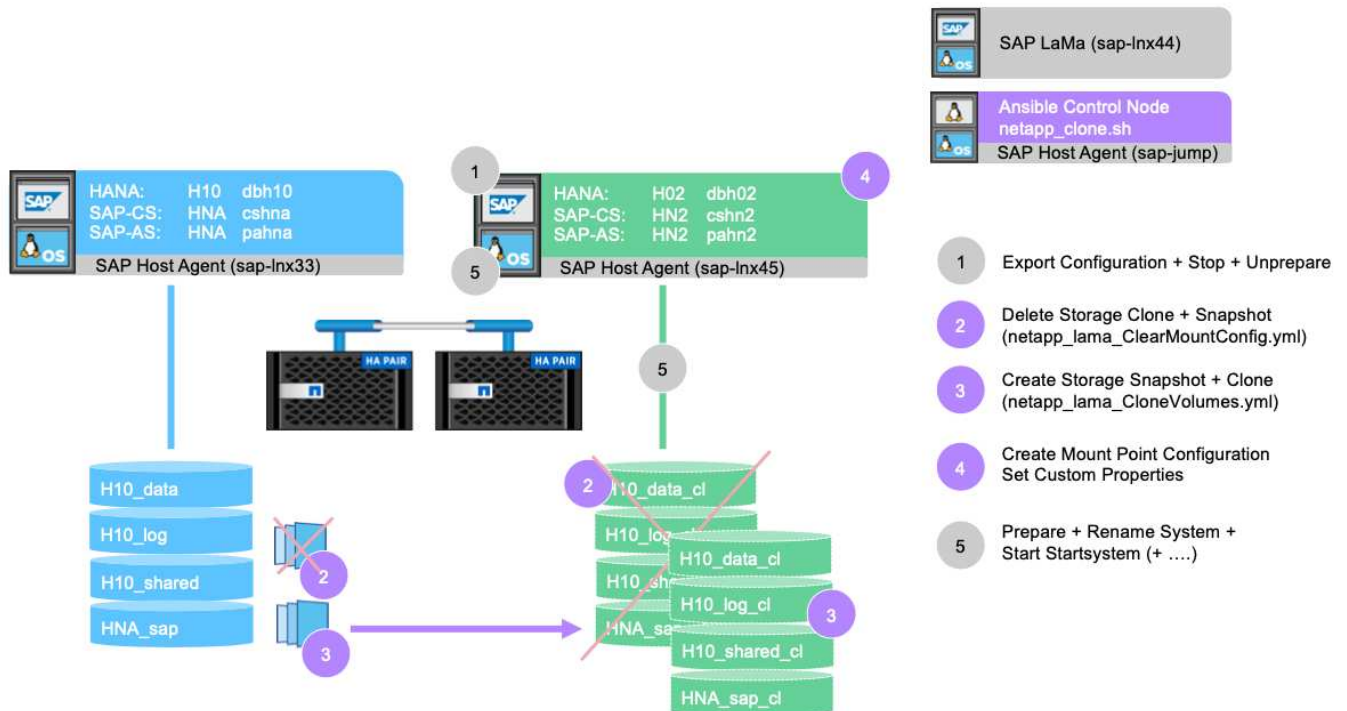


SAP LaMa provisioning workflow - system refresh

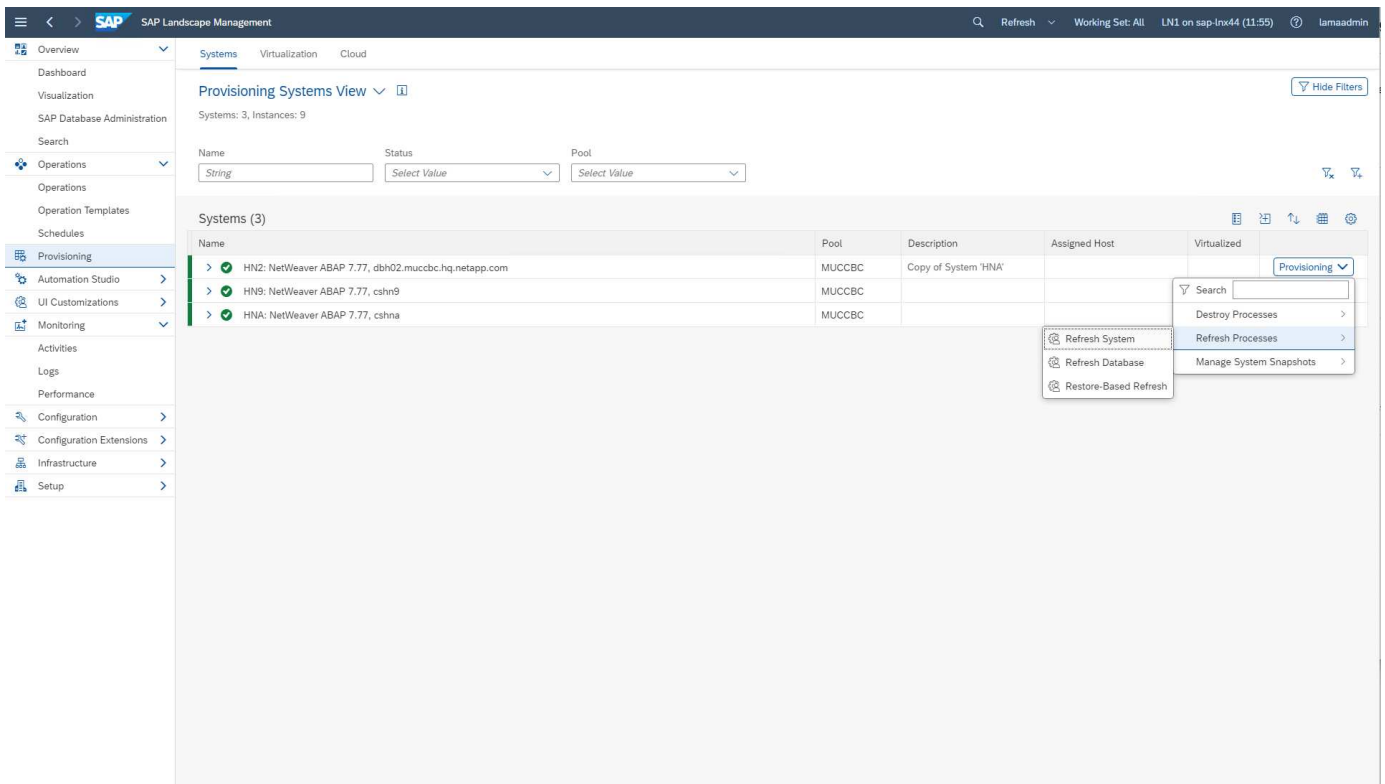
The following figure highlights the main steps executed with the system refresh workflow.



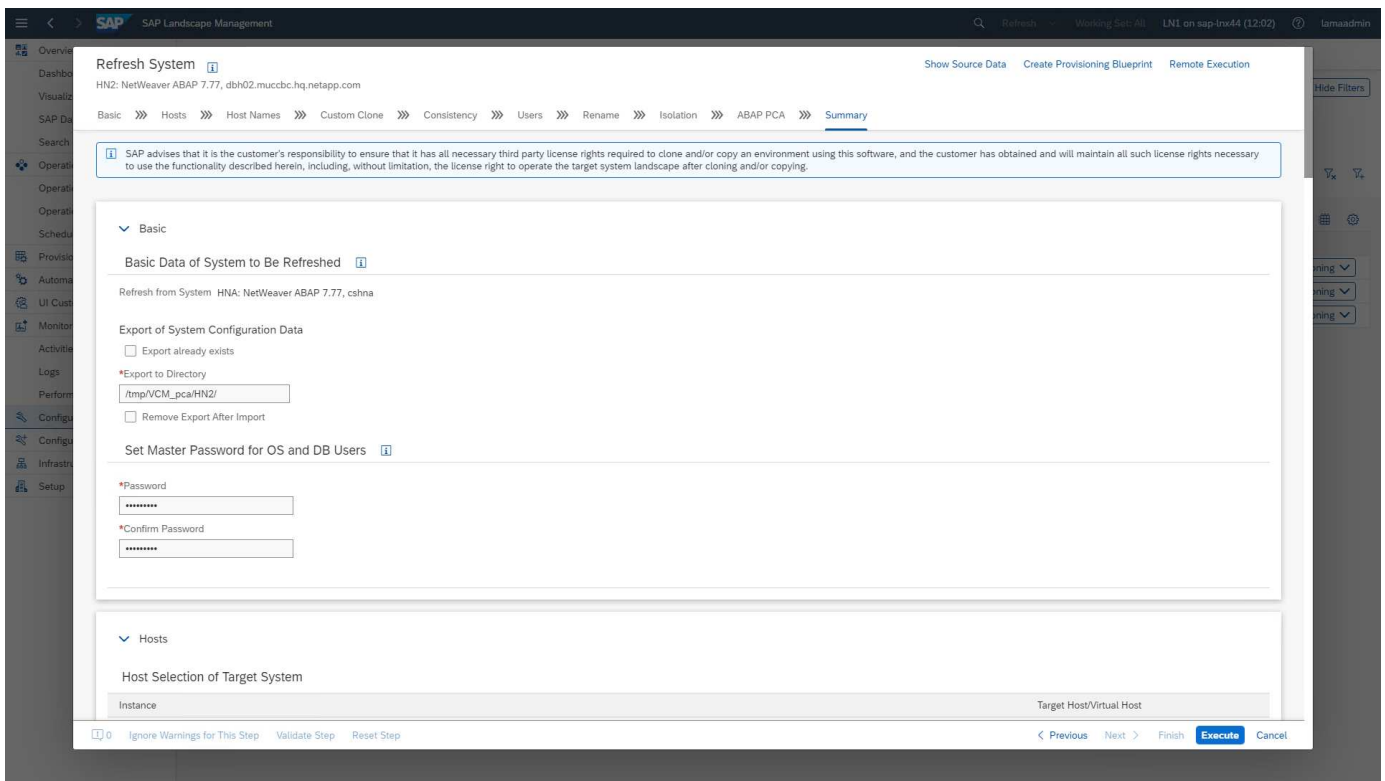
During the refresh workflow, the storage clone must be deleted. You can use the same Ansible playbook as for the system destroy workflow. However, the custom hook is defined to a different step, so the playbook is named accordingly. The process step for the clone doesn't differ.



The refresh workflow can be triggered through the provisioning screen for a copied system.



Again, nothing differs in the input screens from the standard, and the workflow execution can be started from the summary screen.



Provider script configuration and Ansible playbooks

The following provider configuration file, execution script, and Ansible playbooks are used

during the sample deployment and workflow execution in this documentation.



The example scripts are provided as is and are not supported by NetApp. You can request the current version of the scripts via email to ng-sapcc@netapp.com.

Provider configuration file `netapp_clone.conf`

The configuration file is created as described in the [SAP LaMa Documentation - Configuring SAP Host Agent Registered Scripts](#). This configuration file must be located on the Ansible control node where the SAP host agent is installed.

The configured os-user `sapuser` must have the appropriate permissions to execute the script and the called Ansible playbooks. You can place the script in a common script directory. SAP LaMa can provide multiple parameters when calling the script.

In addition to the custom parameters, `PARAM_ClonePostFix`, `PROP_ClonePostFix`, `PARAM_ClonePostFix`, and `PROP_ClonePostFix`, many others can be handed over, as is shown in the [SAP LaMa Documentation](#).

```
root@sap-jump:~# cat /usr/sap/hostctrl/exe/operations.d/netapp_clone.conf
Name: netapp_clone
Username: sapuser
Description: NetApp Clone for Custom Provisioning
Command: /usr/sap/scripts/netapp_clone.sh
--HookOperationName=${HookOperationName} --SAPSYSTEMNAME=${SAPSYSTEMNAME}
--SAPSYSTEM=${SAPSYSTEM} --MOUNT_XML_PATH=${MOUNT_XML_PATH}
--PARAM_ClonePostFix=${PARAM_ClonePostFix} --PARAM_SnapPostFix=${PARAM
-SnapPostFix} --PROP_ClonePostFix=${PROP_ClonePostFix}
--PROP_SnapPostFix=${PROP_SnapPostFix}
--SAP_LVM_SRC_SID=${SAP_LVM_SRC_SID}
--SAP_LVM_TARGET_SID=${SAP_LVM_TARGET_SID}
ResulConverter: hook
Platform: Unix
```

Provider script `netapp_clone.sh`

The provider script must be stored in `/usr/sap/scripts` as configured in the provider configuration file.

Variables

The following variables are hard coded in the script and must be adapted accordingly.

- `PRIMARY_CLUSTER=<hostname of netapp cluster>`
- `PRIMARY_SVM=<SVM name where source system volumes are stored>`

The certificate files `PRIMARY_KEYFILE=/usr/sap/scripts/ansible/certs/ontap.key` and `PRIMARY_CERTFILE=/usr/sap/scripts/ansible/certs/ontap.pem` must be provided as described in [NetApp Ansible modules - Prepare ONTAP](#).



If different clusters or SVMs are required for different SAP systems, these variables can be added as parameters in the SAP LaMa provider definition.

Function: create inventory file

To make Ansible playbook execution more dynamic, an `inventory.yml` file is created on the fly. Some static values are configured in the variable section and some are dynamically created during execution.

Function: run Ansible playbook

This function is used to execute the Ansible playbook together with the dynamically created `inventory.yml` file. The naming convention for the playbooks is `netapp_lama_${HookOperationName}.yml`. The values for `${HookOperationName}` is dependent on the LaMa operation and handed over by LaMa as a command line parameter.

Section Main

This section contains the main execution plan. The variable `${HookOperationName}` contains the name of the LaMa replacement step and is provided by LaMa when the script is called.

- Values with the system clone and system copy provisioning workflow:
 - `CloneVolumes`
 - `PostCloneVolumes`
- Value with the system destroy workflow:
 - `ServiceConfigRemoval`
- Value with the system refresh workflow:
 - `ClearMountConfig`

HookOperationName = CloneVolumes

With this step, the Ansible playbook is executed, which triggers the Snapshot copy and cloning operation. The volume names and mount configuration are handed over by SAP LaMa through an XML file defined in the variable `$MOUNT_XML_PATH`. This file is saved because it is used later in the step `FinalizeCloneVolumes` to create the new mount-point configuration. The volume names are extracted from the XML file and the Ansible cloning playbook is executed for each volume.



In this example, the AS instance and the central services share the same volume. Therefore, volume cloning is only executed when the SAP instance number (`$SAPSYSTEM`) is not 01. This might differ in other environments and must be changed accordingly.

HookOperationName = PostCloneVolumes

During this step, the custom properties `ClonePostFix` and `SnapPostFix` and the mount point configuration for the target system are maintained.

The custom properties are used later as input when the system is decommissioned during the `ServiceConfigRemoval` or `ClearMountConfig` phase. The system is designed to preserve the settings of the custom parameters that were specified during the system provisioning workflow.

The values used in this example are `ClonePostFix=_clone_20221115` and

SnapPostFix=_snap_20221115.

For the volume HN9_sap, the dynamically created Ansible file includes the following values:
datavolumename: HN9_sap, snapshotpostfix: _snap_20221115, and clonepostfix: _clone_20221115.

Which leads into the snapshot name on the volume HN9_sap HN9_sap_snap_20221115 and the created volume clone name HN9_sap_clone_20221115.



Custom properties could be used in any way to preserve parameters used during the provisioning process.

The mount point configuration is extracted from the XML file that has been handed over by LaMa in the CloneVolume step. The ClonePostFix is added to the volume names and send back to LaMa through the default script output. The functionality is described in [SAP Note 1889590](#).



In this example, qtrees on the storage system are used as a common way to place different data on a single volume. For example, HN9_sap holds the mount points for /usr/sap/HN9, /sapmnt/HN9, and /home/hn9adm. Subdirectories work in the same way. This might differ in other environments and must be changed accordingly.

HookOperationName = ServiceConfigRemoval

In this step, the Ansible playbook that is responsible for the deletion of the volume clones is running.

The volume names are handed over by SAP LaMa through the mount configuration file, and the custom properties ClonePostFix and SnapPostFix are used to hand over the values of the parameters originally specified during the system provisioning workflow (see the note at HookOperationName = PostCloneVolumes).

The volume names are extracted from the xml file, and the Ansible cloning playbook is executed for each volume.



In this example, the AS instance and the central services share the same volume. Therefore, the volume deletion is only executed when the SAP instance number (\$SAPSYSTEM) is not 01. This might differ in other environments and must be changed accordingly.

HookOperationName = ClearMountConfig

In this step, the Ansible playbook that is responsible for the deletion of the volume clones during a system refresh workflow is running.

The volume names are handed over by SAP LaMa through the mount configuration file, and the custom properties ClonePostFix and SnapPostFix are used to hand over the values of the parameters originally specified during the system provisioning workflow.

The volume names are extracted from the XML file and the Ansible cloning playbook is executed for each volume.



In this example, the AS instance and the central services share the same volume. Therefore, volume deletion is only executed when the SAP instance number (\$SAPSYSTEM) is not 01. This might differ in other environments and must be changed accordingly.


```

root@sap-jump:~# cat /usr/sap/scripts/netapp_clone.sh
#!/bin/bash
#Section - Variables
#####
VERSION="Version 0.9"
#Path for ansible play-books
ANSIBLE_PATH=/usr/sap/scripts/ansible
#Values for Ansible Inventory File
PRIMARY_CLUSTER=grenada
PRIMARY_SVM=svm-sap01
PRIMARY_KEYFILE=/usr/sap/scripts/ansible/certs/ontap.key
PRIMARY_CERTFILE=/usr/sap/scripts/ansible/certs/ontap.pem
#Default Variable if PARAM ClonePostFix / SnapPostFix is not maintained in
LaMa
DefaultPostFix=_clone_1
#TMP Files - used during execution
YAML_TMP=/tmp/inventory_ansible_clone_tmp_$$.yml
TMPFILE=/tmp/tmpfile.$$
MY_NAME="`basename $0`"
BASE_SCRIPT_DIR="`dirname $0`"
#Sendig Script Version and run options to LaMa Log
echo "[DEBUG]: Running Script $MY_NAME $VERSION"
echo "[DEBUG]: $MY_NAME $@"
#Command declared in the netapp_clone.conf Provider definition
#Command: /usr/sap/scripts/netapp_clone.sh
--HookOperationName=${HookOperationName} --SAPSYSTEMNAME=${SAPSYSTEMNAME}
--SAPSYSTEM=${SAPSYSTEM} --MOUNT_XML_PATH=${MOUNT_XML_PATH}
--PARAM_ClonePostFix=${PARAM_ClonePostFix} --PARAM_SnapPostFix=${PARAM
-SnapPostFix} --PROP_ClonePostFix=${PROP_ClonePostFix}
--PROP_SnapPostFix=${PROP_SnapPostFix}
--SAP_LVM_SRC_SID=${SAP_LVM_SRC_SID}
--SAP_LVM_TARGET_SID=${SAP_LVM_TARGET_SID}
#Reading Input Variables hand over by LaMa
for i in "$@"
do
case $i in
--HookOperationName=*)
HookOperationName="${i#*=}";shift;;
--SAPSYSTEMNAME=*)
SAPSYSTEMNAME="${i#*=}";shift;;
--SAPSYSTEM=*)
SAPSYSTEM="${i#*=}";shift;;
--MOUNT_XML_PATH=*)
MOUNT_XML_PATH="${i#*=}";shift;;
--PARAM_ClonePostFix=*)

```

```

PARAM_ClonePostFix="${i#*=}";shift;;
--PARAM_SnapPostFix=*)
PARAM_SnapPostFix="${i#*=}";shift;;
--PROP_ClonePostFix=*)
PROP_ClonePostFix="${i#*=}";shift;;
--PROP_SnapPostFix=*)
PROP_SnapPostFix="${i#*=}";shift;;
--SAP_LVM_SRC_SID=*)
SAP_LVM_SRC_SID="${i#*=}";shift;;
--SAP_LVM_TARGET_SID=*)
SAP_LVM_TARGET_SID="${i#*=}";shift;;
*)
# unknown option
;;
esac
done
#If Parameters not provided by the User - defaulting to DefaultPostFix
if [ -z $PARAM_ClonePostFix ]; then PARAM_ClonePostFix=$DefaultPostFix;fi
if [ -z $PARAM_SnapPostFix ]; then PARAM_SnapPostFix=$DefaultPostFix;fi
#Section - Functions
#####
#Function Create (Inventory) YML File
#####
create_yml_file()
{
echo "ontapservers:">$YAML_TMP
echo " hosts:">>$YAML_TMP
echo "   ${PRIMARY_CLUSTER}:">>$YAML_TMP
echo "   ansible_host: ''''$PRIMARY_CLUSTER''''>>$YAML_TMP
echo "   keyfile: ''''$PRIMARY_KEYFILE''''>>$YAML_TMP
echo "   certfile: ''''$PRIMARY_CERTFILE''''>>$YAML_TMP
echo "   svmname: ''''$PRIMARY_SVM''''>>$YAML_TMP
echo "   datavolumename: ''''$datavolumename''''>>$YAML_TMP
echo "   snapshotpostfix: ''''$snapshotpostfix''''>>$YAML_TMP
echo "   clonepostfix: ''''$clonepostfix''''>>$YAML_TMP
}
#Function run ansible-playbook
#####
run_ansible_playbook()
{
echo "[DEBUG]: Running ansible playbook
netapp_lama_${HookOperationName}.yaml on Volume $datavolumename"
ansible-playbook -i $YAML_TMP
$ANSIBLE_PATH/netapp_lama_${HookOperationName}.yaml
}
#Section - Main

```

```
#####
#HookOperationName - CloneVolumes
#####
if [ $HookOperationName = CloneVolumes ] ;then
#save mount xml for later usage - used in Section FinalizeCloneVolumes to
generate the mountpoints
echo "[DEBUG]: saving mount config...."
cp $MOUNT_XML_PATH /tmp/mount_config_${SAPSYSTEMNAME}_${SAPSYSTEM}.xml
#Instance 00 + 01 share the same volumes - clone needs to be done once
if [ $SAPSYSTEM != 01 ]; then
#generating Volume List - assuming usage of qtrees - "IP-
Adress:/VolumeName/qtrees"
xmlFile=/tmp/mount_config_${SAPSYSTEMNAME}_${SAPSYSTEM}.xml
if [ -e $TMPFILE ];then rm $TMPFILE;fi
numMounts=`xml_grep --count "/mountconfig/mount" $xmlFile | grep "total: "
| awk '{ print $2 }'`
i=1
while [ $i -le $numMounts ]; do
    xmllint --xpath "/mountconfig/mount[$i]/exportpath/text()" $xmlFile
|awk -F"/" '{print $2}' >>$TMPFILE
i=$((i + 1))
done
DATAVOLUMES=`cat $TMPFILE |sort -u`
#Create yml file and rund playbook for each volume
for I in $DATAVOLUMES; do
datavolumename="$I"
snapshotpostfix="$PARAM_SnapPostFix"
clonepostfix="$PARAM_ClonePostFix"
create_yml_file
run_ansible_playbook
done
else
echo "[DEBUG]: Doing nothing .... Volume cloned in different Task"
fi
fi
#HookOperationName - PostCloneVolumes
#####
if [ $HookOperationName = PostCloneVolumes ] ;then
#Reporting Properties back to LaMa Config for Cloned System
echo "[RESULT]:Property:ClonePostFix=$PARAM_ClonePostFix"
echo "[RESULT]:Property:SnapPostFix=$PARAM_SnapPostFix"
#Create MountPoint Config for Cloned Instances and report back to LaMa
according to SAP Note: https://launchpad.support.sap.com/#/notes/1889590
echo "MountDataBegin"
echo '<?xml version="1.0" encoding="UTF-8"?>'
echo "<mountconfig>"
```

```

xmlFile=/tmp/mount_config_${SAPSYSTEMNAME}_${SAPSYSTEM}.xml
numMounts=`xml_grep --count "/mountconfig/mount" $xmlFile | grep "total: "
| awk '{ print $2 }'`
i=1
while [ $i -le $numMounts ]; do
MOUNTPOINT=`xmllint --xpath "/mountconfig/mount[$i]/mountpoint/text()"
$xmlFile`;
EXPORTPATH=`xmllint --xpath
"/mountconfig/mount[$i]/exportpath/text()" $xmlFile`;
OPTIONS=`xmllint --xpath "/mountconfig/mount[$i]/options/text()"
$xmlFile`;
#Adopt Exportpath and add Clonepostfix - assuming usage of qtrees - "IP-
Adress:/VolumeName/qtrees"
TMPFIELD1=`echo $EXPORTPATH|awk -F"/" '{print $1}'`
TMPFIELD2=`echo $EXPORTPATH|awk -F"/" '{print $2}'`
TMPFIELD3=`echo $EXPORTPATH|awk -F"/" '{print $3}'`
EXPORTPATH=$TMPFIELD1":/${TMPFIELD2}$PARAM_ClonePostFix"/$TMPFIELD3
echo -e '\t<mount fstype="nfs" storagetype="NETFS">'
echo -e "\t\t<mountpoint>${MOUNTPOINT}</mountpoint>"
echo -e "\t\t<exportpath>${EXPORTPATH}</exportpath>"
echo -e "\t\t<options>${OPTIONS}</options>"
echo -e "\t</mount>"
i=$((i + 1))
done
echo "</mountconfig>"
echo "MountDataEnd"
#Finished MountPoint Config
#Cleanup Temporary Files
rm $xmlFile
fi
#HookOperationName - ServiceConfigRemoval
#####
if [ $HookOperationName = ServiceConfigRemoval ] ;then
#Assure that Properties ClonePostFix and SnapPostfix has been configured
through the provisioning process
if [ -z $PROP_ClonePostFix ]; then echo "[ERROR]: Propertiy ClonePostFix
is not handed over - please investigate";exit 5;fi
if [ -z $PROP_SnapPostFix ]; then echo "[ERROR]: Propertiy SnapPostFix is
not handed over - please investigate";exit 5;fi
#Instance 00 + 01 share the same volumes - clone delete needs to be done
once
if [ $SAPSYSTEM != 01 ]; then
#generating Volume List - assuming usage of qtrees - "IP-
Adress:/VolumeName/qtrees"
xmlFile=$MOUNT_XML_PATH
if [ -e $TMPFILE ];then rm $TMPFILE;fi

```

```

numMounts=`xml_grep --count "/mountconfig/mount" $xmlFile | grep "total: "
| awk '{ print $2 }'`
i=1
while [ $i -le $numMounts ]; do
    xmllint --xpath "/mountconfig/mount[$i]/exportpath/text()" $xmlFile
|awk -F"/" '{print $2}' >>$TMPFILE
i=$((i + 1))
done
DATAVOLUMES=`cat $TMPFILE |sort -u| awk -F $PROP_ClonePostFix '{ print $1
}'`
#Create yaml file and rund playbook for each volume
for I in $DATAVOLUMES; do
datavolumename="$I"
snapshotpostfix="$PROP_SnapPostFix"
clonepostfix="$PROP_ClonePostFix"
create_yaml_file
run_ansible_playbook
done
else
echo "[DEBUG]: Doing nothing .... Volume deleted in different Task"
fi
#Cleanup Temporary Files
rm $xmlFile
fi
#HookOperationName - ClearMountConfig
#####
if [ $HookOperationName = ClearMountConfig ] ;then
    #Assure that Properties ClonePostFix and SnapPostfix has been
configured through the provisioning process
    if [ -z $PROP_ClonePostFix ]; then echo "[ERROR]: Propertiy
ClonePostFix is not handed over - please investigate";exit 5;fi
    if [ -z $PROP_SnapPostFix ]; then echo "[ERROR]: Propertiy
SnapPostFix is not handed over - please investigate";exit 5;fi
    #Instance 00 + 01 share the same volumes - clone delete needs to
be done once
    if [ $SAPSYSTEM != 01 ]; then
        #generating Volume List - assuming usage of qtrees - "IP-
Adress:/VolumeName/qtrees"
        xmlFile=$MOUNT_XML_PATH
        if [ -e $TMPFILE ];then rm $TMPFILE;fi
        numMounts=`xml_grep --count "/mountconfig/mount" $xmlFile
| grep "total: " | awk '{ print $2 }'`
        i=1
        while [ $i -le $numMounts ]; do
            xmllint --xpath
"/mountconfig/mount[$i]/exportpath/text()" $xmlFile |awk -F"/" '{print

```

```

$2}' >>$TMPFILE

        i=$((i + 1))
    done
    DATAVOLUMES=`cat $TMPFILE |sort -u| awk -F
$PROP_ClonePostFix '{ print $1 }'`
    #Create yml file and rund playbook for each volume
    for I in $DATAVOLUMES; do
        datavolumename="$I"
        snapshotpostfix="$PROP_SnapPostFix"
        clonepostfix="$PROP_ClonePostFix"
        create_yml_file
        run_ansible_playbook
    done
else
    echo "[DEBUG]: Doing nothing .... Volume deleted in
different Task"
    fi
    #Cleanup Temporary Files
    rm $xmlFile
fi
#Cleanup
#####
#Cleanup Temporary Files
if [ -e $TMPFILE ];then rm $TMPFILE;fi
if [ -e $YAML_TMP ];then rm $YAML_TMP;fi
exit 0

```

Ansible Playbook netapp_lama_CloneVolumes.yml

The playbook that is executed during the CloneVolumes step of the LaMa system clone workflow is a combination of `create_snapshot.yml` and `create_clone.yml` (see [NetApp Ansible modules - YAML files](#)). This playbook can be easily extended to cover additional use cases like cloning from secondary and clone split operations.

```

root@sap-jump:~# cat /usr/sap/scripts/ansible/netapp_lama_CloneVolumes.yml
---
- hosts: ontapservers
  connection: local
  collections:
    - netapp.ontap
  gather_facts: false
  name: netapp_lama_CloneVolumes
  tasks:
    - name: Create SnapShot
      na_ontap_snapshot:
        state: present
        snapshot: "{{ datavolumename }}{{ snapshotpostfix }}"
        use_rest: always
        volume: "{{ datavolumename }}"
        vserver: "{{ svmname }}"
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false
    - name: Clone Volume
      na_ontap_volume_clone:
        state: present
        name: "{{ datavolumename }}{{ clonepostfix }}"
        use_rest: always
        vserver: "{{ svmname }}"
        junction_path: '/{{ datavolumename }}{{ clonepostfix }}'
        parent_volume: "{{ datavolumename }}"
        parent_snapshot: "{{ datavolumename }}{{ snapshotpostfix }}"
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false

```

Ansible Playbook netapp_lama_ServiceConfigRemoval.yml

The playbook that is executed during the ServiceConfigRemoval phase of the LaMa system destroy workflow is combination of delete_clone.yml and delete_snapshot.yml (see [NetApp Ansible modules - YAML files](#)). It must be aligned to the execution steps of the netapp_lama_CloneVolumes playbook.

```

root@sap-jump:~# cat
/usr/sap/scripts/ansible/netapp_lama_ServiceConfigRemoval.yml
---
- hosts: ontapservers
  connection: local
  collections:
    - netapp.ontap
  gather_facts: false
  name: netapp_lama_ServiceConfigRemoval
  tasks:
    - name: Delete Clone
      na_ontap_volume:
        state: absent
        name: "{{ datavolumename }}{{ clonepostfix }}"
        use_rest: always
        vsserver: "{{ svmname }}"
        wait_for_completion: True
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false
    - name: Delete SnapShot
      na_ontap_snapshot:
        state: absent
        snapshot: "{{ datavolumename }}{{ snapshotpostfix }}"
        use_rest: always
        volume: "{{ datavolumename }}"
        vsserver: "{{ svmname }}"
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false
root@sap-jump:~#

```

Ansible Playbook netapp_lama_ClearMountConfig.yml

The playbook, which is executed during the netapp_lama_ClearMountConfig phase of the LaMa system refresh workflow is combination of delete_clone.yml and delete_snapshot.yml (see [NetApp Ansible modules - YAML files](#)). It must be aligned to the execution steps of the netapp_lama_CloneVolumes playbook.


```

root@sap-jump:~# cat
/usr/sap/scripts/ansible/netapp_lama_ServiceConfigRemoval.yml
---
- hosts: ontapservers
  connection: local
  collections:
    - netapp.ontap
  gather_facts: false
  name: netapp_lama_ServiceConfigRemoval
  tasks:
    - name: Delete Clone
      na_ontap_volume:
        state: absent
        name: "{{ datavolumename }}{{ clonepostfix }}"
        use_rest: always
        vsserver: "{{ svmname }}"
        wait_for_completion: True
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false
    - name: Delete SnapShot
      na_ontap_snapshot:
        state: absent
        snapshot: "{{ datavolumename }}{{ snapshotpostfix }}"
        use_rest: always
        volume: "{{ datavolumename }}"
        vsserver: "{{ svmname }}"
        hostname: "{{ inventory_hostname }}"
        cert_filepath: "{{ certfile }}"
        key_filepath: "{{ keyfile }}"
        https: true
        validate_certs: false
root@sap-jump:~#

```

Sample Ansible inventory.yml

This inventory file is dynamically built during workflow execution, and it is only shown here for illustration.

```
ontapservers:
  hosts:
    grenada:
      ansible_host: "grenada"
      keyfile: "/usr/sap/scripts/ansible/certs/ontap.key"
      certfile: "/usr/sap/scripts/ansible/certs/ontap.pem"
      svmname: "svm-sap01"
      datavolumename: "HN9_sap"
      snapshotpostfix: " _snap_20221115"
      clonepostfix: " _clone_20221115"
```

Conclusion

The integration of a modern automation framework like Ansible into SAP LaMa provisioning workflows gives customers a flexible solution to address standard or more complex infrastructure requirements.

Where to find additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

- Collections in the NetApp Namespace

<https://docs.ansible.com/ansible/latest/collections/netapp/index.html>

- Documentation about Ansible Integration and Sample Ansible Playbooks

https://github.com/sap-linuxlab/demo.netapp_ontap

- General Ansible and NetApp Integration

<https://www.ansible.com/integrations/infrastructure/netapp>

- Blog on integrating SAP LaMa with Ansible

<https://blogs.sap.com/2020/06/08/outgoing-api-calls-from-sap-landscape-management-lama-with-automation-studio/>

- SAP Landscape Management 3.0, Enterprise Edition Documentation

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/4df88a8f418c5059e10000000a42189c.html#loio4df88a8f418c5059e10000000a42189c>

- SAP LaMa Documentation – Provider Definitions

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/bf6b3e43340a4cbcb0c0f3089715c068.html>

- SAP LaMa Documentation - Custom Hooks

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/139eca2f925e48738a20dbf0b56674c5.html>

- SAP LaMa Documentation - Configuring SAP Host Agent Registered Scripts

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/250dfc5eef4047a38bab466c295d3a49.html>

- SAP LaMa Documentation - Parameters for Custom Operations and Custom Hooks

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/0148e495174943de8c1c3ee1b7c9cc65.html>

- SAP LaMa Documentation - Adaptive Design

<https://help.sap.com/doc/700f9a7e52c7497cad37f7c46023b7ff/3.0.11.0/en-US/737a99e86f8743bdb8d1f6cf4b862c79.html>

- NetApp Product Documentation

<https://www.netapp.com/support-and-training/documentation/>

Version history

Version	Date	Document version history
Version 1.0	January 2023	Initial release

Copyright information

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.