Cisco UCS Automation for Software-Defined Storage

This document provides an overview of Cisco Unified Computing System™ (Cisco UCS®) automation for software-defined storage. This guide discusses the various aspects of automation for Cisco UCS and takes a close look at how Cisco UCS automation can simplify software-defined storage solutions.

Contents

Executive summary

Data center automation is a vital step toward achieving the business results organizations need to compete effectively. You can use automation to power IT processes across computing, network, and storage layers in both your physical and virtual environments. With Cisco® data center management and automation, you can gain the agility you need to act and achieve results more quickly. From infrastructure to applications, Cisco can support the right data center automation strategy for your organization.

The data center is a critical part of business strategy. The faster it produces results, the more competitive the business can be in a world that values speed. To provide that speed, the data center must be agile—it must be automated.

Software-defined storage plays an important role in automation. Storage environments are becoming bigger, and the difficulties of configuration and installation are increasing. And again, in software-defined storage, agility must be addressed. Businesses need to be more agile and more responsive to changes in storage requirements.

The definition of agility depends on the processes that need automating and the roles of the users:

- End users want to order applications in a self-service manner and take delivery within minutes.
- Application developers need automated delivery of standardized infrastructure resources to develop, test, and deploy applications.
- IT needs to deliver application workloads on demand in an automated and repeatable manner, eliminating manual provisioning and deprovisioning of resources.

Cisco provides automation modules that allow adoption at a pace that is comfortable for your organization. Automation starts with the infrastructure, encompasses the application, and extends to enterprise and hybrid cloud deployments. All modules work together and are operated using a common interface, which simplifies service lifecycle management.

Cisco automation includes software-defined storage. Cisco has the right interfaces and options to help you implement automation in the storage area.

Introduction

The growing volume and variety of data are putting pressure on data centers to keep pace. At the same time, fast-changing market conditions require enterprises to be agile and to innovate constantly. IT teams are no longer considered cost centers. Instead, they are treated as strategic business owners and are expected to suggest technology initiatives to help the business keep up with market demands. Storage infrastructure is the foundational component that helps IT teams meet these growing business demands.

Manual storage provisioning and management, which is prone to human errors, is no longer a feasible option. While storage technologies and infrastructure needs are becoming increasingly complex, operations budgets are shrinking. Therefore, storage automation is now a critical tool for enterprises.

The automation features of software-defined storage are empowering IT to deliver new storage instantly instead of requiring a long-time frame, helping ensure consistent configurations that keep service levels high and enabling the monitoring and management capabilities that keep storage reliable and trouble-free.

In short, automated software-defined storage enables IT-provisioned storage to compete with cloud-based alternatives. That means that users don't have to go outside the organization to get what they want. And that means that IT can maintain control over storage performance, security, and compliance while still providing the rapid response that users today demand.

Software-defined storage replaces traditional, purpose-built storage. It can be managed as a single platform and automates delivery of services based on built-in intelligence and best practices. The preconfigured, standardized storage components that are typical of today's software-defined storage solutions deliver many of the same task-specific capabilities as standalone systems. They just use a different approach to get there—a faster, easier, more cost-effective approach that relies on software-based specialization and automation to meet different requirements quickly and economically. This approach leads to more consistent, predictable storage solutions that don't require proprietary expertise to build or maintain them.

But many storage solutions don't address the diversity of the underlying hardware. They just define the various components that are part of the overall solution and assume that the IT engineer can build that overall solution in a professional manner. Almost all software-defined storage solutions use standardized server hardware with local storage. Configuring and building such a solution doesn't appear to be especially complex when considering the hardware for a single server, but the process can time consuming and complicated when a solution consists of tens or hundreds of such servers. The process is also error prone, and it does not provide a simple way to build a flexible and simplified storage solution.

Table 1 compares two common options available to achieve a faster, easier, and more cost-effective solution than with traditional storage solutions:

- **Prebuilt (appliance) solution:** You can purchase prebuilt hardware with a software-defined storage solution on top of the hardware. Such an appliance can offer a degree of choice and allow you to do what you want to do from a software automation standpoint: provisioning, taking snapshots, expanding your system, etc. However, you have to pay for the preconfiguration, and you may not be able to make changes in the underlying hardware. Flexibility is missing.
- **Open architecture:** With an open design, as long as the technical requirements are met, you can choose the hardware you want. Sometimes you can use only certified vendors for the specific software-defined storage solution, but in other cases the architecture is fully open. That provides a green field for the IT architect, with a lot of flexibility, but also requiring a lot of work to identify the right hardware for your solution. Nevertheless, an open solution allows automation. And you can automate the process of configuration and installation of your hardware as well as your software.

Table 1. Comparison of appliance versus open architecture for software-defined storage

	Appliance	Open architecture
Prebuilt solution	Yes	No
Software choice	No	Yes
Capital costs	Expensive	Less expensive
Flexibility	Somewhat flexible	Very flexible
Automation	Somewhat flexible	Very flexible

Table 1 doesn't fully answer the question of which approach is right for you. Depending on the specific use case, an appliance might make more sense than an open architecture. However, for the purposes of automation, an open architecture offers a wider range of possibilities than an appliance.

But few solutions can perform end-to-end automation. Two methods for automating server hardware are commonly used:

- **Redfish:** Redfish is an open industry-standard specification, API, and schema developed by the Distributed Management Task Force (DMTF) Scalable Platforms Management Forum (SPMF) in 2015. It specifies a representational state transfer (REST) interface and uses JavaScript Object Notation (JSON) and the Open Data Protocol (OData). Redfish is designed to deliver simple and secure management for converged, hybrid IT and the software-defined data center (SDDC). It is suitable

for a wide range of servers, from standalone servers to rack-mount and blade server environments. It scales equally well for large-scale cloud environments.

- **Intelligent Platform Management Interface (IPMI):** IPMI is a set of computer interface specifications for an autonomous computer subsystem that provides management and monitoring capabilities independent of the host system's CPU, firmware (BIOS or Unified Extensible Firmware Interface [UEFI]), and operating system. IPMI defines a set of interfaces used by system administrators for out-of-band management of computer systems and the monitoring of their operation. For example, by using a network connection to the hardware rather than to an operating system or login shell, IPMI provides a way to manage a computer that may be powered off or otherwise unresponsive. The specification is led by Intel and was first published in 1998. The interface is thus relatively outdated.

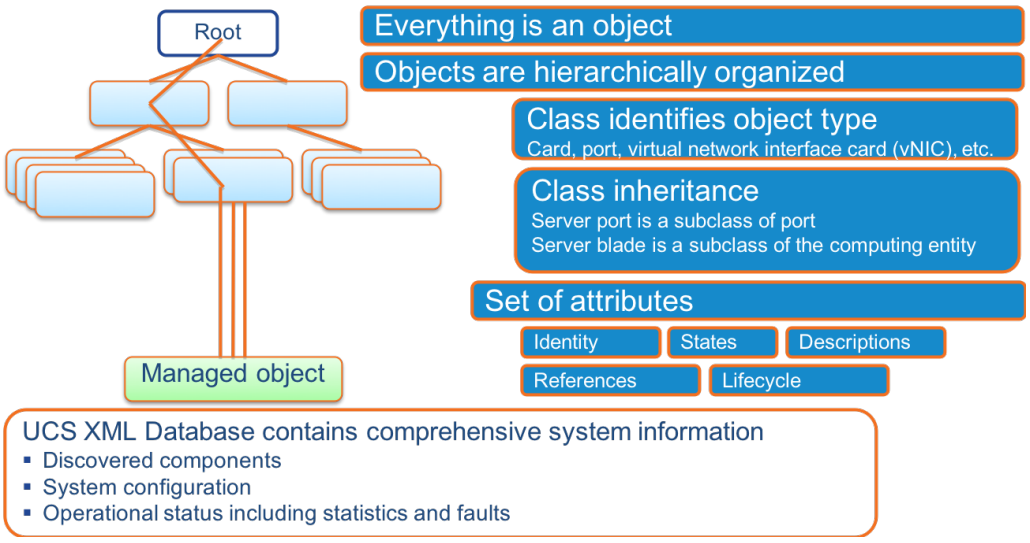
IPMI is an interface that each manufacturer can adapt to its hardware, whereas Redfish is intended to be a more general interface. However, neither can cover all the hardware components of a full-width server architecture. What is needed is a solution that allows you to control and manage the complete server hardware and install and configure your software-defined storage environment.

With the release of the Cisco Unified Computing System™ (Cisco UCS®), Cisco offers a radical change, providing a completely new interface that manages all parts of the Cisco UCS infrastructure.

Cisco UCS management overview

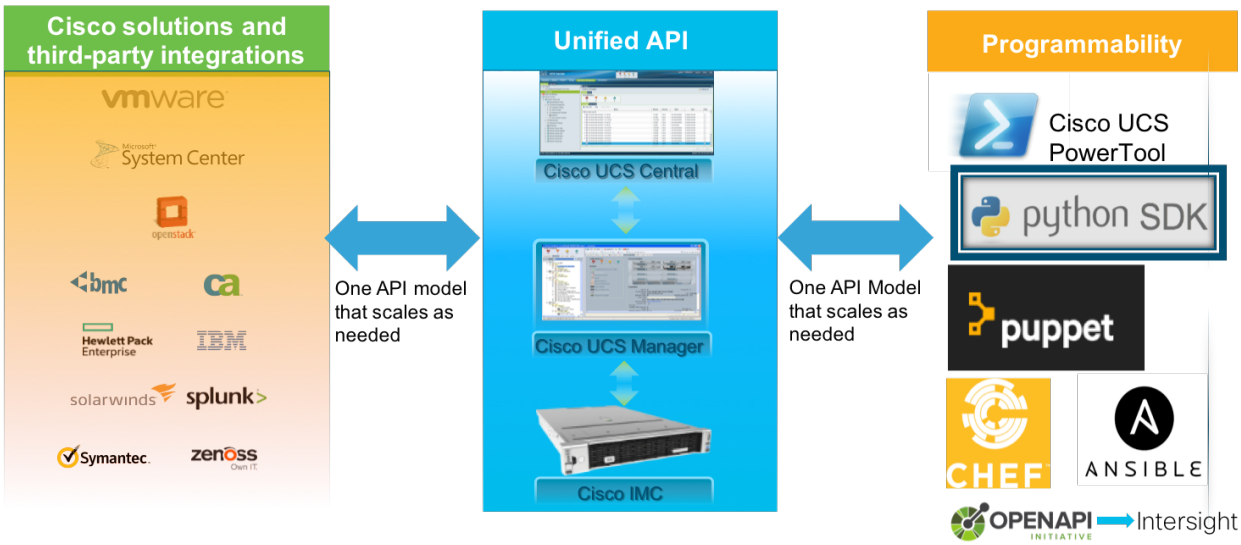
Cisco UCS is the realization of programmable infrastructure. By providing an open full-coverage API, Cisco UCS enables infrastructure developers to manage computing, network, and storage resources with simplicity. Cisco UCS Unified APIs are available in Cisco UCS Manager, Cisco UCS Central Software, and Cisco UCS Integrated Management Controller (IMC). Because these APIs are unified, when you have learned one API, you have also learned the others. The systems or endpoints are represented as hierarchical object models in the management information tree (Figure 1).

Figure 1. Managed objects with Cisco Unified API



Cisco UCS was designed as programmable infrastructure from its inception in 2009. Cisco has continued to enhance the programmability with APIs, tools, orchestration, and integration to address coding requests and requirements (Figure 2).

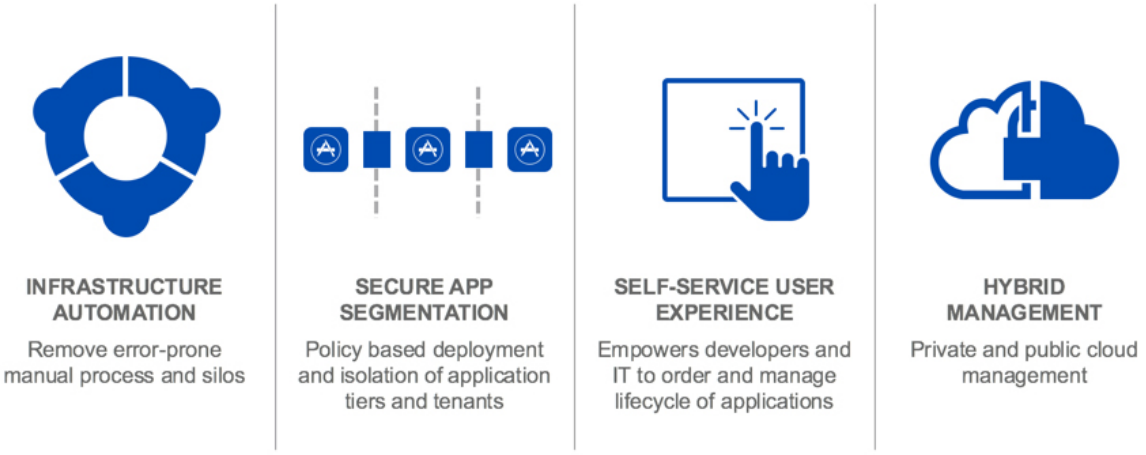
Figure 2. Cisco UCS Unified API



Cisco UCS Director APIs provide access to tasks and workflows that can be used to automate and orchestrate Cisco UCS resources along with other data center resources and give the application and DevOps developers the complete programmatic access they need.

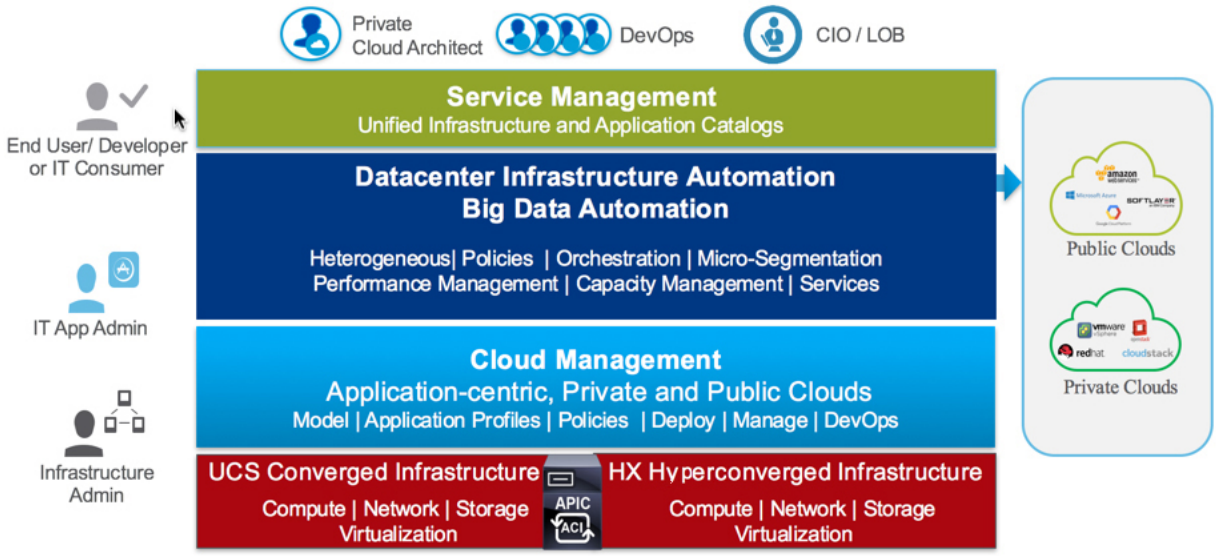
Cisco UCS Unified APIs and Cisco UCS Director APIs work together to enable complete automation and orchestration of the data center. The code samples, blogs, creations, and guides together with the sandboxes, Cisco DevNet support, and community support at communites.cisco.com provide the foundation, and more, for all your data center development needs (Figure 3).

Figure 3. Data center management



In the same way that Cisco UCS Director can consume Cisco UCS management APIs, Cisco UCS Director APIs can be consumed by Cisco CloudCenter™, and Cisco UCS Director can just as easily sit side-by-side with Cisco CloudCenter. The end goal is automation and orchestration of data center resources, and potentially others as well (Figure 4).

Figure 4. Data center automation and orchestration



Cisco UCS unified management

Cisco UCS Manager Release 4, the latest release, brings together in one place support for the third-generation Cisco UCS hardware and the more recent fourth- and fifth-generation Cisco UCS hardware. It includes support for Cisco UCS Mini, Cisco UCS C-Series Rack Servers and B-Series Blade Servers, Cisco HyperFlex™ hyperconverged infrastructure, and Cisco composable infrastructure: the Cisco UCS S3260 Storage Server and Cisco UCS M-Series Modular Servers. It includes an HTML-5 interface and support for latest-generation Cisco UCS 6454 Fabric Interconnect and Cisco UCS Virtual Interface Card (VIC) 1400 platform.

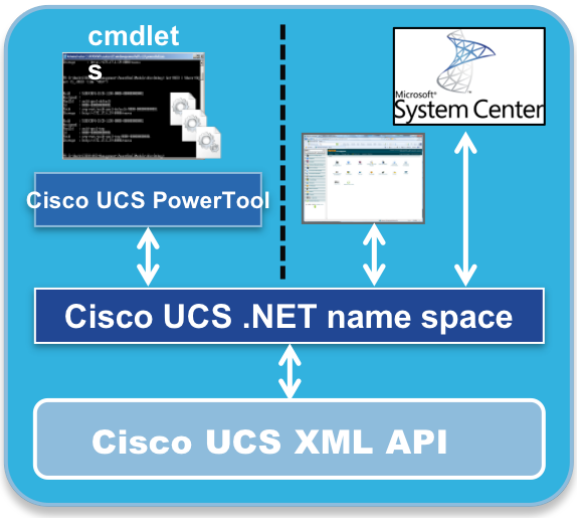
Cisco UCS PowerTool Suite and the Python Software Development Kit (SDK) are built on top of the Cisco UCS management open XML APIs; they compose the XML needed for the API request and parse the response. User guides are available for each of the Cisco UCS XML API options:

- Cisco UCS Manager XML API Programmer's Guide
- Cisco UCS Rack-Mount Servers CIMC XML API Programmer's Guide
- Cisco UCS Central XML API Programmer's Guide

Cisco UCS PowerTool Suite

The Cisco UCS PowerTool Suite (Figure 5) is a Microsoft PowerShell module that helps automate all aspects of Cisco UCS Manager, Cisco UCS Central Software, and Cisco IMC. It also helps automate server, network, storage, and hypervisor management. The PowerTool Suite enables easy integration with existing IT management processes and tools. The PowerTool cmdlets work on the Cisco UCS management information tree. The cmdlets can be used to perform read, create, modify, and delete operations on all the Cisco UCS managed objects in the tree.

Figure 5. Cisco UCS PowerTool Suite



Python SDK for Cisco UCS

The Python SDK for Cisco UCS Manager is a Python module that helps automate and manage configurations within Cisco UCS Manager, including service profiles, policies, pools, equipment, network, and storage.

The Python SDK for Cisco UCS Manager enables easy integration with existing IT management processes and tools. The Python SDK manipulates the Cisco UCS management information tree. The Python SDK allows you to query, create, modify, and delete the managed objects in the tree.

Ansible for Cisco UCS

Ansible for Cisco UCS is built on the Python SDK for Cisco UCS. The new integration of Cisco UCS Manager and Ansible by Red Hat provides a software-defined approach to the management of the entire hardware and software stack. This solution delivers some significant benefits. You can orchestrate all the steps needed to configure the full range of Cisco UCS, Cisco HyperFlex, and converged infrastructure products. As a result, you achieve faster build times, because the entire application stack can be provisioned automatically in minutes. You can also automate Cisco UCS Manager policy, resource pool, and resource profile configuration and ongoing management, including the capability to detect and remediate unintended changes.

Cisco UCS Platform Emulator

The Cisco UCS Platform Emulator is a version of Cisco UCS Manager that runs in a hypervisor (VMware or Microsoft Hyper-V) and is completely free (Cisco.com login is required). No Cisco UCS hardware is required to run the platform emulator.

The Cisco UCS Platform Emulator is designed for the following purposes:

- You can use it to run the Cisco UCS Manager GUI to provide training on Cisco UCS and to run live demonstrations.
- You can use it to learn or develop Cisco UCS PowerTool, Python, and XML scripts and commands. The platform emulator exposes the same XML API as a physical Cisco UCS domain.
- You can use it to stage and model configuration changes. With the platform emulator, you can import the production hardware configuration and production Cisco UCS configurations (pools, policies, templates, and service profiles) and then test-run proposed configuration changes in a risk-free simulated environment that reflects your production environments.

The emulator also includes the complete Cisco UCS Manager object model documentation. This documentation includes all the classes, methods, faults, syslog messages, and more.

Cisco UCS Management Sandbox

Another tool for testing your configurations before you implement them in a production environment is the Cisco UCS Management Sandbox. The management sandbox provides an environment that is suitable for infrastructure, DevOps, and application developers. The sandbox includes the latest shipping product versions of Cisco UCS Director and Cisco UCS Central Software as well as the most recent version of the Cisco UCS Platform Emulator. Additionally, the sandbox launches with both Microsoft Windows and Linux development environments that are already preloaded with Cisco UCS management tools, including the Cisco UCS PowerTool Suite, the Python SDK for Cisco UCS, and Postman for Cisco UCS Director REST API interactions.

PowerShell, Python, Ansible, the emulator, and all Cisco UCS and Cisco HyperFlex products allow you take advantage of all programmatic capabilities that Cisco UCS management options have to offer.

Automation for software-defined storage solutions with Cisco UCS

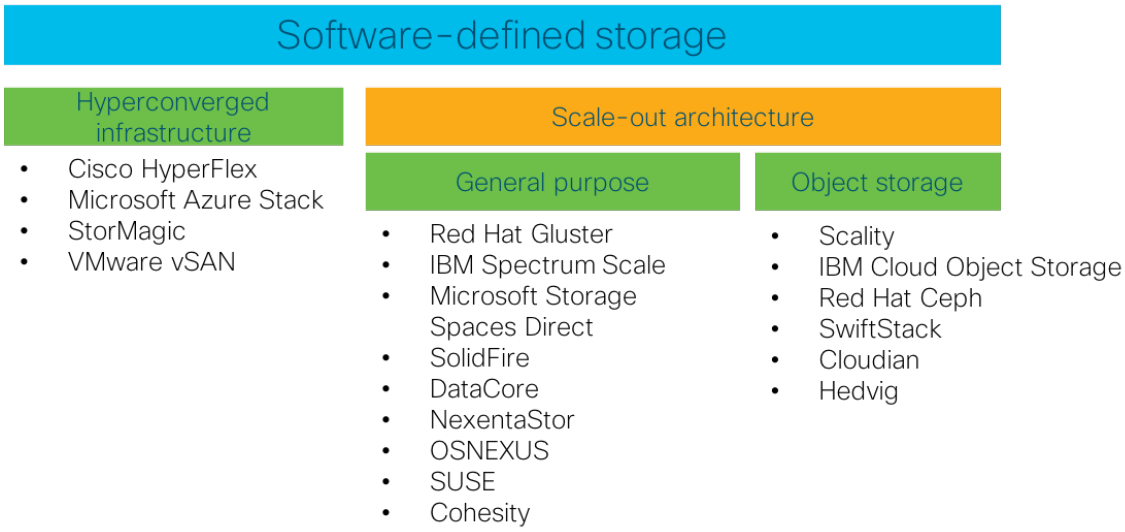
Software-defined storage architecture takes advantage of the superior price and performance of clustered components, facilitates nondisruptive operations, and employs policy-based management for improved efficiency and agility. The combination of software-defined storage architecture and server virtualization capabilities results in a dynamic enterprise data center that can rapidly adapt to changing business requirements, enabling the organization to quickly commission and decommission applications, provision and reprovision resources, and nondisruptively migrate applications and data.

Software-defined storage has a long history with Cisco. In 2009, Cisco introduced the Cisco UCS product line, which aimed to bridge the technology silos to support virtualization. By simplifying and automating tasks notorious for complicating and slowing solution deployment, Cisco UCS has achieved considerable market success. Cisco now is extending the Cisco UCS management framework to a set of software-defined storage solutions for data-intensive computing using Cisco UCS S-Series Storage Servers, and to solutions using Cisco UCS C-Series Rack Servers for smaller deployments. Building on the simplicity of Cisco UCS Manager, Cisco's goal with these solutions is to allow its customers to scale to petabytes within minutes and to experience zero-touch, policy-based management for ongoing operations. In addition, Cisco entered the hyperconverged solution market with the Cisco HyperFlex product family. Hyperconverged solutions are designed to bring web-scale economics to enterprise data centers and private clouds. The hyperconverged product line complements the Cisco UCS solutions for scaling out secondary storage, with both designed to improve economies of scale.

Cisco's current storage solutions for software-defined storage cover three main areas (Figure 6):

- Hyperconverged infrastructure for uses such as virtual desktop infrastructure (VDI) and virtualization with products such as Cisco HyperFlex systems and Microsoft Azure Stack
- Global-purpose scale-out storage for uses such as file serving and high-performance computing with solutions such as Red Hat Gluster Storage and IBM Spectrum Scale
- Object storage for uses such as active archiving and backup operations with solutions such as Scality and IBM Cloud Object Storage

Figure 6. Cisco software-defined storage solution set



The configuration and installation of all these solutions can be fully automated. The Cisco UCS Unified API enables all software-defined storage solutions to be set up in three easy steps:

- Configure computing, networking, and storage through the Cisco UCS Unified API with PowerShell, Python, or Ansible. (You can also use Puppet, Chef, or Redfish.)
- Install the bare-metal OS with preconfigured variables through the Cisco UCS Unified API.
- Install the solution through third-party vendor processes.

This document describes each step to automate various tasks and provides examples using PowerShell, Python, and Ansible. Because Ansible is currently the platform most often used in IT automation, the examples using Ansible are more detailed.

Cisco UCS with Ansible offers the simplest solution for configuration management available. It is designed to be consistent, secure, and highly reliable and with an extremely low learning curve for administrators, developers, and IT managers.

The whole process of software-defined storage automation is implemented in three steps (Figure 7):

- Configure and install the Cisco UCS hardware
- Configure and install the operating system
- Configure and install the third-party software-defined storage solution

Figure 7. Three-step automation framework for software-defined storage with Cisco UCS

Step 1: Configure and install Cisco UCS

The first step in the automation process is to configure and install a complete environment. The entire configuration refers to a Cisco UCS Manager environment. You can also use a pure Cisco IMC environment.

Automation using PowerShell, Python, or Ansible has an advantage over automation using the Cisco UCS Manager GUI in that automation can be implemented completely without the use of a GUI, and the amount of time saved can be immense. Compared to other management platforms, Cisco UCS Manager saves a great deal of time, because the generated service profiles can be widely used, but it always requires the use of the GUI. The APIs, however, avoid this and can configure the hardware more easily and effectively.

In many cases, software-defined storage is used for object storage. Several Cisco Validated Designs show how to configure and install such solutions with Cisco UCS Manager. An integral part of such an environment is the Cisco UCS S3260 dense storage server, which can easily support large capacities. Note, though, that the hard disks used by the Cisco UCS S3260 first must be assigned in Cisco UCS Manager to a corresponding server so that they can then be provided with RAID protection. This and many other processes can be simplified with Ansible or Python automation.

To demonstrate some of the differences among PowerShell, Python, and Ansible, the following example shows the configuration of disks as RAID 0. Each of the 60 hard drives in a Cisco UCS S3260 server is configured as RAID 0. This setup is often used with object storage solutions to make use of the RAID controller's cache.

- Creating RAID 0 disk group policies with PowerShell:

```
Start-UcsTransaction
$mo = Get-UcsOrg -Level root  | Add-UcsLogicalStorageDiskGroupConfigPolicy -Descr "RAID 0 Disk Group for Disk ID $i" -Name "$NameRAID0DiskGroups$i" -PolicyOwner "local" -RaidLevel "stripe"
$mo_1 = $mo | Add-UcsLogicalStorageLocalDiskConfigRef -Role "normal" -SlotNum $i -SpanId "unspecified" /a
$mo_2 = $mo | Set-UcsLogicalStorageVirtualDriveDef -AccessPolicy "read-write" -DriveCache "disable" -IoPolicy "cached" -ReadPolicy "read-ahead" -StripSize "platform-default" -WriteCachePolicy " write-back-good-bbu" -Force
Complete-UcsTransaction
```

- Creating RAID 0 disk group policies with Python:

```
obj = handle.query_dn("org-root")
mo = LstorageDiskGroupConfigPolicy(parent_mo_or_dn=obj, policy_owner="local", raid_level="stripe", name=disk_raid0name + str(i))
mo_1 = LstorageVirtualDriveDef(parent_mo_or_dn=mo, read_policy=read-ahead, drive_cache=disable, strip_size="platform-default", io_policy=cached, write_cache_policy=write-back-good-bbu, security="no", access_policy=read-write)
mo_2 = LstorageLocalDiskConfigRef(parent_mo_or_dn=mo, role="normal", slot_num=str(i), span_id="unspecified")
handle.add_mo(mo, True)
```

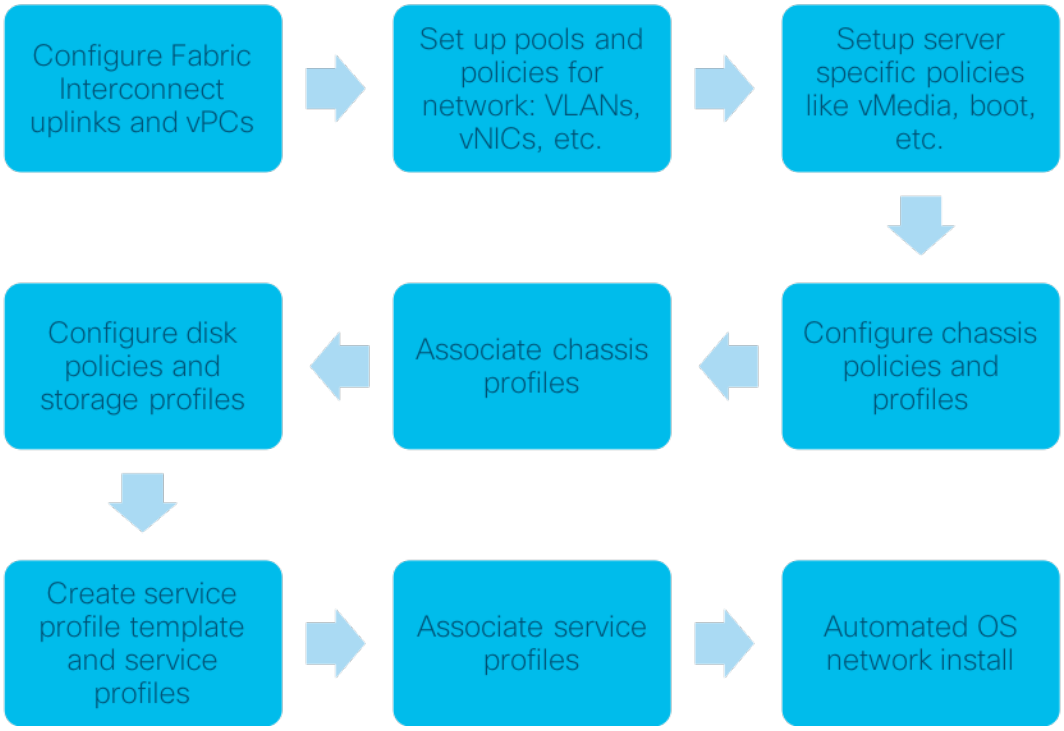
- Creating RAID 0 disk group policies with Ansible:

```
- name: Add Disk Group Policy Protection
```

```
    ucs_disk_group_policy:
      <<: *login_info
      name: Data, 1, 60
      raid_level: stripe
      virtual_drive:
        access_policy: read-write
        drive_cache: disable
        io_policy: cached
        read_policy: read-ahead
        write_cache_policy: write-back-good-bbu
  - name: Add Disk Group Policy Disks
    ucs_disk_group_policy_auto:
      <<: *login_info
      name: Data, 1, 60
      num_drives: 1
      min_drive_size: 10000
```

Figure 8 shows the entire configuration and installation process for software-defined storage on Cisco UCS Manager, with all the steps.

Figure 8. Configuration steps for automation of software-defined storage with Cisco UCS Manager



The first step, before you create a whole block of configurations, is to configure the fabric interconnect uplinks and the virtual port channels (vPCs). Next, before you configure the Cisco UCS S3260 chassis, you create a block of specific server configurations. After you assign the chassis profiles, you configure the disks. Finally, you generate the service profiles and assign them to the servers. The last step is to automatically start the installation of the operating system.

With Ansible, a playbook looks like this:

```
tasks:

  - name: Configure port channels for Fabric Interconnect A.

  - name: Configure port channels for Fabric Interconnect B.

  - name: Configure IPv4 address pools.

  - name: Configure MAC address pool.

  - name: Configure unique user ID (UUID) address pool.

  - name: Configure VLAN.

  - name: Configure network control policy.

  - name: Configure vNIC template.

  - name: Configure VLAN group.
```

```
- name: Configure LAN connectivity policy.
- name: Configure adapter policy.
- name: Configure boot policy.
- name: Configure maintenance.
- name: Configure power control.
- name: Configure virtual media (vMedia) policy.
- name: Configure scrub policy.
- name: Add chassis maintenance policy.
- name: Add chassis firmware policy.
- name: Add chassis connection policy.
- name: Add chassis disk zoning policy.
- name: Add chassis profile template.
- name: Add chassis profile from template.
- name: Change chassis profile association.
- name: Add disk group policy protection.
- name: Add disk group policy.
- name: Add storage profile.
- name: Add local logical unit number (LUN) to storage profile.
- name: Configure service profile template.
- name: Configure service profile from template.
- name: Change service profile association.
```

The processes in the Ansible playbook are largely identical to those in the Cisco Validated Design for Scale-Out Storage with the Cisco UCS S3260 Storage Server.

To implement the preceding configuration and installation processes with Ansible, you have three options:

- Run a complete Ansible role for Cisco UCS manually (see https://github.com/CiscoUcs/ansible-role-ucs).
- Run an Ansible playbook manually:
  - Hard coded
  - With a JSON configuration file
- Use Red Hat Ansible Tower to run Ansible playbooks.

Basically, it doesn't matter which of these methods you use, because they all achieve the same goal. For organizations using Ansible Tower, of course, it's easier to implement the playbooks in Ansible Tower and to run them in Ansible Tower as well. In addition, Ansible has the advantage that you can change even small details later in the playbook, and you can then run the playbook as a whole again. For example, you might later change the size of the maximum transmission unit (MTU) to 9000 for a corresponding vNIC. The following code snippet shows the difference.

Before:

```
- name: Configure vNIC template
  ucs_vnic_template:
    <<: *login_info
    name: vNIC-Storage
    fabric: A-B
    template_type: updating-template
    vlans_list:
    - name: vlan21
      native: 'no'
    network_control_policy: Enable_CDP
    mac_pool: Mac-SDS
```

After:

```
- name: Configure vNIC template
  ucs_vnic_template:
    <<: *login_info
    name: vNIC-Storage
    fabric: A-B
    template_type: updating-template
    mtu: 9000
    vlans_list:
    - name: vlan21
      native: 'no'
    network_control_policy: Enable_CDP
    mac_pool: Mac-SDS
```

In this example, you would run the same Ansible playbook again to make the change in the configuration.

In summary, as you can see in the first step in software-defined storage automation with Cisco UCS, automation is easy to implement with APIs such as PowerTool, Python, and Ansible. Because Ansible now has a fairly wide distribution in the data center area, it is the preferred platform for implementing automation.

Step 2: Configure and install the OS

The second step in the automation of software-defined storage is to install and configure the operating system. This process is fully automated and follows seamlessly from the previous step of configuring and installing the Cisco UCS hardware. Basically, you already performed the configuration step to create the operating system in step 1. With the creation of the vMedia policy with Ansible in Cisco UCS Manager, for example, the Cisco UCS S3260 server is automatically configured and installed as needed for the subsequent processes.

The vMedia policy looks like this:

```
- name: Configure vMedia policy

  ucs_managed_objects:

    <<: *login_info

    objects:

    - {

          "module": "ucsmsdk.mometa.cimcvmedia.CimcvmediaMountConfigPolicy",

          "class": "CimcvmediaMountConfigPolicy",

          "properties": {

              "parent_mo_or_dn": "org-root",

              "name": "SDS"

          },

         "children": [

               {

                  "module": "ucsmsdk.mometa.cimcvmedia.CimcvmediaConfigMountEntry",

                  "class": "CimcvmediaConfigMountEntry",

                  "properties": {

                      "device_type": "cdd",

                      "image_file_name": "rhel-server-7.5-x86_64-dvd-boot.iso",

                      "image_path": "install",

                      "mapping_name": "RHEL75",

                      "mount_protocol": "http",
```

```
                    "remote_ip_address": "10.100.200.35"
                },
            },
            {
                "module": "ucsmsdk.mometa.cimcvmedia.CimcvmediaConfigMountEntry",
                "class": "CimcvmediaConfigMountEntry",
                "properties": {
                    "device_type": "hdd",
                    "image_name_variable": "service-profile-name",
                    "image_path": "install",
                    "mapping_name": "Kickstart",
                    "mount_protocol": "http",
                    "remote_ip_address": "10.100.200.35"
                },
            }
        ]
    }
```

The basis for creating the vMedia policy has already been presented and is extensively described

here (see https://ciscoucs.github.io/site/OS/REDHAT.html). With this procedure, you can create any number of image files, depending on the number of servers that you want to configure and install automatically. After the Cisco UCS Manager service profiles have been created and assigned to the appropriate servers, each server pulls the appropriate image file assigned to it during installation. The result of step 2 is a fully installed server with the appropriate operating system, which can then be used in the installation of the software-defined storage software in step 3. vMedia has an advantage compared to other methods such as the use of preboot execution environment (PXE) servers in that no separate installation needs to be performed in the infrastructure.

For a summary of the automated, complete configuration and installation of Cisco UCS servers and the operating system using Cisco UCS with Ansible, see this video (see https://www.youtube.com/watch?v=MCZvvsBdpS4). As you can see, the association of a Cisco UCS S3260 server takes approximately 15 minutes, and the installation of the operating system takes an additional 25 minutes. The time required for the configuration and installation of the Cisco UCS hardware and the operating system is independent of the number of servers because the process runs in parallel with Ansible.

Step 3: Configure and install third-party software

With the Cisco UCS hardware now configured and installed and the operating system automatically installed for further use, the next step is to automatically install the appropriate software-defined storage software. Different options are available depending on the manufacturer. Again, Ansible can be used to make preconfigure the installation.

Two examples illustrate the process.

Red Hat Ceph Storage

Red Hat Ceph Storage can be implemented based on Ansible configuration and installation and Cisco UCS. As described here, the storage cluster can be prepared relatively simply and then installed using an Ansible playbook. You configure the all.yml, osds.yml, and other .yml files as needed for your request and then simply run the playbook with the following command:

```
[root@admin ceph-ansible]# ansible-playbook site.yml
```

Normally, you would first perform the automated hardware installation with the operating system, so that you can then run regular checks, if you want, if everything works up to this point. This process would then be followed by the automated configuration and installation of Red Hat Ceph Storage. Thus, it would be relatively easy to build a large environment in a few hours.

Scality Ring

Starting with Ring 7.4, configuration files can be fed to the Scality installer to fully automate the Ring installation. The Scality installer menu will not be displayed. You first need to extract all the Scality tools (this step is not needed if you already ran the installer). Then you can generate the configuration file. With the configuration file named main.conf, you can simply run the automated installation with the following command:

```
[root@supervisor ~]# ./scality-ring-7.4.0.0.run --conf main.conf
```

Again, the entire installation is relatively simple, and large environments can be set up quickly and easily.

Conclusion

Automation in the IT world is becoming increasingly important. With automation, large environments, including software-defined storage, can be set up relatively easily and quickly. A platform such as Ansible can help build such large environments and enable you to change them quickly, according to your requirements. However, the prerequisite for this process is an API to the hardware that can implement the process. Cisco UCS provides such an API, and Cisco UCS with Ansible makes it possible to automate software-defined storage environments automatically and easily. You can deploy with confidence this flexible and scalable solution to meet your critical data, services, and application needs today as well as your evolving needs in the future.

For more information

For additional information, see the following:

- Cisco Validated Design for Scale-Out Storage
- GitHub: Cisco UCS
- Cisco blog: Scaling to PB Within Minutes—The Road to Full Automation for Scale-Out Storage with Cisco UCS

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C11-741259-00 08/18