

Do More with Less: Automating IBM Storage FlashSystem Tasks with REST APIs, Scripting, and Ansible

Sergei Kubin

Uwe Schreiber

Vasfi Gucer



Storage

Contents

Notices	iii
Trademarks	iv
Preface	v
Authors	v
Now you can become a published author, too!	vi
Comments welcome	vi
Stay connected to IBM Redbooks	vi
Chapter 1. Introduction	1
1.1 Introduction	2
1.1.1 Increased agility and faster storage provisioning	2
1.1.2 Reduced human error and improved consistency	2
1.1.3 Simplified management of complex storage tasks	3
1.2 Example use cases	3
1.2.1 Automated provisioning	3
1.2.2 Automated backup and recovery	3
1.2.3 Performance optimization	3
1.2.4 Automating storage reporting	3
1.2.5 Automated user access control	4
1.3 Infrastructure as Code (IaC)	4
1.3.1 Role of IaC in automation	4
1.4 Storage Virtualize automation opportunities	5
Chapter 2. User management and security	7
2.1 System security introduction	8
2.1.1 System security	8
2.1.2 Data security	8
2.2 User management	8
2.2.1 Users and roles	8
2.2.2 Password policies and account locking	11
2.2.3 Setting session time-outs	13
2.3 Superuser security	13
2.3.1 Superuser specifics	13
2.3.2 Locking and unlocking the superuser account	15
2.3.3 Superuser password reset	15
2.4 Additional security with MFA and TPI	16
2.4.1 Multifactor Authentication	17
2.4.2 Two person integrity	17
2.5 TLS Certificates	19
Chapter 3. Automation with CLI scripting	23
3.1 Advanced CLI usage and scripting	24
3.1.1 Secure restricted shell	24
3.1.2 General tips	24
3.1.3 Internal scripting commands	27
3.1.4 Script examples	28
Chapter 4. Automation with REST API	31

4.1 REST API on IBM Storage Virtualize	32
4.1.1 REST API Explorer	32
4.1.2 Using curl to access REST API.	34
4.1.3 Using REST API with PowerShell.	36
4.1.4 Using REST API in Python and Perl scripts	39
Chapter 5. Automation with Red Hat Ansible.	43
5.1 Introduction to Ansible.	44
5.1.1 Key features of Ansible.	44
5.1.2 Common use cases for Ansible	44
5.2 Automation with Red Hat Ansible	45
5.2.1 Red Hat Ansible	45
5.2.2 Red Hat Ansible editions.	46
5.2.3 Requirements	46
5.2.4 Essential terminology in an Ansible environment	47
5.2.5 Automating IBM Storage with Ansible.	47
5.2.6 Getting started	49
5.2.7 Securing credentials in Ansible	53
5.2.8 Creating an Ansible playbook	53
5.2.9 More automation	59
Related publications	61
IBM Redbooks	61
Online resources	61
Help from IBM	61



IBM Redbooks

**Do More with Less: Automating IBM Storage
FlashSystem Tasks with REST APIs, Scripting, and
Ansible**

July 2024

Note: Before using this information and the product it supports, read the information in “Notices” on page iii.

First Edition (July 2024)

This edition applies to IBM Storage Virtualize Version 8.7.

This document was created or updated on July 15, 2024.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <https://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

FlashCopy®

IBM®

IBM FlashSystem®


IBM Security®

IBM Spectrum®

Orchestrate®

QRadar®

Redbooks®

Redbooks (logo) ®

Storwize®

The following terms are trademarks of other companies:

ITIL is a Registered Trade Mark of AXELOS Limited.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Ansible, Red Hat, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The ever-expanding demands of data storage in today's information-intensive landscape place a significant burden on IT professionals tasked with managing storage infrastructure. Manual processes for provisioning resources, troubleshooting issues, and performing routine maintenance tasks can be time-consuming and resource-intensive, hindering overall efficiency.

This IBM Redpaper equips you with the knowledge and skills to automate essential IBM Storage FlashSystem and IBM SAN Volume Controller tasks through the implementation of REST APIs, scripting languages, and Ansible. Practical examples are provided to illustrate the concepts and guide your learning process.

The target audience of this book is IBM Storage FlashSystem administrators and Storage Automation Professionals.

Authors

This paper was produced by a team of specialists from around the world.



Sergei Kubin is a Senior Storage Support Engineer working in GBM Qatar. He holds an Electronics Engineer degree from Ural Federal University in Russia and has more than 15 years of experience in IT. In GBM, he provides support and guidance for customers using IBM and multi-vendor storage solutions. His expertise includes file and block storage, and storage area networks.



Uwe Schreiber is a Solution Architect at SVA System Vertrieb Alexander GmbH. He has been working with Storage Virtualize and IBM SAN Volume Controller since 2002 (until 2011 as customer and since 2012 as a Business Partner employee). Uwe is an experienced professional providing technical pre-sales and post-sales solutions for IBM server and storage systems since 1995. He holds an engineering diploma in Computer Sciences from the University of Applied Science in Darmstadt, Germany.



Vasfi Gucer leads projects for the IBM Redbooks team, leveraging his 20+ years of experience in systems management, networking, and software. A prolific writer and global IBM instructor, his focus has shifted to storage and cloud computing in the past eight years. Vasfi holds multiple certifications, including IBM Certified Senior IT Specialist, PMP, ITIL V2 Manager, and ITIL V3 Expert.

Thanks to the following people for their contributions to this project:

Elias Luna
IBM USA

Lucy Harris, Evelyn Perez, Chris Bulmer
IBM UK

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

<https://www.linkedin.com/groups/2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/subscribe>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<https://www.redbooks.ibm.com/rss.html>



Introduction

In this chapter we discuss the following topics:

- ▶ “Introduction”
- ▶ “Example use cases”
- ▶ “Infrastructure as Code (IaC)”
- ▶ “Storage Virtualize automation opportunities”

1.1 Introduction

Automating the management of IBM Storage Virtualize systems (IBM Storage FlashSystem and IBM SAN Volume Controller) offers numerous benefits that can significantly enhance the efficiency and reliability of storage operations.

Here is a detailed look at these benefits:

1.1.1 Increased agility and faster storage provisioning

The following are the benefits of increased agility and faster storage provisioning.

Rapid deployment

Automation enables the quick deployment of storage resources, empowering organizations to adapt swiftly to evolving business needs. Instead of time-consuming manual provisioning, automation scripts and tools configure storage systems in minutes, ensuring agility and responsiveness.

Scalability

Automated storage management helps businesses scale effortlessly. As data needs evolve, storage scales automatically, preventing disruptions and ensuring consistent performance for your applications. This translates to uninterrupted business operations and a competitive edge.

Adaptability

Automation empowers IBM Storage FlashSystem to handle new workloads and applications with impressive speed. This streamlined approach ensures efficient storage support for your ever-changing business needs.

1.1.2 Reduced human error and improved consistency

Benefits of reduced human error and improved consistency are as follows.

Standardized procedures

Automation streamlines storage management tasks, guaranteeing consistent execution every time. Scripts and workflows eliminate the inconsistencies inherent in manual processes, leading to standardized configurations and smoother operations.

Error reduction

Manual storage management can be prone to human error, leading to downtime and data loss. Automation offers a reliable solution by minimizing errors through automating repetitive and complex tasks. This ensures consistent and reliable storage operations.

Compliance and best practices

Automation enforces adherence to industry standards and best practices, automatically ensuring all configurations meet predefined security and performance requirements. This simplifies compliance and minimizes risk for your storage environment.

1.1.3 Simplified management of complex storage tasks

The following are the benefits of simplified management of complex storage tasks.

Task automation

Automating complex storage tasks like replication, backups, and tiered storage management streamlines IT operations. This frees up valuable IT staff time, allowing them to focus on strategic initiatives that drive business value.

Centralized control

Automation tools transform storage management by offering centralized interfaces. This user-friendly approach allows administrators to effortlessly monitor and control multiple storage systems and tasks from a single console. The unified view simplifies management, saving time and minimizing potential errors.

Predictive maintenance and optimization

Automated systems go beyond basic automation by leveraging analytics and machine learning. This proactive approach predicts potential issues before they occur and optimizes storage performance for peak efficiency. Automated tuning and load balancing, for example, ensure optimal utilization of storage resources.

1.2 Example use cases

In this section we discuss several example use cases for automation.

1.2.1 Automated provisioning

Automation streamlines the process of onboarding new applications. When storage requirements emerge, automation tools automatically handle provisioning, configuration, and preparation – eliminating manual steps and ensuring a smooth workflow for IT teams.

1.2.2 Automated backup and recovery

Automation scripts streamline data protection by scheduling regular backups, verifying their integrity automatically, and guaranteeing the readiness of recovery processes. This eliminates manual intervention and human error, ensuring a reliable and efficient backup and recovery system.

1.2.3 Performance optimization

Automation ensures your applications always run smoothly. Automated systems continuously monitor and optimize storage performance, eliminating bottlenecks and dynamically allocating resources. This translates to consistently high performance and uninterrupted business operations.

1.2.4 Automating storage reporting

Automation can generate reports on storage usage, performance, and capacity trends.

1.2.5 Automated user access control

Automation can be used to manage user access to storage resources and enforce security policies.

In conclusion: Automating IBM Storage FlashSystem management delivers agility, speed, and reduced errors. This translates to efficient, reliable storage that empowers you to excel in today's data-driven world.

1.3 Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is a modern, foundational IT practice that involves managing and provisioning computing infrastructure through machine-readable configuration files, rather than through manual processes or interactive configuration tools. This approach allows for the automation and codification of the setup, configuration, and management of infrastructure.

By treating infrastructure configuration as code, organizations can achieve more reliable, efficient, and scalable IT environments.

The following are key concepts of IaC.

Declarative versus imperative

Declarative IaC describes the desired state of the infrastructure. Tools like Ansible, Terraform, and CloudFormation follow this model, where you specify what the end configuration should look like, and the tool figures out how to achieve it.

Imperative IaC describes the exact commands needed to achieve the desired configuration. This approach involves specifying step-by-step instructions.

Version control

IaC files are treated as code and stored in version control systems (for example, Git). This allows for tracking changes, collaboration, and rollback capabilities in case of issues.

Idempotency

Idempotent operations ensure that applying the same configuration multiple times will have the same effect, preventing unintended changes and maintaining consistency.

Reusability and modularity

IaC promotes the use of reusable and modular code, which simplifies the management of infrastructure by breaking it down into smaller, manageable pieces.

1.3.1 Role of IaC in automation

IaC has an important role in automation, as described below.

Consistency and reliability

IaC ensures that infrastructure is configured consistently across different environments (development, testing, production). This reduces the chances of configuration drift and human errors.

Speed and efficiency

Automation with IaC allows for rapid provisioning and de-provisioning of infrastructure, significantly speeding up deployment processes and reducing manual intervention.

Scalability

IaC enables the easy scaling of infrastructure. By defining configurations in code, additional resources can be provisioned automatically based on demand.

Improved collaboration

Since IaC files are stored in version control systems, they facilitate better collaboration among teams. Changes can be reviewed, tested, and approved in a controlled manner.

Documentation

IaC serves as a form of living documentation for the infrastructure. The code itself describes how the infrastructure is set up and managed, providing clear and up-to-date documentation.

1.4 Storage Virtualize automation opportunities

IBM Storage Virtualize systems can be including into your automation landscape by using different methods.

► Secure CLI with SSH access

IBM Storage Virtualize provides CLI for system's administration. It is available with SSH protocol and it can run bash-like scripts directly on the system. You can also include non-interactive SSH command calls into bash or PowerShell scripts running on the management host.

► REST API

Systems has as RESTful API server that can be accessed with HTTPS protocol. It provides a set of command targets, which allows passing arguments to Storage Virtualize command line interface. Full set of CLI commands is available through the REST API, which means that absolutely every storage management task can be performed with it.

► Ansible playbooks

IBM provides an Ansible Collection *ibm.storage_virtualize*, which is a set of Ansible modules and plugins for interacting with the IBM Storage Virtualize family products to be used in your Ansible Playbooks.

All the methods are available and consistent through all the Storage Virtualize portfolio, scripts or playbooks can be written once and used for all members of the family.



2

User management and security

This chapter provides an overview of system features for securing access to management interfaces, including GUI, CLI, and REST API. It covers the following topics:

- ▶ “System security introduction” on page 8
- ▶ “User management” on page 8
- ▶ “Superuser security” on page 13
- ▶ “Additional security with MFA and TPI” on page 16
- ▶ “TLS Certificates” on page 19

2.1 System security introduction

IBM Storage Virtualize features multiple levels of security to protect against threats and to keep the attack vector as small as possible:

- ▶ Strict verification features that prevent unauthorized users from using login interfaces and gaining access to the system and its configuration.
- ▶ Least privilege features that restrict the environment and limit any effect if a malicious actor managed to gain access to a system configuration interface.
- ▶ Minimal, locked down mode to prevent damage spreading to the kernel and the rest of the operating system.
- ▶ Protection of data at rest that is stored on the system from theft, loss, and malicious or accidental data corruption.

Security measures available on the system can be broadly divided into two categories: System security and data security.

2.1.1 System security

System security includes features that prevent unauthorized access to the system, protect the logical configuration of the storage system, and restrict what actions users can perform. System security also ensures visibility and reporting of system-level events that can be used by a Security Information and Event Management (SIEM) solution, such as IBM QRadar®.

These features include, but are not limited to, multifactor authentication (MFA), role-based access control (RBAC), object-based access control (OBAC), or disabling access to the command line interface (CLI), graphical user interface (GUI), and Representational State Transfer interface (REST) API.

2.1.2 Data security

This chapter focuses on system security features, specifically those related to administrative access protection and separation of duties.

These features ensure the system's data is protected against theft, loss, or attack. While data security features like Data-at-Rest Encryption (EDAR) and IBM Safeguarded Snapshot also play a crucial role, they are outside the scope of this chapter. Here, we will explore how to improve default configurations for enhanced security.

2.2 User management

This section discusses how to choose appropriate user settings and how to set security restrictions on them. Also advanced system security features such as two person integrity (TPI), multi-factor authorization (MFA) and single sign-on (SSO) are covered.

2.2.1 Users and roles

To administer, configure, and monitor the system, an authenticated user is required. The system supports local and remote users. Local users are defined on the system itself and managed internally by the system. This must be done on each individual Storage Virtualize

system. Remote users are defined and maintained in an authentication service external to the Storage Virtualize system, for example in your organization's LDAP directory.

Each user belongs to a user group. Each user group is assigned a role that is associated with a set of privileges. The user group privileges determine which configuration commands can be run by the users that are members of this group.

As a general best practice, every user must have the most restrictive role that allows him to fulfill his duties. For example, a user created to run scripts or make REST API calls solely for extracting performance statistics should be assigned to a group with *Monitor* permissions. Table 2-1 on page 9 shows use cases, user roles and their main characteristics.

Table 2-1 Choosing a user role

Use case	User/Role	Abilities
Read system's status and configured objects states, check system performance.	Monitor role	Can not make any changes on the system, can only run commands that show system information.
Read states and execute specialized tasks.	Copy operator, FlashCopy® administrator role	Same as monitor plus run commands that create or change objects related to the role.
Base system service and maintenance. <i>Not recommended for using with automation.</i>	Service role	Same as monitor, plus add/reboot/remove nodes and perform software update.
Daily administrative tasks.	Administrator role	Can perform most of the tasks, excluding user management. <ul style="list-style-type: none"> ▶ Create/Remove regular volumes and pools, hosts and mappings. ▶ Grow/Shrink volumes and pools. ▶ Can configure Safeguarded Snapshot function but can not remove Safeguarded snapshots.
Maintenance tasks; Administrative tasks without a permission to delete objects.	Restricted Admin role	All that Admin can do, but can not remove volumes, hosts or mappings.

Use case	User/Role	Abilities
User and authentication management; manage Safeguarded snapshots. <i>Not recommended for using with automation.</i>	Security Administrator role	All that an admin can do, plus: <ul style="list-style-type: none"> ▶ User management: create and delete user and user groups. ▶ Change system date time settings and time. ▶ Change security settings. ▶ LDAP server settings. ▶ Change certificates. ▶ Password rules. ▶ MFA settings. ▶ Change Safeguarded snapshot configuration. ▶ Delete Safeguarded snapshots.
Two person integrity when managing users and Safeguarded snapshots.	Security Admin + TPI = Restricted Security Administrator	Can do everything as a Security Admin, but requires approve from another Security Admin to complete risky or critical tasks.
Full control. <i>Not recommended for using with automation.</i>	superuser	Security Administrator with extra privileges. <ul style="list-style-type: none"> ▶ Highest authority. ▶ Completely unrestricted. ▶ Can run maintenance commands and perform initial setup. ▶ Can rebooting nodes in a failure state. ▶ Installing software in service state.

In addition to the role's restrictions, a user group can be configured to prohibit some of the access methods. For example, It is possible to configure a user group to accept only REST API calls, or restrict GUI access and leave only REST and CLI. For instructions, refer to [User management commands](#), `mkusergrp` and `chusergrp`, and to Example 2-1.

Example 2-1 Creating a user group with monitor role which allows only REST API access

```
IBM_FlashSystem:ITS0:securityadmin>mkusergrp -name REST_only -role Monitor
-disablegui yes -disablecli yes
```

```
Modifying the authentication setting for this user group will affect logins for
all users in the group. Are you sure you want to continue? (y/yes to confirm) y
User Group, id [6], successfully created
```

```
IBM_FlashSystem:ITS0:securityadmin>lsusergrp REST_only
id 6
name REST_only
role Monitor
remote no
owner_id
owner_name
multi_factor no
password_sshkey_required no
gui_disabled yes
cli_disabled yes
rest_disabled no
```

cim_disabled yes

Note: *superuser* built-in account does not follow settings for the default SecurityAdmin group and has separate controls to enable or disable GUI and REST API access, MFA and SSH key requirements. Refer to [chsecurity](#) CLI command for details.

A local user is a user whose account is managed entirely within the system itself. A local user can belong to one user group only. It must have a password, an SSH public key, or both. Each user has a username, which must be unique across all users in a system.

When you connect to the CLI by using an SSH client, SSH public key authentication is attempted first, with the username and password combination available as an alternative. The SSH key authentication method is available for CLI access and file transfer using Secure copy protocol (SCP). For GUI access, only the combination of username and password is used. Locally administered users can coexist with remote authentication.

Remote users are administered in the organization's LDAP directory infrastructure, where the actual authentication of remote users is done. The LDAP infrastructure can be a single, dedicated LDAP server, or part of a more complex environment, such as Microsoft Active Directory Services (MSAD) or Red Hat Directory Server.

For instructions on how to set up remote authentication, refer to [Remote Authentication](#) section in IBM Documentation.

Another way to delegate user management from the Storage Virtualize system is to configure authentication with Single Sign-On (SSO). In distinction to remote users authenticating with LDAP, SSO-authenticated users can log in only to GUI, and can not use CLI or REST API. See [Single Sign-On](#) in IBM Documentation for use difference between SSO and LDAP methods and for configuration instructions.

2.2.2 Password policies and account locking

While it is expected that Directory Services will take care about the password requirements and lifecycle for remote users, for local users a password policy can be configured. A policy is system-wide and applies to all users, including superuser. However superuser is immune to user account locking policy, unless it is explicitly enabled.

Only Security Administrator group member or Restricted Security Administrator with an active privilege escalation can set password policy.

The following attributes can be defined:

- ▶ Minimum password length: 6–64 characters.
- ▶ Minimum number of:
 - Uppercase characters: 0–3.
 - Lowercase characters: 0–3.
 - Special characters: 0–3.
 - Digits: 0–3.
- ▶ History check (0–10) before password reuse.
- ▶ Password expiry: 0–365 days.
- ▶ Password expiry warning (0–30 days) which is displayed on CLI at login only.
- ▶ Password age (1–365 days) which is the minimum age before a password can change.

A user with an expired password can log in to the system, but cannot run any commands that modify configuration (**svctask** group commands, or the commands that do not start with **1s***), until they change their password.

Complete the following steps to set the password policy in the GUI:

1. Select **Settings** → **Security**.
2. Select the **Password Policies** tab, as shown in Figure 2-1.

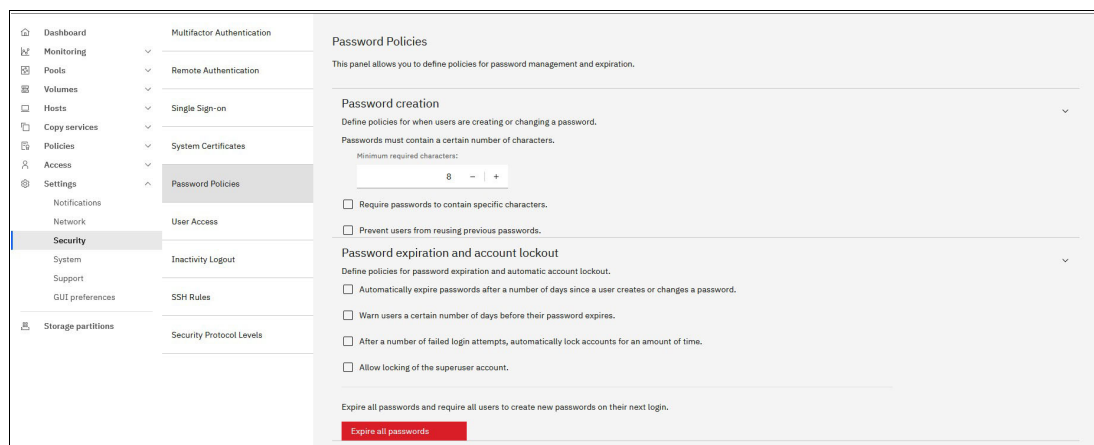


Figure 2-1 Configuring password policy

3. Set required password parameters in **Password creation** and **Password expiration and account lockout** sections. Additional input windows will be shown as you click on a feature enable checkbox.
4. After all attributes are set, click **Save**.
5. You can also command the system to expire all existing local user passwords immediately, so all users are forced to create a new password at their next login.

Alternatively, you can use CLI **chsecurity** command, as demonstrated in Example 2-2 on page 12.

Example 2-2 Using CLI to change password policy

```
IBM_FlashSystem:ITS0:securityadmin>chsecurity -checkpasswordhistory yes
-expirywarning 14 -maxpasswordhistory 6 -minpasswordage 1 -minpasswordlength 8
-passworddigits 1 -passwordexpiry 0 -passwordlowercase 1 -passwordspecialchars 1
-passworduppercase 1
```

The **chsecurity** command with the **maxfailedlogins** and **lockoutperiod** parameters controls user account locking. When a user exceeds the maximum number of allowed failed login attempts, their account gets locked, preventing them from logging in to the system. This lockout period determines how long the account remains inaccessible. If the lockoutperiod is set to 0, the account becomes permanently locked and requires manual unlocking by a Security Administrator.

Example 2-3 shows CLI commands for manual locking and unlocking of a user account.

Example 2-3 Locking and unlocking user accounts

```
IBM_FlashSystem:ITS0:securityadmin>chuser -lock Alice
IBM_FlashSystem:ITS0:securityadmin>chuser -unlock Bob
```


2.2.3 Setting session time-outs

The Security settings window also allows you to configure *Inactivity Logout* settings. These settings are independent for the CLI and GUI, letting you define the time (in minutes) after which a user is automatically logged off from the respective interface.

This same interface panel also manages the validity period of REST API security tokens. Once a token expires, it becomes invalid and requires regeneration for continued API access.

Figure 2-2 on page 13 shows the settings including the allowed values:

- ▶ A configurable CLI time-out of 5–240 minutes
- ▶ A configurable GUI time-out of 5–240 minutes
- ▶ A REST API token time-out of 10–120 minutes

In CLI, `chsecurity` command with `gui timeout`, `cli timeout` and `restapi timeout` parameters is used to control user authorisation time-out settings.

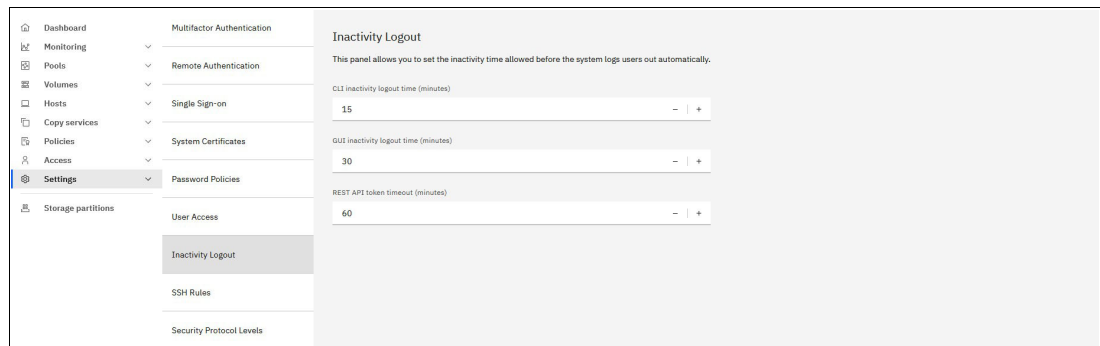


Figure 2-2 Setting session time-outs and REST API token time-out

2.3 Superuser security

During initial configuration, the system defines a single local Security Administrator user that is called a *superuser*.

For cluster maintenance and recovery tasks performed through Service Assistant, only the superuser login account has the necessary permissions. This superuser account has a higher level of access than a Security Administrator.

2.3.1 Superuser specifics

Superuser is always present on the system and can not be deleted or configured for remote authentication, but it can be disabled (not allowed to log in) for higher system security, and temporarily re-enabled back when required.

A superuser password must be changed immediately after the cluster is created. System enforces the change and will not allow you to perform configuration before it is done. Superuser password can't be set or reset to match the default value until cluster configuration is intentionally destroyed. It is crucial to set a strong password and to store it securely.

If you choose to leave the superuser account enabled, implement the strongest possible security measures to protect it. Use a complex password and limit knowledge of the password.

Superuser is the only account that can run commands with Service Assistant prefixes, **satask** and **sainfo**. Tasks that can be performed exclusively by superuser are:

- ▶ Initial system setup
- ▶ Rebooting nodes in a failure state
- ▶ Installing software in a failure or service state
- ▶ Cluster recovery (T3 recovery)

For other superuser Service Assistant tasks, equivalent cluster (**svctask** or **svcinfo**) command exists. Refer to Table 2-2 for a list of commands that can be used to replace superuser-only commands in Storage Virtualize 8.6.0 and later.

Table 2-2 Svctask/svcinfo replacement for satask/sainfo commands

Task	Service Assistant command	Cluster command
Change or set node's Service Assistant IP address (service IP)	satask chserviceip	chnodeserviceip chnodecanisterserviceip svctask chnodeserviceip Runs on active config node unless -node parameter is specified.
Put a node into a service state	satask startservice	stopsystem -enterservicestate Requires -node parameter.
Exit from a service state to candidate state or to return to cluster	satask stopservice	startsystem -exitervicestate Requires -node parameter.
Restart one of the services running on the node without affecting I/O processing on that node	satask restartservice	restartservice Runs on active config node unless -node parameter is specified.
Check node status and VPD data.	sainfo lsservicestatus	l1nodestatus l1nodecanisterstatus Runs on active config node unless -node parameter is specified.

Tip: Cluster commands include the word “node” on SVC and “nodecanister” on enclosure-based systems, for example, **l1nodestatus** and **l1nodecanisterstatus**. By adding the optional prefix **svctask** or **svcinfo** to the SVC commands (for example, **svctask l1nodestatus**), you can make them work on both SVC and enclosure-based platforms.

Superuser is a member of the default SecurityAdmin user group. However access settings for the group do not apply to superuser: for example, even if you set up SecurityAdmin group to reject GUI logins, superuser will still be able to do it.

2.3.2 Locking and unlocking the superuser account

As an additional measure to increase the security level and to minimize the attack surface, the superuser account can be locked. A good use case is remote authentication-only system: configure remote authentication, create a remote security administrator, and disable the local superuser. Now, no local accounts can log in to the system.

Note: Disabling the superuser account and session time-outs are available only on platforms with a dedicated Technician port.

Locking superuser account is a two-step process. First, you need to enable superuser locking, then run a command to lock it. Steps to disable the superuser account by using the CLI are shown in Example 2-4:

Example 2-4 Disabling superuser account

```
IBM_FlashSystem:ITS0:security_admin>chsecurity -superuserlocking enable
Changing the system security settings could result in a loss of access to the
system via SSH or the management GUI. Refer to the Command Line Interface help for
more information about the risks associated with each parameter. Are you sure you
wish to continue? (y/yes to confirm) yes
IBM_FlashSystem:ITS0:security_admin>chuser -lock superuser
```

A user cannot lock their own account, so the second step must be done from a non-superuser Security Administrator account.

Note: If superuser locking is enabled, but the account is not explicitly locked, superuser will follow the defined password policy for account expiration and locking. This means it can be locked automatically, for example, after several unsuccessful login attempts.

To unlock superuser:

- ▶ Log in with another Security Administrator user account. Such an account must be configured prior to locking superuser. It can be a local account with additional security measures (for example, configured to require both password and SSH key, or password and MFA), or a remote account (LDAP).

There are also emergency unlock methods:

- ▶ Unlock by logging in through the dedicated Technician port. This method requires physical access to the hardware.
- ▶ If Remote Support Assistance is configured, IBM support engineer can unlock superuser. See [IBM Documentation](#) for instructions on how to configure support assistant feature.

2.3.3 Superuser password reset

If superuser password is lost or forgotten, another Security Administrator can change it.

In case it is not possible to log in with a remote or local Security Administrator account, you need physical access to the system hardware to reset superuser password. You will see superuser password reset link when accessing the system Service Assistant GUI through the Technician port. Alternatively, you can use [USB flash drive](#) interface with **satask resetpassword** command. This method is also suitable for systems that do not have a dedicated Technician port. With the command you must specify new password and you can not use the default one.

If superuser account is locked, it will be automatically unlocked when the password is reset.

Storage Virtualize offers the option to disable the superuser password reset function. However, this decision requires careful consideration. If superuser password is lost, there are no other Security Administrators, and password can not be reset, then there is no way recover superuser access to the cluster. In such a scenario, recovering access would necessitate:

1. Migrating data off the system using host-side tools.
2. Destroying the existing cluster.
3. Recreating the cluster from scratch.

This process can be time-consuming and disruptive.

If password_reset_enabled key in `sainfo lsservicestatus` (or `svcinfo lsnodestatus`) shows “no”, there will be no option to reset superuser password with USB interface or technician port. Refer to Example 2-5 for CLI commands to disable password reset function.

Example 2-5 Working with password reset disable feature

```
# check if password reset is allowed
IBM_FlashSystem:ITS0:superuser>setpwdreset -show
Password status: [1]
IBM_FlashSystem:ITS0:superuser>lsnodecanisterstatus | grep reset
password_reset_enabled yes
# disable superuser password reset and check the result
IBM_FlashSystem:FlashSystem:superuser>setpwdreset -disable
IBM_FlashSystem:FlashSystem:superuser>setpwdreset -show
Password status: [0]
IBM_FlashSystem:FlashSystem:superuser>lsnodecanisterstatus | grep reset
password_reset_enabled no
# enable password reset
IBM_FlashSystem:FlashSystem:superuser>setpwdreset -enable
```

Note: If you disable password reset and you do not have local or remote Security Administrator users except superuser, there is no way to restore management access if superuser password is lost.

2.4 Additional security with MFA and TPI

Multifactor Authentication (MFA) and Two person Integrity (TPI) are security features that can be configured to strengthen system security.

Multifactor authentication requires users to provide multiple pieces of information when they log in to the system to prove their identity. MFA can be configured for both remote (LDAP) and local user, it can also be configured for superuser. It supports both CLI and GUI interfaces, but can not be used with REST API.

Two Person Integrity (TPI), also known as the four-eyes-principle, makes a defined set of administration tasks require agreement between two administrators, so a single person, such as a rogue user with administration permissions, can not perform them. TPI limits only several tasks that are normally performed by Security Administrator and works in GUI and CLI, and also does not affect REST API usage.

When TPI is enabled, superuser account is automatically locked.

2.4.1 Multifactor Authentication

In addition to username and password, which is something a user *knows*, a second factor is required to log in when MFA is enabled. The second factor can be, for example, an application or a device generating a one-time passcode (OTP). The second factor can also be a biometrical characteristics such as a fingerprint or a retinal scan.

For a list of supported MFA providers, refer to the [Supported Authentication Providers](#) page on IBM website.

MFA feature uses OpenID Connect (OIDC) authentication protocol for GUI authentication and providers API for CLI access.

Detailed configuration instructions, including configuration required from the authentication provider side, are available in the [Multifactor-Authentication](#) section in IBM Documentation. Overall, configuration workflow is the following:

- ▶ Configure DNS (unless you plan to specify authentication server by IP address).
- ▶ Allow network access from a system to provider's endpoint (directly or via proxy).
- ▶ Verify that inactivity time-outs for GUI and CLI are longer than the time needed to receive OTP password.
- ▶ Receive MFA provider endpoint address and credentials.
- ▶ Configure MFA provider endpoint and credentials for GUI and CLI on the system.
- ▶ Configure existing user groups or create new user groups to use MFA.
- ▶ If required, enable MFA for superuser.
- ▶ Check MFA fallback policy.

MFA fallback policy defines system behavior in a situation, when authentication provider can not be contacted. It can be set to allow or deny users, configured to use MFA. By default, it will deny logins, which means that if MFA is configured for all users including superuser, and communication to the provider fails, there is no way to access system's management interfaces without physical access to the hardware. With the hardware access, superuser is able to log in with the technician port, regardless of the MFA configuration.

2.4.2 Two person integrity

When TPI is enabled, all users with the Security Administrator role are changed to the Restricted Security Administrator role, and *superuser* account gets automatically locked. When a configuration task requires privileges that are exclusive to Security Administrator, Restricted Security Administrator requests temporary privilege elevation, which must be approved by a second user with Restricted Security Administrator role. This privilege elevation is limited to a maximum time allowed of 24 hours.

Demonstration video: To view a demonstration video, see [Configuring Two Person Integrity with IBM Storage Virtualize V8.6](#).

Note: Only local users and users authenticated through LDAP Remote Authentication with Security Administrator privileges can request a role elevation. This functionality is not available to users who are authenticated through SSO, even if they have Security Administrator privileges within SSO.

Enabling TPI is restricted to IBM Storage Virtualize systems featuring a Technician Port. This is to avoid users locking themselves out of the system because a locked superuser account can be unlocked from the Technician Port interface.

At least two local users with Security Administrator role (excluding superuser), or a remote user group with this role, must be set up before TPI is configured.

The set of tasks that require privilege elevation is fixed and includes system-critical and security-related tasks:

- ▶ Create or change users or user groups.
- ▶ Change security settings related to LDAP, MFA, SSO.
- ▶ Change certificates.
- ▶ Change the system time or NTP settings.
- ▶ Remove and change Safeguarded backups and Safeguarded backup locations.
- ▶ Delete Safeguarded snapshots.
- ▶ Remove the Safeguarded snapshot policy association from a volume group.

Configuration instructions are provided in the [Two person integrity](#) section in IBM Documentation.

Note, that if you are logged in as superuser to enable the feature, you will not be able to continue as the changes are applied immediately, as Example 2-6 on page 18 shows.

Example 2-6 Superuser loses access as soon as TPI is enabled

```
IBM_FlashSystem:ITS0:superuser>chsecurity -twopersonintegrity yes
Changing the system security settings could result in a loss of access to the
system via SSH or the management GUI. Refer to the Command Line Interface help for
more information about the risks associated with each parameter. Are you sure you
wish to continue? (y/yes to confirm) y
IBM_FlashSystem:ITS0:superuser>lscurrentuser
CMMVC6510E The task has failed because the user name or password is not correct.
```

When two person integrity feature is enabled, all Security Administrator users are switched to Restricted Security Administrator role. Restricted SecAdmin can issue a request for privilege elevation. Another Restricted SecAdmin approves this request, but can limit approved time, if necessary. After request is approved, role immediately changes to Security Administrator. See Example 2-7 for workflow.

Example 2-7 The workflow

```
IBM_FlashSystem:ITS0:rsecadmin>lscurrentuser
name rsecadmin
role RestrictedSecurityAdmin
...
IBM_FlashSystem:ITS0:rsecadmin>mktwopersonintegrityrequest -minutesrequested 90
Two person integrity request, id [0], successfully created

### Another Restricted SecAdmin logs in and approves, but sets shorter limit
IBM_FlashSystem:ITS0:approver>lscurrentuser
name approver
role RestrictedSecurityAdmin
...
IBM_FlashSystem:ITS0:approver>lstwopersonintegrityrequest
request_id user_name      is_remote minutes_requested minutes_approved ....
0          rsecadmin      no         90                 0                 ....
IBM_FlashSystem:ITS0:approver>chtwopersonintegrityrequest -approve
-minutesapproved 30 -username rsecadmin
IBM_FlashSystem:ITS0:approver>lstwopersonintegrityrequest
request_id user_name      is_remote minutes_requested minutes_approved ....
```

```
0          rsecadmin      no          90          30          ....
```

```
### Role of a user who requested elevation changes
IBM_FlashSystem:ITS0:rsecadmin>lscurrentuser
name ITS0:rsecadmin
role SecurityAdmi
```

2.5 TLS Certificates

Note: TLS has succeeded the deprecated Secure Sockets Layer (SSL) protocol. However, TLS certificates are commonly still referred to as SSL certificates. Therefore, the terms TLS and SSL can be used interchangeably.

During system initialization, a Transport Layer Security (TLS) certificate is automatically generated and signed by the system-internal root certificate authority (CA).

The purpose of the internal Root CA of an IBM Storage Virtualize system is to add its certificate to truststores on systems communicating through TLS-encrypted paths. This can include administrator workstations that are used to access the management GUI, Encryption Key Servers or other Storage Virtualize systems in a Policy Based Replication partnership. The advantage of applying the internal root CA certificate over using the system's certificate is that the communication can seamlessly continue between these systems, even if the certificate was renewed after it had expired.

Your organization's policy might restrict usage of certificates, not signed by a trusted Root CA, which can be either an internal one, for example part of your organization's own Public Key Infrastructure (PKI), or a public trusted CA such as GlobalSign, Let's Encrypt, and others. Web browsers maintain a list of trusted CAs that are identified by their root certificate. The root certificate must be included in this list for the signed certificate to be trusted, so no security warning will be raised.

Organization-internal Root CAs are usually configured as trusted authorities across your company.

To see the details of your system's certificate in the GUI, select **Settings** → **Security** and then click **System Certificates**, as shown in Figure 2-3. In the CLI, you can use the command `lssystemcert`.

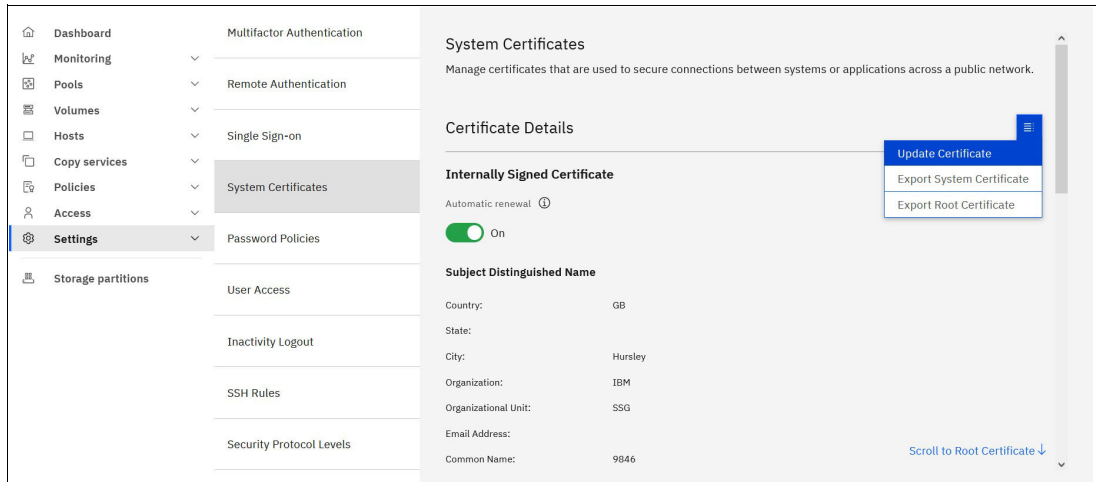


Figure 2-3 Accessing the System Certificates window

Depending on your organization's security policies and needs, you can create a new certificate, which is a self-signed by the system. Another option is to create a Certificate Signing Request (CSR), which can be submitted to be signed by a CA within your own organization or by an external, trusted CA. Also, beginning with Storage Virtualize V8.5.3, the option to use a system-internal root CA was added, which can be used to sign certificate signing requests created for this system.

An internally signed system certificate can be renewed automatically 30 days before its expiration date, if automatic renewal is enabled. You can also re-generate it manually, key type requirements or any of the other certificate properties need to be changed. For instructions, refer to [Updating internally signed certificate](#) in IBM Documentation.

If your company's security policy requests certificates to be signed by a trusted certificate authority (CA), check instructions in the [Requesting and installing externally signed certificate](#) section in IBM Documentation. The workflow is:

1. Prepare certificate sign request (CSR) using system's GUI or CLI. For filling the information, use tips provided in IBM Documentation article linked above. For the Country field, use a two-letter country code according to [ISO 3166-1 alpha-2](#).
2. Sign CSR with your CA. The resulting generated certificate must be saved in PEM format. Whole certificate chain (endpoint certificate, intermediate CA certificate(s)) must be concatenated together in a single file. The root certificate can optionally be excluded. Concatenation can be done in any order. Example 2-8 shows an example if a file in base64-encoded PEM format that the system expects.

Example 2-8 TLS Certificate chain in PEM format

```
-----BEGIN CERTIFICATE-----
MIIFODCCBCCgAwIBAgIIRNDAZI11U80wDQYJKoZIhvcNAQELBQAwwGZgCzAJBgNV
....
1U3o4s0+Tb712Wd9HZT/xXvzibK023cryRwGVsGtBcGg+9x1oIa0be5VQFw=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
BAYTAKFUMQ0wCwYDVQQIEwRXaWVudQ0wCwYDVQQHEwRXaWVudQ0wCwYDVQQKEwRC
....
IvY/T3hzJz15ZdxoowCDIsHj1URk9RDSV/ccgw/E3Ig5E0IE69E00ErnMhB=
-----END CERTIFICATE-----
```


3. Install certificate chain on the Storage Virtualize system.
4. If required, export certificate and install it to the services that must be aware of the updated system's certificate (for example, if MFA with IBM Security® Verify is in use).



Automation with CLI scripting

This chapter empowers you to manage IBM Storage Virtualize systems for seamless integration into your automation and management processes. It dives deep into advanced command-line interface (CLI) usage best practices and unveils the power of CLI commands for writing and running complex scripts, streamlining your storage operations.

Sections in this chapter are:

- ▶ “Secure restricted shell” on page 24
- ▶ “General tips” on page 24
- ▶ “Internal scripting commands” on page 27
- ▶ “Script examples” on page 28

3.1 Advanced CLI usage and scripting

This section describes management methods that can be used to control and monitor the IBM Storage Virtualize system by using command line interface and basic scripts.

Scripting can be used to automate management tasks or for reporting and monitoring. For complex management tasks, using Ansible or REST API is preferred.

3.1.1 Secure restricted shell

SSH is the only supported method for accessing system's CLI. It is a network protocol that enables secure communication and operation over an insecure network.

To authenticate on the system, login and password or SSH key pair is used. For scripting purposes, SSH key is the recommended method.

IBM Storage Virtualize provides SSH clients a functional Unix bash-like shell, running in restricted environment. General CLI usage instructions are provided in [IBM Documentation](#).

Interactive shell

The use of an SSH client within a shell is a common way of running commands. When using interactive shell, it is still possible to use scripts, but there is no way to save script source code or script outputs on the system itself.

Non-interactive shell

Instead of opening an interactive session, you can use SSH client to run a command on the remote system, and forward the output to the local computer.

Leveraging SSH for remote command execution empowers you to write powerful scripts for IBM Storage Virtualize. These scripts can:

1. Collect data from the storage system.
2. Transfer the collected data to your local machine.
3. Process the data on your local computer using a wider range of tools and utilities available at your disposal.

Example 3-1 shows how to use SSH to run a command (`svcinfo lssystem`) on the IBM Storage Virtualize system (`mystorage`) with the privileges of `myuser` and piping the output to the local system to filter only a line that shows system's code version.

Example 3-1 Using ssh and grabbing selected information

```
$ ssh -i /path/to/key -o "BatchMode yes" myuser@mystorage 'svcinfo lssystem' | grep code
```

Batch mode is recommended for scripts running in unattended or non-interactive mode. When it is enabled, system will try to authenticate using a provided key, but will not fallback to password authentication if it is failed. So an unattended script will not hang on password request, but generate error and exit.

3.1.2 General tips

Consider the following general tips when writing SSH scripts.

Implement error checking

While error checking might seem unnecessary for simple scripts, it becomes crucial for complex, multi-step ones.

Effective error checking and handling allow the script to adapt its behavior in case of issues. This includes:

- ▶ **Stopping execution:** If an SSH connection problem arises, the script can terminate gracefully.
- ▶ **Modifying execution logic:** If the IBM Storage Virtualize response indicates an error, the script can adjust its subsequent steps accordingly.

By incorporating error handling, you ensure your scripts are robust and can handle unexpected situations.

In both interactive and non-interactive shells, IBM Storage Virtualize uses exit codes to indicate command success or failure:

- ▶ **Exit code 0:** Successful command execution.
- ▶ **Exit code 1:** A CMMVC error message was encountered.
- ▶ **Exit codes 2-225:** Other execution failures.

By checking the exit code, scripts can define logic to handle errors appropriately, such as taking specific actions or stopping script execution.

When running a command via SSH from a host over a non-interactive shell, the host will correctly display the exit code. Any error text will be sent to the host's standard error (STDERR).

Example 3-2 shows command exit codes.

Example 3-2 Command exit codes

```
IBM_FlashSystem:ITS0:redbooks>lsvdisk 0
CMMVC5753E The specified object does not exist or is not a suitable candidate.
IBM_FlashSystem:ITS0:redbooks>echo $?
1
IBM_FlashSystem:ITS0:redbooks>svdis 0
rbash: lsvdis: command not found
IBM_FlashSystem:ITS0:redbooks>echo $?
127
....
[user@host] $ ssh redbooks@9.18.77.111 lsvdisk -badparameter
CMMVC5709E [-badparameter] is not a supported parameter.
[user@host] $ echo $?
1
```

Prefixes svcinfo and svctask

In the past, all IBM Storage Virtualize cluster commands required prefixes like **svcinfo** or **svctask**. While these prefixes are now optional in all supported releases, it is recommended to keep using them for your scripts. This practice ensures better compatibility and readability.

There might be slight variations in CLI commands between different IBM Storage Virtualize products. For instance, the **1snode** command is not available in IBM FlashSystem, which uses **1snodecanister** for the same functionality.

Recommendation: To maintain compatibility across different storage systems, using `svcinfo lsnode` is recommended. This command works on both IBM FlashSystem and SAN Volume Controllers. If you are specifically targeting IBM Storage Virtualize controllers, using `lsnode` alone is acceptable.

Generally, cluster commands that return information on the current state or start with `ls`, can use `svcinfo` prefix. Commands that modify current state of the system, use `svctask` prefix.

Note that service-level task and information commands require `sainfo` and `satask` prefixes, it can not be omitted or replaced with `svcinfo/svctask`.

Selectable delimiter for outputs

By default, output fields in `ls*` commands are separated by a single or multiple spaces, making them easier for a human to read, but not convenient for automated parsing.

`ls-commands` support `-delim` parameter, which allows to provide a single, selectable character that will be inserted between columns of the output instead of spaces. The colon (`:`) or tilde (`~`) are good choices for a delimiter, as they can not be found on most of the data fields and therefore will not break result parsing.

Extended data tables

Commands displaying a list of objects, such as `lsdrive` or `lsdisk`, contain only major object's details. To see more, you have to use same command with the object's ID or name as an argument, but you will need to issue a command against each particular object.

Many `ls` commands in IBM Storage Virtualize can leverage the `-gui` parameter. This parameter allows you to retrieve an extended data table similar to the output displayed in the IBM Storage Virtualize GUI. It provides more detailed information for all objects of the requested type. For a clear comparison, refer to Example 3-3, which showcases the output headers of the `lsdrive` command with and without the `-gui` parameter.

Example 3-3 *lsdrive* outputs without and with `-gui` parameter

```
IBM_FlashSystem:ITS0:redbooks>lsdrive -delim '~'
id~status~error_sequence_number~use~tech_type~capacity~mdisk_id~mdisk_name~member_
id~enclosure_id~slot_id~node_id~node_name~auto_manage~drive_class_id
...
IBM_FlashSystem:ITS0:redbooks>lsdrive -delim '~' -gui
id~status~error_sequence_number~use~UID~tech_type~capacity~block_size~vendor_id~pr
oduct_id~FRU_part_number~FRU_identity~RPM~firmware_level~FPGA_level~mdisk_id~mdisk
_name~member_id~enclosure_id~slot_id~node_id~node_name~quorum_id~port_1_status~por
t_2_status~was_spare~interface_speed~auto_manage~drive_class_id~write_endurance_us
ed~transport_protocol~compressed~physical_capacity~physical_used_capacity~effectiv
e_used_capacity~date_of_manufacture~protection_enabled~write_endurance_usage_rate~
replacement_date~anomaly_detection_active
...
```

Remove unwanted information

The `-nohdr` parameter eliminates the need for code to skip the header line in the `ls` command output. In addition, when used with the `-nomsg` parameter, object creation commands like `mkvdisk` will return only the created object ID and skip the “*successfully created*” message. You can see the output comparison in Example 3-4. This method simplifies parsing the object creation command results and allows you to reuse the object ID later in your script.

Example 3-4 mkvdisk command with -nomsg parameter

```
IBM_FlashSystem:ITS0:redbooks>mkvdisk -mdiskgrp 0 -size 1 -unit gb
Virtual Disk, id [3], successfully created
IBM_FlashSystem:ITS0:redbooks>mkvdisk -mdiskgrp 0 -size 1 -unit gb -nomsg
4
```

Control command rate

Running frequent complex command queries, especially on systems with thousands of configured objects, can be resource consuming and therefore not recommended. Running CLI commands that return hundreds of lines of the output multiple times per second can impact system's management.

3.1.3 Internal scripting commands

Scripts can be run on the administrator's system, where they can prepare and potentially send CLI commands to the storage system. Alternatively, scripts can be written directly within the IBM Storage Virtualize CLI environment. When writing scripts directly on the storage system, you have access to a set of internal bash and system commands.

Key point: This flexibility allows you to choose the approach that best suits your needs. You can leverage your existing scripting environment or write scripts directly on the storage system using the provided internal commands.

Loops: for and while

CLI supports **while** and **for loops**. Loop structure is shown in Example 3-5.

Example 3-5 Loop structure

```
IBM_FlashSystem:ITS0:redbooks>i=0; while [ $i -lt 5 ]; do i=$((i+1)); echo $i;
done
IBM_FlashSystem:ITS0:redbooks>for loop_count in {1..5}; do echo $loop_count; done
```

While loop runs until the given statement is no longer true. Example 3-5 shows a loop which prints numbers 1 to 5.

For loop provides the same output. It iterates over the provided set of values and changing the loop variable with each iteration. This set can be defined in several ways:

- ▶ **Range:** Using curly braces {start..end} to specify a range of values (for example, *for i in {1..5}; do ...*).
- ▶ **Space-separated list:** Separating values with spaces (for example, *for i in Monday Tuesday Wednesday; do ...*).
- ▶ **Command substitution:** Executing a command and using its output as the list (for example, *for i in \$(lsdrive -nohdr -delim : | cut -d : -f 1); do ...*).

Both type of loops can be stopped with the Ctrl-C combination.

Data parsing

There is a set of data filtering and parsing tools available in the CLI.

- ▶ **grep** is a GNU grep command implementation. When a command output is piped to grep, it will process it string by string and search for the specified pattern.
- ▶ **cut** is a GNU cut, it is used to print only selected parts of the line with delimiters;

- ▶ **sort**, GNU coreutils sorting tool;
- ▶ **wc**, GNU tool to print newline, word, and byte counts in the input stream;
- ▶ **sed**, powerful GNU line-oriented text processing utility.

For available command options, issue the command with **--help** parameter in the system's CLI, or refer to the appropriate Unix documentation.

3.1.4 Script examples

Use the examples below to write IBM Storage Virtualize scripts that fit your purpose. Scripts are shown in multi-line notation for better readability; they can be submitted as-is or in a one-line notation.

Example 3-6 shows a script that creates four 100 GB vdisks with the provided names and immediately maps them to the existing host with ID 0.

Example 3-6 Vdisk creation and mapping script

```
for name in Prod_DB1 Prod_DB2 Test_DB1 Test_DB2;
do
    id=$(svctask mkvdisk -mdiskgrp 0 -size 100 -unit gb -name $name -nomsg);
    mkvdiskhostmap -host 0 $id;
done
```

Example 3-7 on page 28 showcases a script that monitors the system code update process.

Here is how it works:

1. The script enters an infinite loop.
2. Within the loop, it uses **grep** to search the output of the **lsupdate** command for a line that begins with "status". This line indicates the current update state.
3. The script then uses **cut** to extract the current update status value, which is the second field in the line separated by a colon (":").
4. The script displays the current state along with the cluster time.
5. If the status indicates successful completion, the script exits the loop.
6. If the update is not yet complete, the script waits for 15 seconds before starting the next iteration.
7. You can manually stop the script using Ctrl-C if the update takes longer than expected.

Example 3-7 Process tracking script

```
while true;
do
    result=$(lsupdate -delim : | grep ^status | cut -d : -f 2);
    echo $(svqueryclock) - $result;
    [[ "$result" == "success" ]] && break;
    sleep 15;
done
```

Example 3-8 demonstrates a script that extracts "Used Capacity" data for virtual disks (vdisks), which isn't available in the standard or even extended object listing view.

Here is how the script works:

1. It displays a table header that includes the "Used Capacity" column.

2. Then, it retrieves a list of all vdisk IDs using a command.
3. The script iterates through each vdisk ID.
 - a. For each ID, it retrieves detailed vdisk properties, including capacity information in bytes.
 - b. It uses **grep** to filter the output and isolate the desired capacity parameters.
 - c. **sed** is used to extract these parameters and assign them to separate variables.
4. Before displaying the results, the script converts the capacities from bytes to Gibibytes for a more user-friendly format.

Example 3-8 Volume capacity report script

```
echo ID:Name:Capacity:Used_capacity;
lsvdisk -nohdr | while read -r id rest;
do
    details=$(svcinfolsvdisk -bytes -delim : $id | grep -E
'^((name|capacity|used_capacity):' | cut -d : -f 2);
    name=$(echo "$details" | sed -n 1p);
    capacity=$(echo "$details" | sed -n 2p);
    used=$(echo "$details" | sed -n 3p);
    echo "$id:$name:$((capacity/1024/1024/1024)):$((used/1024/1024/1024))";
done
```



4

Automation with REST API

This chapter provides methods of managing IBM Storage Virtualize systems that can be used to integrate the system in your automation and management processes.

It describes REST API interface and provides several examples of how to interact with it.

Sections in this chapter are:

- ▶ REST API on IBM Storage Virtualize
- ▶ REST API on IBM Storage Virtualize

4.1 REST API on IBM Storage Virtualize

The IBM Storage Virtualize Representational State Transfer (REST) application programming interface (API) consists of command targets that are used to retrieve system information and to create, modify, and delete system resources. These command targets allow command parameters to pass through unedited to the IBM Storage Virtualize command line interface (CLI), which handles parsing parameter specifications for validity and error reporting. Hypertext Transfer Protocol Secure (HTTPS) is used to communicate with the REST API server.

API limits

Rate limiting helps with security and the prevention of an attack, such as a denial of service in which unlimited work is sent to the system. The rate limiting is implemented at second granularity and creates a return code (429 - too many requests) when a violation occurs. REST API rate limits are listed in the documentation [Storage Virtualize RESTful API](#).

4.1.1 REST API Explorer

The REST API Explorer is based on the Swagger UI and runs within a browser. It offers an easy way to become familiar with the API and to test the commands that it contains.

The use of the REST API Explorer requires authentication with login and password, that are used for generating a token, which is needed for all further actions. Figure 4-1 shows how to create an authentication token within the IBM SAN Volume Controller information and IBM SAN Volume Controller Task actions.

To access the REST API Explorer, enter the following URL in a browser:

```
https://<management_ip>:7443/rest/explorer
```

To view the REST API documentation, see [Storage Virtualize RESTful API](#). Support can also be found directly on the system within the REST API Explorer.

For the command, the following definitions apply:

- ▶ `-k` parameter is required if self-signed certificates are used on the Storage Virtualize system;
- ▶ `-L` parameter allows curl to follow redirects;
- ▶ `-X POST` instructs curl to use the only HTTPS method that the Storage Virtualize REST API supports;
- ▶ Headers `<header_1>` and `<header_2>` are individually specified HTTP headers, such as, Content-Type and X-Auth-Username.
- ▶ Parameter `-d` followed by the input in Java script Object Notation (JSON) is used to provide command parameters, if they are required. For example, `{“raid_level”: “raid5”}`.
- ▶ `<StorageVirtualize_ip_address>` is the IP address of the IBM Storage Virtualize system to which you are sending requests.
- ▶ `<api_version>` specifies which version of the API should get used. v1 is recommended.
- ▶ `<target>` is the target command, execution of which is requested.
- ▶ `<ID>` is an object id, against which a command given in `target` is executed.

Note: Compatibility with an earlier version is implemented by auto redirection of non-versioned requests to v0. It is recommended to use versioned endpoints for guaranteed behavior.

Authentication

Aside from data encryption, the HTTPS server requires authentication of a valid username and password for each API session. It is required to specify two authentication header fields for your credentials: X-Auth-Username and X-Auth-Password.

Initial authentication requires that you POST the authentication target (`/auth`) with the username and password. The REST API server returns a JWT token. The token remains authorized only for the configured time, 10 to 120 minutes. You can also set a limit for how long an unused token is valid

Example 4-1 shows a process of getting an authentication token using login and password, parsing received json with jq, and save token in the variable for further use:

Example 4-1 Authenticating with REST API

```
$ token=$(curl -k -L -X POST -H 'X-Auth-Username: rest_user' -H 'X-Auth-Password:
passw0rd' https://itso.lab:7443/rest/auth | jq -r '.token')
$ curl -k -L -X POST -H X-Auth-Token:$token https://itso.lab:7443/rest/v1/lssystem
{ "id": ..... "on" }
```

The X-Auth-Token header in combination with the authentication token replaces the username and password for all further actions.

REST API query with parameters and arguments

Most actions can be taken only after authentication. To see a list of available actions and parameters they require and return, refer to Swagger endpoint interface, available on your system at `https://<management_ip>/rest/explorer/`.

Technically, for almost every cluster-level (i.e. commands that do not need `sainfo` or `satask` prefixes) there is a REST target. Also, with REST API it is possible to download files such as

performance statistics, IP quorum apps and support packages (snaps) from the node's internal locations.

Review the following examples to see how commands can be converted to REST API queries. According to the command syntax shown in Example 4-2, command requires one parameter **-node**, and one argument, which is a numeric ID of the port, which is port id 2 here.

Example 4-2 CLI command syntax

```
IBM_FlashSystem:ITSO:redbooks>lsportstats -?
..
>>-lsportstats-- -node--node_id_or_name--+-+-----+-----><
                                         '-port_id-'

IBM_FlashSystem:ITSO:redbooks>lsportstats -node node1 2
... output truncated ...
```

To translate that into REST API query, parameters must be provided in a json notation. Also, a *Content-Type* header to specify the kind of provided POST data is needed. Command's argument is translated to the endpoint ID and specified in the URL. Example 4-3 demonstrates this.

Example 4-3 Supplying parameters to REST API query

```
$ curl -k -L -X POST -H X-Auth-Token:$token -H 'Content-Type: application/json'
https://itso.lab:7443/rest/v1/lsportstats/2 --data-raw '{"node": "node1"}'
[{"... output truncated ..."}]
```

The same approach is used if you want to refer an object by its name instead of the id.

If you need to supply parameters to a command that do not require a value, such as **-bytes**, they need to be set to true in the json.

See Example 4-4 showing these concepts.

Example 4-4 Requesting object details by its name with REST API

```
IBM_FlashSystem:ITSO:redbooks>lsmdisk -bytes MDisk1
... output truncated ...

$ curl -k -L -X POST -H X-Auth-Token:$token -H 'Content-Type: application/json'
https://itso.lab:7443/rest/v1/lsmdisk/MDisk1 --data-raw '{"bytes": true}'
[{"... output truncated ..."}]
```

4.1.3 Using REST API with PowerShell

A PowerShell script will inherit the security settings for TLS/SSL from the operating system if they are not explicitly set in the script. This could cause communication failures if the TLS/SSL setting does not match the configuration of **lssecurity** (sslprotocol) on the Storage Virtualize system.

By default, a Storage Virtualize system uses self-signed SSL certificates.

During deployment you can choose to use the same certificates or replace them with CA signed certificates. Additional consideration is required in the PowerShell script when working with self-signed certificates.

The examples in this chapter are very basic because the intention is mainly to demonstrate the command syntax for different use cases with PowerShell.

For an actual deployment of a script, use recommended practices. For example, use PowerShell credentials, secure strings or hash files on disk for credentials, instead of having them in plain text in the script.

It is possible to access the REST API using two PowerShell cmdlets.

- ▶ Invoke-WebRequest
- ▶ Invoke-RestMethod

The examples within this chapter will focus on the `Invoke-RestMethod`.

Table 4-1 shows a syntax comparison of Curl, PowerShell, and PowerShell Core

Table 4-1 Syntax comparison

Description	curl	Windows PowerShell Invoke-RestMethod	PowerShell Core Invoke-RestMethod
SSL/TLS version	--tlsX.Y	Separate command	-SslProtocol
SSL verification	-k	Custom scripting needed	-SkipCertificateCheck
HTTP method	X	-Method	-Method
Headers	-H	-Headers	-Headers
Content Type Header	-H	-ContentType	-ContentType
Data	-d	-Body	-Body
URL/URI	argument	-Uri	-Uri

There are some differences in the PowerShell language between Windows PowerShell and PowerShell (Core). The differences are most notable in the availability and behavior of PowerShell cmdlets between Windows and non-Windows platforms and the changes resulting from the differences between the .NET Framework and the .NET Core.

PowerShell on Linux and macOS uses .NET Core, which is a subset of the full .NET Framework on Microsoft Windows. This is important because PowerShell provides direct access to the underlying framework types and methods.

This difference becomes visible when using the `Invoke-RestMethod` cmdlet to determine a specific SSL/TLS version or SSL verification behavior.

While the `Invoke-RestMethod` cmdlet from PowerShell Core provides dedicated options for this, using the cmdlet in conjunction with Windows PowerShell requires a separate command or additional scripting.

Authentication

The script that is shown in Example 4-5 shows an example of the authentication and creation of an access token using the Windows PowerShell cmdlet `Invoke-RestMethod`.

Example 4-5 Using the REST API by using Windows PowerShell

```
# Storage management IP or FQDN and according credentials
$myStorage = 'myStorage'
$myUser = 'myUser'
```

```

$myPassword = 'myUserPassword'

# Force TLS V1.2
[System.Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

#Code to handle self signed certificates
if (-not
([System.Management.Automation.PSTypeName]'ServerCertificateValidationCallback').Type)
{
$certCallback = @"
    using System;
    using System.Net;
    using System.Net.Security;
    using System.Security.Cryptography.X509Certificates;
    public class ServerCertificateValidationCallback
    {
        public static void Ignore()
        {
            if(ServicePointManager.ServerCertificateValidationCallback ==null)
            {
                ServicePointManager.ServerCertificateValidationCallback +=
                    delegate
                    (
                        Object obj,
                        X509Certificate certificate,
                        X509Chain chain,
                        SslPolicyErrors errors
                    )
                    {
                        return true;
                    };
            }
        }
    }
"@
    Add-Type $certCallback
}
[ServerCertificateValidationCallback]::Ignore()
# Get auth token
$token = Invoke-RestMethod -Method 'Post' -Headers @{'X-Auth-Username' = $($myUser);
'X-Auth-Password' = $($myPassword)} -ContentType 'application/json' -Uri
https://$($myStorage):7443/rest/v1/auth

```

The script that is shown in Example 4-6 shows an example of the authentication and creation of an access token using the PowerShell Core cmdlet `Invoke-RestMethod`.

As the cmdlet of PowerShell Core does provide the parameters `SslProtocol` and `SkipCertificateCheck`, there is no need for additional code within the script to workaroud that circumstances. Example assumes that credential variables are already set as in previous example.

Example 4-6 Using the REST API to get a token by using PowerShell Core

```

# Get auth token
$token = Invoke-RestMethod -SslProtocol Tls12 -SkipCertificateCheck-Method 'Post' -Headers
@{'X-Auth-Username' = $($myUser); 'X-Auth-Password' = $($myPassword)} -ContentType
'application/json' -Uri https://$($myStorage):7443/rest/auth

```

Submitting REST API requests

The script that is shown in Example 4-7 shows an example an API query to create a vdisk using the Windows PowerShell cmdlet `Invoke-RestMethod`. Script assumes that token is already received and stored in `$token` variable.

Example 4-7 Using the REST API by using Windows PowerShell - providing data as body

```
#Define a hash table with the parameters that you would like to pass to the REST API call
in "-body"
$body = @{
    "mdiskgrp" = 1
    "unit" = "gb"
    "size" = 100
    "name" = "my_vDisk_01"
}
#It is required to convert the content defined within the body to JSON format
$body_json = $body | ConvertTo-Json

# Create vdisk
Invoke-RestMethod -Method 'Post' -Headers @{ 'X-Auth-Token' = $Token.token } -ContentType
'application/json' -Body $body_json -Uri https://$(myStorage):7443/rest/v1/mkvdisk
```

The script that is shown in Example 4-8 shows an API request to list pool properties. This example demonstrates how to pass a Boolean value to `-Body` from a dictionary, for example a parameter like, `-bytes` does not have an argument so passing of a boolean value is required.

Example 4-8 Using the REST API by using Windows PowerShell - passing boolean values

```
#Define a hash table with the parameters for passing to the REST API call in "-body"
$body = @{
    "bytes" = [bool]::Parse('true')
}
#It is required to convert the content defined within the body to JSON format
$body_json = $body | ConvertTo-Json

# Get mdiskgroups
Invoke-RestMethod -Method 'Post' -Headers @{ 'X-Auth-Token' = $Token.token } -ContentType
'application/json' -Body $body_json -Uri https://$(myStorage):7443/rest/v1/lsmdiskgrp
```

With PowerShell core, the script will be almost the same, the only difference is that as it was explained before you can specify SSL version and skip certificate check. Example is shown in Example 4-9 on page 39. It assumes that token is already created and saved as a variable.

Example 4-9 Getting MDisk list with PowerShell Core

```
# Get mdisks
Invoke-RestMethod -SslProtocol Tls12 -SkipCertificateCheck -Method 'Post' -Headers
@{ 'X-Auth-Token' = $Token.token } -ContentType 'application/json' -Uri
https://$(myStorage):7443/rest/v1/lsmdisk
```

4.1.4 Using REST API in Python and Perl scripts

Python

The script that is shown in Example 4-10 shows an authentication and creation of an access token for further use in the context of querying all available MDisks with a python script.

The output of the script provides the following information about each MDisk:

- ▶ Name
- ▶ Name of the providing controller
- ▶ Name of the MDisk group
- ▶ Capacity
- ▶ Status

The script uses the `getpass` module to prompt for the password and prevent the storage of the credentials in clear text within the script.

Example 4-10 Using the REST API by using Python

```
#!/usr/bin/python

import json
import requests
import getpass

myStorage = 'myStorage'
myUser = 'myUser'
myPassword = getpass.getpass()

### disable SSL verification
ssl_verify = False

### ignore warning for SSL not being used
from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

### get session token
tokenRequest = requests.post('https://' + myStorage + ':7443/rest/v1/auth',
                             headers={
                                 'Content-type':      'application/json',
                                 'X-Auth-Username':   myUser,
                                 'X-Auth-Password':   myPassword
                             },
                             params="", data="", verify=ssl_verify)

### convert to JSON
_token = json.loads(tokenRequest.text)
token = _token['token']

### get mdisks
mdiskRequest = requests.post('https://' + myStorage + ':7443/rest/v1/lsmdisk',
                              headers={
                                  'Content-type':      'application/json',
                                  'X-Auth-token':     token
                              },
                              params="", data="", verify=ssl_verify)

_mdisk = json.loads(mdiskRequest.text)

print( '{:32.32s} {:20.20s} {:32.32s} {:8.8s} {:10.10s}' \
        .format("name","controller_name","mdisk_grp_name","capacity","status") )

for mdisk in _mdisk:
    print( '{:32.32s} {:20.20s} {:32.32s} {:8.8s} {:10.10s}' \
           .format(mdisk['name'],mdisk['controller_name'],mdisk['mdisk_grp_name'],mdisk['capacity'],mdisk['status']) )
```

The use of the `verify=False` option allows insecure SSL connections. By default, every SSL connection is verified to be secure. This option allows the request to get proceed; otherwise, the connection is considered insecure. If you use a signed SSL certificate, you do *not* need this option.

Perl

The script that is shown in Example 4-11 shows an example of the authentication and creation of an access token for further use in the context of querying all available MDisks.

The output of the script provides the following information about each MDisk:

- ▶ Name
- ▶ Name of the providing controller
- ▶ Name of the MDisk group
- ▶ Capacity
- ▶ Status

The script uses the `IO::Prompter` module to prompt for the password and prevent the storage of the credentials in clear text within the script.

Example 4-11 Using the REST API by using Perl

```
#!/usr/bin/perl

use strict;
use JSON;
use REST::Client;
use IO::Prompter;

my $myStorage = 'myStorage';
my $myUser = 'myUser';

my $myPassword = prompt 'Please enter your password:', -echo=>"";

my $restURL = 'https://' . $myStorage . ':7443/rest/v1/';

### get the session token
my $tokenRequest = REST::Client->new();
$tokenRequest->addHeader('Content-type', 'application/json');
$tokenRequest->addHeader('X-Auth-Username' , $myUser);
$tokenRequest->addHeader('X-Auth-Password', $myPassword);
$tokenRequest->getUseragent()->ssl_opts('verify_hostname' => 0);
$tokenRequest->POST($restURL . '/auth');
my $token = decode_json($tokenRequest->responseContent()->{'token'});

### get the list of mdisks
my $mdiskRequest = REST::Client->new();
$mdiskRequest->addHeader('Content-type', 'application/json');
$mdiskRequest->addHeader('X-Auth-Token', $token);
$mdiskRequest->getUseragent()->ssl_opts('verify_hostname' => 0);
$mdiskRequest->POST($restURL . '/lsmdisk');

my $mdiskList = $mdiskRequest->responseContent();
my @mdiskListJSON = @{decode_json($mdiskList)};

for my $key (@mdiskListJSON) {
    printf "%32s %20s %32s %8s %10s\n",
        $key->{'name'},
        $key->{'controller_name'},
        $key->{'mdisk_grp_name'},
```

```
$key->{'capacity'},  
$key->{'status'};  
}
```

The use of the `getUseragent()->ssl_opts('verify_hostname' => 0)` method allows insecure SSL connections. By default, every SSL connection is verified to be secure. This option allows the request to proceed; otherwise, the connection is considered insecure. If you use a signed SSL certificate, you do *not* need this option.



5

Automation with Red Hat Ansible

This chapter provides information about scripting and automation tasks that can occur in an IBM Storage Virtualize environment that uses Ansible.

This chapter includes the following topics:

- ▶ “Introduction to Ansible” on page 44
- ▶ “Automation with Red Hat Ansible” on page 45

5.1 Introduction to Ansible

Ansible is a widely used open-source tool that revolutionizes the way infrastructure and applications are managed. As a powerful Infrastructure as Code (IaC) tool, Ansible allows IT professionals to automate the deployment, configuration, and management of systems and applications.

5.1.1 Key features of Ansible

The key features of Ansible are as follows.

Agentless architecture

Unlike many other automation tools, Ansible operates without requiring agents to be installed on remote systems. It uses SSH for Linux/Unix systems and WinRM for Windows systems to communicate and execute tasks.

Simple yet powerful

Ansible's configuration files, known as playbooks, are written in YAML, which is both human-readable and easy to write. This simplicity doesn't compromise its power and flexibility, making it suitable for complex automation tasks.

Idempotency

Ansible ensures that applying the same configuration multiple times will result in the same state, preventing unintended changes and ensuring consistency across deployments.

Extensive ecosystem

Ansible has a vast collection of modules that cover a wide range of tasks, from provisioning and configuration management to application deployment and orchestration. The Ansible Galaxy is a community-driven repository where users can share and access pre-written roles and playbooks.

Cross-platform support

Ansible supports a wide range of operating systems and cloud providers, making it a versatile tool for managing diverse IT environments, including on-premises servers, cloud instances, and containerized applications.

Scalability

Designed to handle both small and large-scale environments, Ansible can manage everything from a few servers to thousands of nodes, scaling seamlessly as your infrastructure grows.

Integration with DevOps practices

Ansible integrates well with other DevOps tools and practices, facilitating continuous integration, continuous delivery (CI/CD), and infrastructure as code (IaC). This integration streamlines the deployment pipeline and enhances collaboration between development and operations teams.

5.1.2 Common use cases for Ansible

The following are common use cases for Ansible.

Provisioning

Automatically setting up new servers and cloud instances with the necessary software and configurations.

Configuration management

Ensuring that systems remain in the desired state by managing configuration files, packages, and services.

Application deployment

Automating the deployment of applications, ensuring consistency across different environments (development, testing, production).

Orchestration

Coordinating complex workflows and interdependencies between multiple systems and services.

Security and compliance

Implementing security policies and ensuring compliance with industry standards by automating audits and enforcement.

5.2 Automation with Red Hat Ansible

Automation is a priority for maintaining today's busy storage environments. Automation software allows for the creation of repeatable sets of instructions. It also reduces the need for human interaction with computer systems.

Red Hat Ansible and other third-party automation tools are becoming increasingly used across the enterprise IT environments. It is not unexpected that their use in storage environments is to become more popular.

5.2.1 Red Hat Ansible

The IBM Storage Virtualize family includes integration with Red Hat Ansible automation platform. This integration allows IT to create an Ansible playbook that automates repetitive tasks across an organization in a consistent way, which helps improve outcomes and reduces errors.

Ansible is an agentless automation management tool that uses primarily the SSH protocol.

REST API support: Ansible itself does not have a built-in REST API, but you can achieve interaction with external REST APIs over HTTPS within Ansible playbooks using the following methods:

- ▶ **URI (Uniform Resource Identifier) module:** Ansible provides the uri module which allows you to make HTTP/HTTPS requests to external APIs. This module can be used within playbooks to interact with various REST APIs secured with HTTPS. The uri module allows you to specify the URL, method (GET, POST, PUT, etc.), headers, and body for the HTTP/HTTPS request.
- ▶ **Community modules:** Community modules offer additional functionality specific to the API they are designed for, such as authentication handling or automatic data parsing.

Some of the modules in SpecV Ansible collection use REST API over HTTPS.

Supported platforms for Ansible include Red Hat, SUSE, Debian, CentOS, macOS, and any of the Berkeley Software Distribution (BSD) versions.

Windows support: The recommended and officially supported platform for the Ansible control node is a Linux distribution. Windows Subsystem for Linux (WSL) allows you to run a Linux environment within Windows 10 or later versions. While it can be used for some basic Ansible tasks, it might not be ideal for production environments due to potential performance limitations or compatibility issues.

5.2.2 Red Hat Ansible editions

The following Red Hat Ansible editions are available:

- ▶ Ansible Core

Ansible Core is the command line tool that is installed from community repositories or the official Red Hat repositories for Ansible.

- ▶ Ansible Tower

Ansible Tower is the GUI tool that is used to run Ansible tasks. Tower requires a license that is based on the number of systems Ansible Tower is to manage. Ansible Tower is available as Standard or Premium Edition (24x7 support is included in the Premium Edition).

5.2.3 Requirements

Ansible server (Control Node) features the following requirements:

- ▶ Python 3 (versions 3.9 and higher).

Note: Certain control node plugins may have additional requirements. Refer to the plugin documentation for details.

- ▶ Host requirements:
 - Although you do not need a daemon on your managed nodes, you need a way for Ansible to communicate with them.
 - For most managed nodes, Ansible makes a connection over SSH and transfers modules by using SFTP. If SSH works but SFTP is not available on some of your managed nodes, you can switch to SCP in `ansible.cfg`.
 - For any machine or device that can run Python, you also need Python 2 (version 2.7 or later) or Python 3 (version 3.5 or later).
 - In addition for non-Python machines, PowerShell 3 - 5.1 can be used.

Note: Some modules feature more requirements that must be met on the *target* machine (the managed node). These requirements are listed in [ibm.spectrum_virtualize](#).

5.2.4 Essential terminology in an Ansible environment

Ansible environment features the following essential terminology:

- ▶ Ansible Galaxy: A hub for finding and sharing Ansible content.
- ▶ Ansible server: The machine with Ansible installed, which runs all tasks and playbooks.
- ▶ Playbook: A framework where Ansible automation tasks are defined (written in YAML).
- ▶ Task: A section that contains a single procedure that you want to be run.
- ▶ Tag: A name that you can assign to a task.
- ▶ Play: The execution of a playbook.
- ▶ Hosts: The devices that you manage with Ansible.
- ▶ Modules: A command or set of commands that are made for execution on the client side.
- ▶ Handler: A task that is called only if a notifier is present.
- ▶ Notifier: A section that is assigned to a task that calls a handler if the output is changed.
- ▶ Inventory: A file that contains Ansible client/server data.
- ▶ Fact: Information that is retrieved from the client from global variables by using the `gather-facts` operation.
- ▶ Roles: A structured way of grouping tasks, handlers, variables, and other properties.
- ▶ Container: Ansible Container uses Ansible roles to build images, initialize projects, and add services to projects.

5.2.5 Automating IBM Storage with Ansible

IBM data storage provides simple storage solutions that address modern data requirements and provides a solution to your hybrid multicloud strategy.

With the speed, scale, and complexity of hybrid multicloud and even traditional on-premises environments, automation became a priority.

IBM Storage FlashSystem family for hybrid multicloud includes integration with Red Hat Ansible Automation Platform. It allows IT to create an Ansible playbook that automates the tasks that are repeated across an organization in a consistent way, which helps improve outcomes and reduces risk.

It also standardizes how IT and application owners interact together and features the following benefits:

- ▶ With Red Hat Ansible Automation Platform and IBM Storage, customers can easily automate tasks, such as configuration management, provisioning, workflow orchestration, application deployment, and lifecycle management.
- ▶ By using Red Hat Ansible Automation Platform and IBM Storage, customers can reduce system inconsistencies with the automation modules.
- ▶ Red Hat Ansible Automation Platform can also be used to configure end-to-end infrastructure in an orchestrated fashion.
- ▶ Ansible provides a single pane of glass visibility to multicluster, multicloud environments, which allows lines of business to use playbooks to accomplish their goals without needing to understand the details of how the work is being done.

IBM is a Red Hat-certified support module vendor that provides simple management for the following commands that are used in the IBM Storage Virtualize Ansible Collection (This is not a complete list. For most up-to-date information refer to this [link](#))

- ▶ **Collect facts:** Collect basic information, including hosts, host groups, snapshots, consistency groups, and volumes.
- ▶ **Manage hosts:** Create, delete, or modify hosts.
- ▶ **Manage volumes:** Create, delete, or extend the capacity of volumes.
- ▶ **Manage MDisk:** Create or delete a managed disk.
- ▶ **Manage pool:** Create or delete a pool (managed disk group).
- ▶ **Manage volume map:** Create or delete a volume map.
- ▶ **Manage consistency group snapshot:** Create or delete consistency group snapshots.
- ▶ **Manage snapshot:** Create or delete snapshots.
- ▶ **Manage volume clones:** Create or delete volume clones.

This collection provides a series of Ansible modules and plug-ins for interacting with the IBM Storage Virtualize family storage products. The modules in the IBM Storage Virtualize Ansible collection use the REST API to connect to the IBM Storage Virtualize storage system. These products include:

- ▶ IBM SAN Volume Controller
- ▶ IBM Storage FlashSystem family members that are built with IBM Storage Virtualize
- ▶ IBM Storwize® family
- ▶ IBM Storage Virtualize for Public Cloud

For more information, see *Automate and Orchestrate® Your IBM FlashSystem Hybrid Cloud with Red Hat Ansible*, REDP-5598.

For IBM Storage Virtualize modules, Ansible version 2.14 or higher is required. For more information about IBM Storage Virtualize modules, see the Ansible web page [ibm.spectrum_virtualize](#). Also, see [IBM Documentation - Using Ansible](#).

5.2.6 Getting started

The Ansible Collection (`ibm.storage_virtualize`) provides a series of Ansible modules and plug-ins for interacting with the IBM Storage Virtualize family storage products.

As of this writing, the Ansible collection for IBM Storage Virtualize is available in version 2.4.

All information in this section is based on this version.

Prerequisites for using the modules

Paramiko must be installed to use `ibm_svctask_command` and `ibm_svcinfo_command` modules.

Paramiko is a Python (2.7, 3.4+) implementation of the SSHv2 protocol, and provides client and server functions.

Although Paramiko is a Python C extension for low-level cryptography, it is a pure Python interface around SSH networking concepts.

Current limitations

The modules in the IBM Storage Virtualize Ansible collection use the REST API to connect to the IBM Storage Virtualize storage system.

This collection includes the following limitations:

- ▶ The use of the REST API to list more than 2000 objects might create a loss of service from the API side because it automatically restarts because of memory constraints.
- ▶ The Ansible collection can run on all supported IBM Storage Virtualize storage versions.
- ▶ It is not possible to access the REST API by using a cluster IPv6 address.
- ▶ With the release of Storage Virtualize Ansible v1.8.0, automation of license agreement acceptance, including the EULA, is only available for Licensed Machine Code (LMC) systems. Non-LMC systems require users to accept the license agreement through a GUI setup wizard during login, regardless of whether Ansible modules were used for initial configuration.

What is LMC and non-LMC code?

- ▶ **Licensed Machine Code (LMC):** This is the code provided by the storage system vendor (like IBM in the case of Storage Virtualize) and is subject to a specific license agreement, often an End-User License Agreement (EULA). LMC typically includes critical firmware, microcode, and other essential code that governs the core operations of the storage system. The vendor retains ownership of the LMC, and its use is governed by the terms of the license agreement. This may include restrictions on copying, modifying, or reverse engineering the code.
- ▶ **Non-LMC code:** This category encompasses code that does not fall under the vendor's specific licensing terms. It can include:
 - **Open-source code:** Some storage systems may utilize open-source components that are not subject to vendor licensing restrictions.
 - **User-developed code:** Users might have the ability to write custom scripts or programs for the storage system using supported APIs. These user-developed codes wouldn't be considered LMC.

Prerequisites

Ensure that the following prerequisites are met:

- ▶ Ansible is installed and configured on a controller node.
- ▶ Ansible Galaxy Collection `ibm.storage_virtualize` is installed on the same controller node.
- ▶ Network access is available from the controller node to IBM Storage Virtualize Management IP.
- ▶ A user with sufficient authorization to create or delete objects on IBM Storage Virtualize.
- ▶ IBM Storage Virtualize operates at version 8.1.3 or higher above 8.1.3, with the exceptions of versions 8.3.1.3, 8.3.1.4, and 8.3.1.5.

Installing or upgrading Ansible Galaxy Collection `ibm.storage_virtualize`

To install the IBM Storage Virtualize collection that is hosted in Galaxy, use the following command:

```
ansible-galaxy collection install ibm.storage_virtualize
```

To upgrade to the latest version of the IBM Storage Virtualize collection, use the following command:

```
ansible-galaxy collection install ibm.storage_virtualize --force
```

Functions provided by IBM Storage Virtualize Ansible modules

Table 5-1 list the modules provided by the `ibm.storage_virtualize` collection version 2.0.0.

Table 5-1 *Provided modules with `ibm.spectrum_virtualize` collection version 2.0.0*

Name of module	Description
<code>ibm_svc_auth</code>	Generates an authentication token for a user on the IBM Storage Virtualize family storage system.
<code>ibm_svc_complete_initial_setup</code>	Completes the initial setup configuration for Licensed Machine Code (LMC) systems. For non-LMC systems, logging in to the user interface is required to complete the automation of Day 0 configuration.
<code>ibm_svc_host</code>	Manages hosts that are on the IBM Storage Virtualize system.
<code>ibm_svc_hostcluster</code>	Manages the host cluster that is on the IBM Storage Virtualize system.
<code>ibm_svc_info</code>	Collects information about the IBM Storage Virtualize system.
<code>ibm_svc_initial_setup</code>	Manages initial setup configuration on the IBM Storage Virtualize system.
<code>ibm_svc_manage_callhome</code>	Manages configuration of the Call Home feature on the Storage Virtualize system.
<code>ibm_svc_manage_consistgrp_flashcopy</code>	Manages the FlashCopy consistency groups that are on the IBM Storage Virtualize system.

Name of module	Description
ibm_svc_manage_cv	Manages the change volume in remote copy replication that is on the IBM Storage Virtualize system.
ibm_svc_manage_flashcopy	Manages the FlashCopy mappings that are on the IBM Storage Virtualize system.
ibm_svc_manage_ip	Manages IP provisioning on the IBM Storage Virtualize system.
ibm_svc_manage_migration	Manages volume migration between clusters on IBM Storage Virtualize systems.
ibm_svc_manage_mirrored_volume	Manages the mirrored volumes that are on the IBM Storage Virtualize system.
ibm_svc_manage_ownershipgroup	Manages ownership groups on the IBM Storage Virtualize system.
ibm_svc_manage_portset	Manages IP portset on IBM Storage Virtualize system.
ibm_svc_manage_replication	Manages the remote copy replication that is on the IBM Storage Virtualize system.
ibm_svc_manage_replicationgroup	Manages the remote copy consistency group on the IBM Storage Virtualize system.
ibm_svc_manage_safeguarded_policy	Manages safeguarded policy configuration on the IBM Storage Virtualize system.
ibm_svc_manage_sra	Manages the remote support assistance configuration on the IBM Storage Virtualize system.
ibm_svc_manage_storage_partition	Manages storage partitions on the IBM Storage Virtualize system.
ibm_svc_manage_syslog_server	Manages syslog server on the IBM Storage Virtualize system.
ibm_svc_manage_user	Manages users on the IBM Storage Virtualize system.
ibm_svc_manage_usergroup	Manages user groups on the IBM Storage Virtualize system.
ibm_svc_manage_volume	Manages the standard volumes on the IBM Storage Virtualize system.
ibm_svc_manage_volumegroup	Manages the volume groups that are on the IBM Storage Virtualize system.
ibm_svc_mdisk	Manages the MDisks for IBM Storage Virtualize system.
ibm_svc_mdiskgrp	Manages pools for IBM Storage Virtualize system.
ibm_svc_start_stop_flashcopy	Starts or stops the FlashCopy mapping and consistency groups that are on the IBM Storage Virtualize system.

Name of module	Description
ibm_svc_start_stop_replication	Starts or stops the remote copy relationship or group on the IBM Storage Virtualize system.
ibm_svc_vol_map	Manages the volume mapping for IBM Storage Virtualize system.
ibm_svcinfo_command	Runs the svcinfo CLI command on the IBM Storage Virtualize system over an SSH session.
ibm_svctask_command	Runs the svctask CLI commands on the IBM Storage Virtualize system over and SSH session.
ibm_sv_manage_awss3_cloudaccount	Manages Amazon S3 cloud account configuration on IBM Storage Virtualize system.
ibm_sv_manage_cloud_backups	Manages cloud backup on the IBM Storage Virtualize system.
ibm_sv_manage_drive	This module manages drives on IBM Storage Virtualize family storage systems.
ibm_sv_manage_fc_partnership	Manages Fibre Channel (FC) partnership on the IBM Storage Virtualize system.
ibm_sv_manage_fcportsetmember	Manages addition or removal of ports from the Fibre Channel (FC) portsets on the IBM Storage Virtualize system.
ibm_sv_manage_ip_partnership	Manages IP partnership configuration on the IBM Storage Virtualize system.
ibm_sv_manage_provision_policy	Manages provisioning policies on the IBM Storage Virtualize system.
ibm_sv_manage_replication_policy	Manages policy-based replication configuration on the IBM Storage Virtualize system.
ibm_sv_manage_security	Manages security options on IBM Storage Virtualize family systems.
ibm_sv_manage_snapshot	Manages snapshots (mutual consistent images of a volume) on the IBM Storage Virtualize system.
ibm_sv_manage_snapshotpolicy	Manages snapshot policy configuration on the IBM Storage Virtualize system.
ibm_sv_manage_ssl_certificate	Exports an existing system certificate on to IBM Storage Virtualize system.
ibm_sv_manage_truststore_for_replication	Manages certificate trust stores for replication on the IBM Storage Virtualize system.
ibm_sv_restore_cloud_backup	Restores cloud backups on IBM Storage Virtualize system.
ibm_sv_switch_replication_direction	Switches the replication direction on the IBM Storage Virtualize system.

Note: Beginning with version 1.6.0, the `ibm_svc_vdisk` module is considered a deprecated feature. A new module (`ibm_svc_manage_volume`) was introduced to manage standard volumes.

Getting help for IBM Storage Virtualize Ansible modules

To get the online documentation for a specific module that is displayed, use the following command:

```
ansible-doc <collection-name>.<module-name>
```

The output of the help includes all permissible options and some examples of how to use the module (see Example 5-1).

Example 5-1 Example displaying online help

```
ansible-doc ibm.storage_virtualize.ibm_svc_manage_volume
```

```
> IBM_SVC_MANAGE_VOLUME
```

```
Ansible interface to manage 'mkvolume', 'rmvolume', and 'chvdisk' volume commands.
```

```
* This module is maintained by The Ansible Community
```

```
OPTIONS (= is mandatory):
```

```
- buffersize
```

```
    Specifies the pool capacity that the volume will reserve as a buffer for thin-provisioned and compressed volumes.
```

```
    Parameter 'thin' or 'compressed' must be specified to use this parameter. The default buffer size is 2%.
```

```
    `thin' or `compressed' is required when using `buffersize'.
```

```
    Valid when `state=present' to create a volume.
```

```
    [Default: (null)]
```

```
    type: str
```

```
= clustername
```

```
    The hostname or management IP of the Storage Virtualize storage system.
```

```
    type: str
```

5.2.7 Securing credentials in Ansible

While working with Ansible, you can create several playbooks, inventory files, variable files, and so on. Some of the files might contain sensitive data, such as access credentials. To protect this kind of data, Ansible provides the Ansible Vault, which helps to prevent this data from being exposed. Sensitive data and passwords are kept in an encrypted file rather than in plain text files.

5.2.8 Creating an Ansible playbook

The playbook consistently automates the tasks that are repeated across an organization, which improves outcomes and reduces risk. It also standardizes how IT and application owners interact.

In this section, we discuss creating an Ansible playbook. The creation of the playbook is based on the use case that is used here.

For a new VMware ESX cluster that consists of two new servers, two vdisks are to be created and mapped to the host cluster object.

Note: The idempotency property might be included in a mathematics or computer science operation. It roughly means that an operation can be carried out multiple times without changing the result.

The IBM Storage Virtualize Ansible modules provide idempotency in Ansible playbooks.

The IBM Storage Virtualize Ansible modules check whether the object to be created exists in the defined state and does not attempt to create it again.

Table 5-2 lists the variable parameters and their values for the example playbook.

Table 5-2 Variable parameters and their values for the example playbook

Attribute	Value
Name of new host cluster	ESX-Cluster-1
Name of new host 1	ESX-Host-1
WWPNs of new host 1	100000109C400798, 1000001AB0440446
Name of new host 2	ESX-Host-2
WWPNs of new host 2	100000109B600424, 1000001BC0660146
Name of vdisk 1	Datastore1
Name of vdisk 2	Datastore2

The steps that are used to create an Ansible playbook are described next.

Step 1: Authentication

Example 5-2 shows the required YAML notation for the part of the playbook to authenticate at the IBM Storage Virtualize REST API to obtain a token for further use. To avoid storing the password in clear text within the playbook, the password was encrypted in a vault.

The generation of a token is an optional task only required for optimization purposes, as all modules accept username and password to authenticate by itself.

Example 5-2 YAML notation for obtaining an authentication token

```
vars:
  clustername: <Cluster management ip | hostname>
  domain: <FQDN>
  username: myuser
  password: !vault |
    $ANSIBLE_VAULT;1.1;AES256
62653531313434393266646438306537396264306433653638343439643136333238383139616561
6530373430636265316639626234376336306630343333640a326332626564656233323336333239
39633132656631353030386430663736363631656438343364346235653534316333333233333531
3166343263626538360a633664616264326133643339336333363638323232373962393839356637
6138
tasks:
```

```

- name: Obtain an authentication token
  register: result
  ibm_svc_auth:
    clustertype: "{{ clustertype }}"
    domain: "{{ domain }}"
    username: "{{ username }}"
    password: "{{ password }}"

```

Note: It is valid to use the parameter <domain> when a hostname is used for the parameter <clustertype>.

For more information about how to work with Ansible vaults, see [Protecting sensitive data with Ansible vault](#).

Step 2: Creating the host cluster object

Example 5-3 shows the required YAML notation for the part of the playbook to create an empty host cluster object.

Example 5-3 YAML notation for creating an empty host cluster

```

- name: Define a new host cluster
  ibm_svc_hostcluster:
    clustertype: "{{ clustertype }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: <hostcluster_name>
    state: present

```

Step 3: Creating an FC host

Example 5-4 on page 55 shows the required YAML notation for the part of the playbook to create an FC host object.

Example 5-4 YAML notation for creating a new FC host object

```

- name: Define a new FC host
  ibm_svc_host:
    clustertype: "{{ clustertype }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: "{{ hostname }}"
    state: present
    fcwwpn: "{{ fcwwpn(s) }}"
    iogrp: 0:1:2:3
    protocol: scsi
    type: generic
    hostcluster: "{{ hostcluster_name }}"

```

Step 4: Creating a thin-provisioned volume

Example 5-5 shows the required YAML notation for the part of the playbook to create a thin-provisioned volume.

Example 5-5 YAML notation to create a thin-provisioned volume

```
- name: Create a thin-provisioned volume
  ibm_svc_manage_volume:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: "volume_name"
    state: "present"
    pool: "<pool_name>"
    size: "<size>"
    unit: "<size_unit>"
    thin: true
    buffersize: 10%
```

Step 5: Mapping the new volume to the host cluster object

Example 5-6 shows the required YAML notation for the part of the playbook to map the new volume to the hostcluster.

Example 5-6 YAML notation to map a volume to the hostcluster

```
- name: Map a volume to a hostcluster
  ibm_svc_vol_map:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    volname: <volume_name>
    hostcluster: <hostcluster_name>
    state: present
```

If a SCSI-ID must be specified, use the `scsi: <scsi-id>` parameter.

Putting it all together

Example 5-7 shows the combined required tasks for the use of the IBM Storage Virtualize collection to create hostcluster volumes (this use case is featured in this chapter) to be used as a playbook by Ansible. All customized lines of the playbook are highlighted in bold in the example.

Example 5-7 Complete playbook for specified use-case

```
- name: Using Storage Virtualize collection to create hostcluster - hosts -
  volumes
  hosts: localhost
  collections:
    - ibm.storage_virtualize
  gather_facts: no
  connection: local

# definition of global variables
vars:
  clustername: mySVC
```

```

    domain: mydomain.com
    username: myuser
    password: !vault |
               $ANSIBLE_VAULT;1.1;AES256
62653531313434393266646438306537396264306433653638343439643136333238383139616561
6530373430636265316639626234376336306630343333640a326332626564656233323336333239
39633132656631353030386430663736363631656438343364346235653534316333333233333531
3166343263626538360a633664616264326133643339336333363638323232373962393839356637
6138
    log_path: /tmp/redbook-example.log

# define variables for running the playbook
# hostcluster
    hostcluster_name: ESX-Cluster-1

# host 1
    host1_name: ESX-Host-1
    host1_fcwvpn: 100000109C400798{{":"}}1000001AB0440446

# host 2
    host2_name: ESX-Host-2
    host2_fcwvpn: 100000109B600424{{":"}}1000001BC0660146

# pools to use for volume mirror
    pool1_name: pool1
    pool2_name: pool2

# volume 1
    volume1_name: Datastore1
    volume1_size: '10'
    volume1_size_unit: tb

# volume 2
    volume2_name: Datastore2
    volume2_size: '10'
    volume2_size_unit: tb

tasks:
# creating an authentication token for further usage within the playbook
- name: Obtain an authentication token
  register: result
  ibm_svc_auth:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    username: "{{ username }}"
    password: "{{ password }}"
    log_path: "{{ log_path }}"

# create the hostcluster object
- name: Create the hostcluster
  ibm_svc_hostcluster:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"

```

```

    name: "{{ hostcluster_name }}"
    state: present

# create first host object
- name: Define first FC host
  ibm_svc_host:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: "{{ host1_name }}"
    state: present
    fcwwpn: "{{ host1_fcwwpn }}"
    iogrp: 0:1:2:3
    protocol: scsi
    type: generic
    hostcluster: "{{ hostcluster_name }}"

# create second host object
- name: Define second FC host
  ibm_svc_host:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: "{{ host2_name }}"
    state: present
    fcwwpn: "{{ host2_fcwwpn }}"
    iogrp: 0:1:2:3
    protocol: scsi
    type: generic
    hostcluster: "{{ hostcluster_name }}"

# create first mirrored thin-provisioned volume
- name: Create first thin-provisioned volume
  ibm_svc_manage_volume:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    name: "{{ volume1_name }}"
    state: "present"
    pool: "{{ pool1_name }}:{{ pool2_name }}"
    size: "{{ volume1_size }}"
    unit: "{{ volume1_size_unit }}"
    thin: true
    buffersize: 10%

# create second mirrored thin-provisioned volume
- name: Create second thin-provisioned volume
  ibm_svc_manage_volume:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"

```

```

    name: "{{ volume2_name }}"
    state: "present"
    pool: "{{ pool1_name }}:{{ pool2_name }}"
    size: "{{ volume2_size }}"
    unit: "{{ volume2_size_unit }}"
    thin: true
    buffersize: 10%

# mapping of first volume to the hostcluster
- name: Map first volume to the hostcluster
  ibm_svc_vol_map:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    volname: "{{ volume1_name }}"
    hostcluster: "{{ hostcluster_name }}"
    state: present

# mapping of second volume to the hostcluster
- name: Map second volume to the hostcluster
  ibm_svc_vol_map:
    clustername: "{{ clustername }}"
    domain: "{{ domain }}"
    token: "{{ result.token }}"
    log_path: "{{ log_path }}"
    volname: "{{ volume2_name }}"
    hostcluster: "{{ hostcluster_name }}"
    state: present

```

5.2.9 More automation

The use case that is described in this chapter can be extended by completing the following steps:

1. Create the required FC zoning.
2. Scan the HBA for the newly created volumes.
3. Create a VMFS data store on the discovered volumes.
4. Create one or more virtual machines (VMs).

For more information about the Brocade FOS FC collection on Ansible Galaxy, see the Ansible web page [brocade.fos](#).

For more information about the `community.vmware` Ansible Collection on Ansible Galaxy, see this Ansible web page [community.vmware](#).

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Unleash the Power of Flash: Getting Started with IBM Storage Virtualize Version 8.7 on IBM Storage FlashSystem and IBM SAN Volume Controller, SG24-8561*
- ▶ *Automate and Orchestrate® Your IBM FlashSystem Hybrid Cloud with Red Hat Ansible, REDP-5598*

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Documentation - Using Ansible
<https://www.ibm.com/docs/en/flashsystem-9x00/8.7.x?topic=started-using-ansible/>
- ▶ IBM Documentation - Storage Virtualize RESTful API
<https://www.ibm.com/docs/en/flashsystem-9x00/8.7.x?topic=interface-storage-virtualize-restful-api>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



REDP-5736-00

ISBN

Printed in U.S.A.

Get connected

