

Deployment and Operations of MinIO Object Storage with Cisco Intersight Appliance and Terraform

Contents

Executive summary	3
Introduction	3
Solution design	4
Deployment hardware and software	11
Configure MinIO infrastructure with Terraform	18
MinIO object storage	29
High performance	29
Kubernetes native	29
Scalability	29
Multitenant	29
Secure	29
MinIO erasure coding and hardware sizing	30
MinIO erasure code calculator	30
MinIO installation and configuration	31
Day-2 operations for MinIO with Cisco Intersight and Terraform	36
Appendix	56
Conclusion	73
For more information	73

This document describes the deployment and operation of MinIO Object Storage on Cisco UCS with the Cisco Intersight appliance and Terraform provider for Cisco Intersight. In addition, it describes typical Day-2 operations on cloud-scale storage with Cisco Intersight and Terraform provider for Cisco Intersight.

Executive summary

This document describes the deployment and Day-2 operations of MinIO on the Cisco Unified Computing System™ (Cisco UCS®) C240 Rack Servers. It provides the framework of deploying MinIO software on Cisco UCS C240 Rack Servers together with Cisco Intersight™ appliances. Cisco UCS provides the storage, network, and storage access components for MinIO, deployed as a single cohesive system.

With the continuous evolution of Software Defined Storage (SDS), demand to have MinIO solutions validated on Cisco UCS servers has increased. The Cisco UCS C240 Rack Server, originally designed for the data center, together with MinIO is optimized for such object storage solutions, making it an excellent fit for use cases such as secondary storage, disaster recovery, and archiving, and it is unmatched at overcoming the challenges associated with machine learning, analytics, and cloud-native application workloads.

Cisco and MinIO offer a solution that solves the problem of connecting storage and managing data effectively. Cisco UCS provides an enterprise-grade compute, network, and storage infrastructure, building the foundation for MinIO. To offer a more intelligent level of management that enables IT organizations to analyze, simplify, and automate their environments, the Cisco Intersight solution as a systems management platform plays a major role in building and managing the infrastructure. With its ability to build, change, and keep versions of infrastructure up-to-date safely and efficiently, Terraform is an ideal tool for building and managing these hybrid cloud storage infrastructures.

Introduction

The challenge of operating a data center has never been greater. But the opportunities have never been larger. Today, data-driven businesses are changing rapidly to stay competitive. There are more applications than ever, in addition to more different classes of people and machines using the applications. And there's constant change in the shape of business infrastructures, as data "centers" become less centralized thanks to edge- and cloud-based storage and compute resources.

The opportunity in IT is to enable these changes. When doing it well, IT is in strategic partnership with business. Being flexible and responsive to business means that strategic planners will find encouragement for their plans in the technology core where we work, making IT leaders "copilots" in business growth.

But the only way for IT to offer this level of service is to have comprehensive understanding of what the data center does, how everyone and everything uses it, how it responds to the demands on it, and most importantly, how it will respond to new loads that arise as new business processes develop.

Some of these capabilities are part of running a network, compute, or storage, what is now often called Day-2 Operations (Ops). To make Day-2 Ops easier for customers, especially the compute and storage aspects of their operations, Cisco Intersight technology brings all the capabilities of configuration, provisioning, and installation. Policy-based profiles and templates for deployment, configuration, and the creation of multiple server profiles enable customers to consistently deploy and configure servers, eliminating configuration errors and minimizing configuration drift. These capabilities enable people in IT operations to do their jobs more easily, and in the process add value to the data center and the business itself.

To enhance the capabilities of the Cisco Intersight appliance and to add an automation part to the process, Terraform Provider for Cisco Intersight enables IT operations to be more effective and run recurring jobs more easily. By defining an Infrastructure as code environment, users can easily make and process changes to the environment at any time.

Storage environments, especially scale-out ones, do have such regular Day-2 operations. Starting small but growing fast comes with many operations in parallel; for example, adding more disks, expanding a cluster with more nodes, attaching more network interfaces to increase the bandwidth, or replacing failed disks. Because of the recurring tasks in a scale-out storage cluster, Cisco Intersight and Terraform Provider for Cisco Intersight offer great benefits.

To look into typical Day-2 operations for scale-out storage, we used MinIO, a high-performance, distributed object storage system. MinIO is a cloud-native object server that is simultaneously performant, scalable, and lightweight. Although MinIO excels at traditional object storage use cases such as secondary storage, disaster recovery, and archiving, it is unique at overcoming the challenges associated with machine learning, analytics, and cloud-native application workloads. MinIO operates on commodity servers with locally attached drives (“just a bunch of disks”/“just a bunch of flash” [JBOD/JBOF]). All the servers in a cluster are equal in capability (fully symmetrical architecture). No name nodes or metadata servers are needed. MinIO is designed for large-scale, multi-data center cloud storage services. Each tenant runs its own MinIO cluster, fully isolated from other tenants, protecting them from any disruption on upgrade, update, or security incidents. Each tenant scales independently by federating clusters across geographies.

This document describes the deployment and Day-2 operation procedures of MinIO on Cisco UCS C240 M5 servers together with Cisco Intersight and Terraform Provider for Cisco Intersight.

Solution design

Solution overview

In this architecture, we have MinIO deployed on Cisco UCS with Cisco Intersight and Terraform provider for Cisco Intersight. We automatically set up four Cisco UCS C240 M5L servers with Terraform provider for Cisco Intersight, simplifying the process of orchestrating a scale-out storage environment. We expanded the running MinIO cluster with two more nodes to show Day-2 example operations. All six servers were installed with the latest Red Hat Enterprise Linux 8 operating system.

Note: The recommendation from MinIO for expansions is a minimum of four nodes. In our tests, we could expand with only two nodes because of the limited availability of hardware.

We manually deployed the Cisco Intersight virtual appliance and Terraform provider as virtual machines. Both virtual machines were deployed on a Cisco UCS HyperFlex™ Edge cluster, connected to a pair of Cisco Nexus® switches. The Cisco HyperFlex™ Edge cluster is not part of the overall deployment, but it fits well into the overall solution because of its ability to host multiple virtual machines by simple deployment and management. Figure 1 shows an overview of this solution.

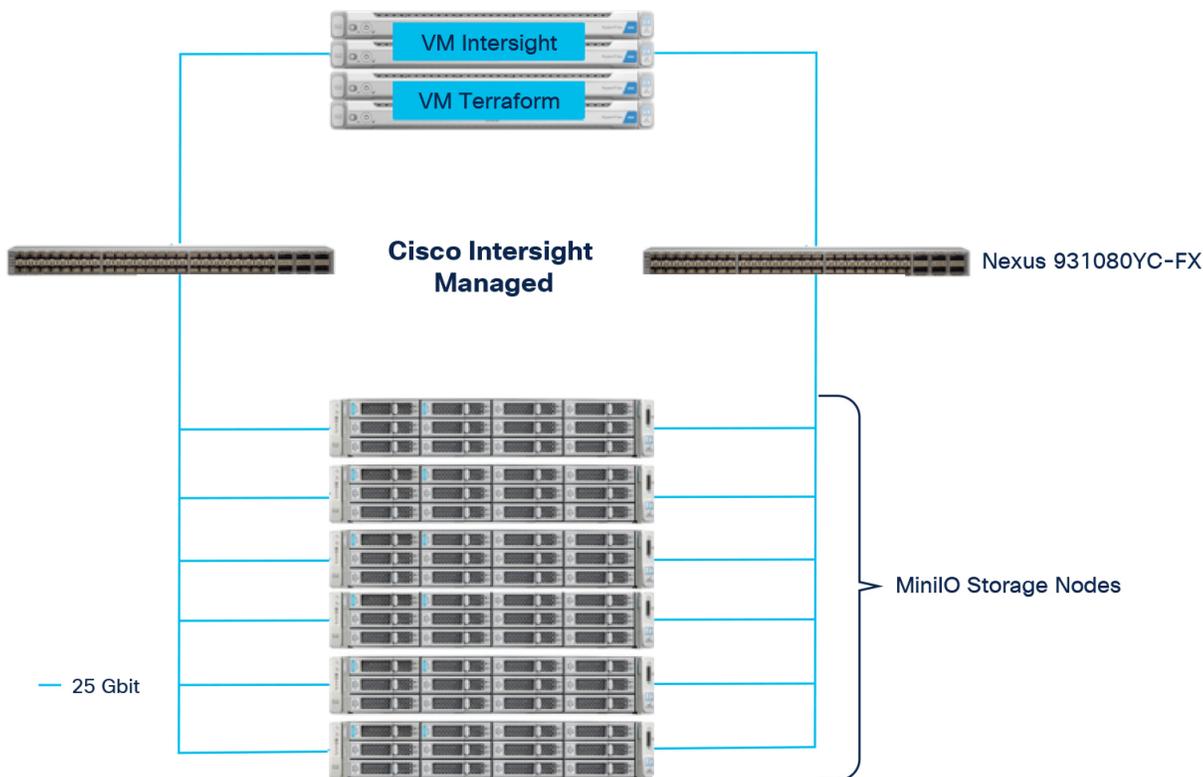


Figure 1.
Solution overview

Solution flow

The solution setup consists of multiple parts. It covers basic setup of the network components, policies, and profiles, and installations of various parts as well. It shows typical Day-2 operations with Cisco Intersight and Terraform and a basic performance and high-availability testing. The high-level flow of the solution setup follows:

1. Install and configure Cisco UCS C240 M5 with Cisco Intersight and Terraform provider for Cisco Intersight.
2. Deploy Red Hat Enterprise Linux and MinIO.
3. Perform functional tests of the whole solution.
4. Expand MinIO cluster with disks, nodes, and network ports through Cisco Intersight and Terraform.
5. Replace a failed disk.

Requirements

The following sections detail the physical hardware, software revisions, and firmware versions required to install a single MinIO cluster on Cisco UCS. These requirements are specific to the solution built in this white paper.

Table 1 lists the hardware components of the network built in this paper.

Table 1. Hardware components used in this white paper

Component	Model	Quantity	Comments
Switches	Cisco Nexus® 93180YC-FX	2	
Cisco UCS	Cisco UCS C240 M5L	6	Each node: 2 x Intel Xeon Silver 4214R (2.4 GHz, 12 cores) 384-GB Memory Cisco 12-Gigabit Ethernet (GE) modular Redundant Array of Independent Disks (RAID) controller with 2-GB cache 2 x 960-GB 6-Gbps SATA SSD for system 12 x 10-TB 12-Gbps NL-SAS HDD for data 1 x VIC 1455
Cisco Intersight virtual appliance	Virtual machine	1	16 vCPU 32-GB memory 500-GB disk 1 x network
Terraform	Virtual machine	1	2 vCPU 16-GB memory 100-GB disk 1 x network

Software components

Table 2 lists the required software distribution versions for the solution.

Table 2. Software versions

Layer	Component	Version or release
Cisco UCS C240 M5L	Firmware version	4.1(3b)
Network Nexus 93180YC-FX	BIOS	07.67
	NXOS	9.3(4)
Cisco Intersight virtual appliance	Version	1.0.9-302
Software	Terraform	0.14.9
Software	Terraform provider for Intersight	1.0.15
Software	Red Hat Enterprise Linux	8.4
Software	MinIO	MinIO version Release.2021-08-05T22-01-19Z

Physical topology

Topology overview

The solution contains one topology configuration. There are six Cisco UCS C240 M5 servers connected to a pair of Cisco Nexus 93180YC-FX switches. Each Cisco UCS C240 M5 server relates to one 25-Gbps cable to each Cisco Nexus 93180YC-FX. All six Cisco UCS C240 M5 servers use a Red Hat Enterprise Linux (RHEL) network interface card (NIC) teaming with a “load-balance” runner to achieve high availability and high performance.

Figure 2 illustrates the details of the configuration.

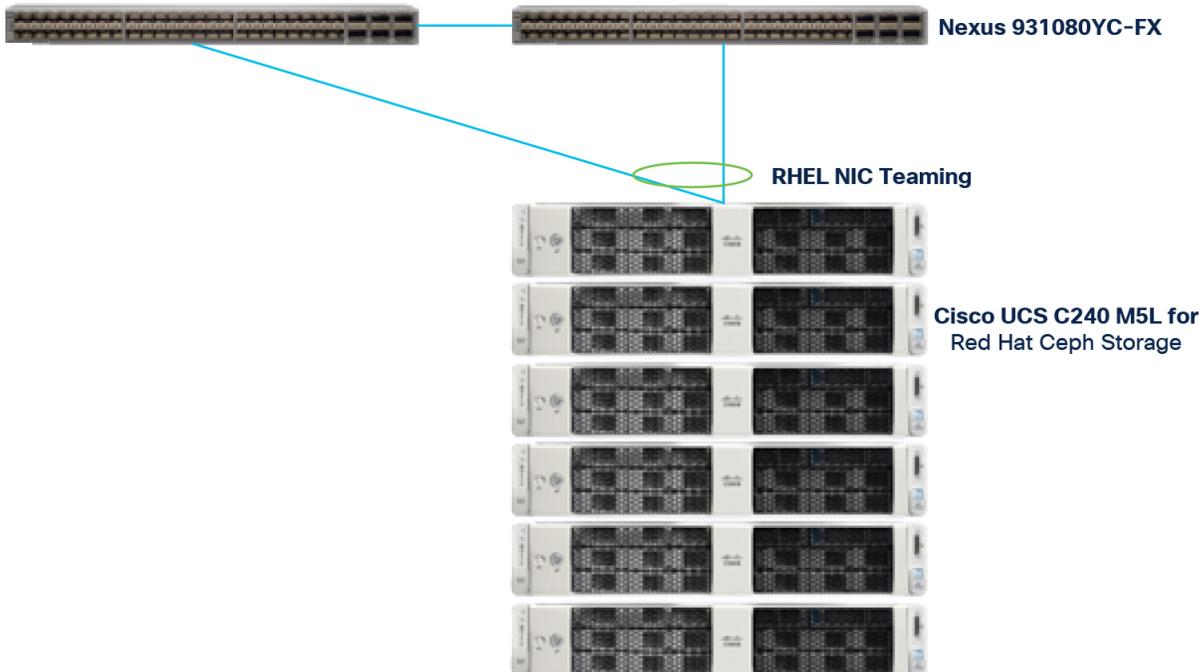


Figure 2.
Datacenter topology

Network design - VLAN and subnets

For the base configuration, multiple virtual LANs (VLANs) must be carried to the Cisco UCS domain from the upstream LAN; these VLANs are also defined in the Cisco UCS configuration. Table 3 lists the VLANs that the Cisco Intersight appliance created for use in this testing and their functions. VLAN configuration is not a requirement for MinIO and is specific to this exercise, but we recommend using VLANs in general to separate network traffic securely.

Table 3. VLANs and subnets

VLAN name	VLAN ID	Subnet	Purpose
Management	300	172.16.32.0/24	Cisco UCS Integrated Management Controller (Cisco IMC) management interfaces Cisco Intersight appliance Terraform Administration network for MinIO

VLAN name	VLAN ID	Subnet	Purpose
Client	301	172.16.33.0/24	Client network for MinIO
Storage	302	172.16.34.0/24	Storage network for MinIO

Jumbo frames

All traffic traversing the client and storage VLAN, and subnet is configured by default to use jumbo frames, or to be precise, all communication is configured to send IP packets with a Maximum Transmission Unit (MTU) size of 9000 bytes. Using a larger MTU value means that each IP packet sent carries a larger payload, so it transmits more data per packet, and consequently sends and receives data faster. MinIO does not require jumbo frames, and enabling them is specific to this exercise.

Naming scheme and DNS

We highly recommend that you configure Domain Name System (DNS) servers for querying Fully Qualified Domain Names (FQDNs). You must create DNS records before you begin the installation. At a minimum, we highly recommend that you create A records and reverse PTR records.

Table 4 lists the required DNS information for the installation.

Table 4. DNS server information

Item	Name
DNS server	192.168.10.51
DNS domain	sjc02dmz.net
vCenter server name	sjc02dmz-vcsa
Cisco Nexus® 93180YC-FX #1	sjc02dmz-i14-n93180ycfx-a
Cisco Nexus 93180YC-FX #2	sjc02dmz-i14-n93180ycfx-b
Cisco Intersight virtual appliance	sjc02dmz-intersight
Cisco UCS C240 M5 #1	minio1
Cisco UCS C240 M5 #2	minio2
Cisco UCS C240 M5 #3	minio3
Cisco UCS C240 M5 #4	minio4
Cisco UCS C240 M5 #5	minio5
Cisco UCS C240 M5 #6	minio6
Terraform	sjc02dmz-i14-terraform

Cabling

The physical layout of the solution was previously described in the section “Topology Overview”. The Cisco Nexus switches and the Cisco UCS server must be cabled properly before you begin the installation activities. Table 5 provides the cabling map for installation of a MinIO solution on Cisco UCS.

Table 5. Cabling map of Cisco Nexus 93180YC-FX

Device	Port	Connected To	Port	Note
sjc02dmz-i14-n93180ycfx-a	1	minio1	Port 0	
sjc02dmz-i14-n93180ycfx-a	2	minio2	Port 0	
sjc02dmz-i14-n93180ycfx-a	3	minio3	Port 0	
sjc02dmz-i14-n93180ycfx-a	4	minio4	Port 0	
sjc02dmz-i14-n93180ycfx-a	5	minio5	Port 0	
sjc02dmz-i14-n93180ycfx-a	6	minio6	Port 0	
sjc02dmz-i14-n93180ycfx-a	12	minio1	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	13	minio2	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	14	minio3	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	15	minio4	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	16	minio5	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	17	minio6	Port 1	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-a	49	sjc02dmz-i14-n93180ycfx-b	Eth1/49	vPC peer link
sjc02dmz-i14-n93180ycfx-a	50	sjc02dmz-i14-n93180ycfx-b	Eth1/50	vPC peer link
sjc02dmz-i14-n93180ycfx-b	1	minio1	Port 2	
sjc02dmz-i14-n93180ycfx-b	2	minio2	Port 2	
sjc02dmz-i14-n93180ycfx-b	3	minio3	Port 2	
sjc02dmz-i14-n93180ycfx-b	4	minio4	Port 2	
sjc02dmz-i14-n93180ycfx-b	5	minio5	Port 2	
sjc02dmz-i14-n93180ycfx-b	6	minio6	Port 2	
sjc02dmz-i14-n93180ycfx-b	12	minio1	Port 3	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-b	13	minio2	Port 3	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-b	14	minio3	Port 3	Day 2 – Add network
sjc02dmz-i14-n93180ycfx-b	15	minio4	Port 3	Day 2 – Add network

sjc02dmz-i14-n93180ycfx-b	16	minio5	Port 3	Day 2 - Add network
sjc02dmz-i14-n93180ycfx-b	17	minio6	Port 3	Day 2 - Add network
sjc02dmz-i14-n93180ycfx-b	49	sjc02dmz-i14-n93180ycfx-a	Eth1/49	vPC peer link
sjc02dmz-i14-n93180ycfx-b	50	sjc02dmz-i14-n93180ycfx-a	Eth1/50	vPC peer link

Rack layout

The core solution with the C240 M5L takes 12 rack units (12RU) of space in a standard rack. The HyperFlex cluster is not required, but it hosts the Cisco Intersight virtual appliance and the Terraform administration host. Figure 3 shows the rack layout.



Figure 3.
Rack layout

Deployment hardware and software

Create a Terraform configuration environment for Cisco Intersight appliance

Before you start the automated configuration, you need to prepare the environment:

- Install Terraform provider.
- Generate application-programming-interface (API) keys.
- Define Cisco Intersight provider.
- Configure variables.

Install Terraform

You will install Terraform on an administration host; in our solution, we used a virtual Linux RHEL machine. HashiCorp distributes Terraform as a binary package. You can also install Terraform using popular package managers.

To install Terraform, follow these steps:

1. Obtain the [appropriate package](#) for your system and download it as a zip archive.
2. After downloading Terraform, unzip the package. Terraform runs as a single binary named terraform. You can safely remove any other files in the package and Terraform will still function. (You can also [compile the Terraform binary](#) from source.)
3. Make sure that the terraform binary is available on your PATH. This process will differ depending on your operating system.

```
[root@sjc02dmz-i14-terraform ~]# ll
total 16448
-rw-----. 1 root root      1812 Jun 15 10:49 anaconda-ks.cfg
-rw-r--r--. 1 root root 34869112 Jun 16 06:23 terraform_0.14.9_linux_amd64.zip
[root@sjc02dmz-i14-terraform ~]# unzip terraform_0.14.9_linux_amd64.zip
Archive:  terraform_0.14.9_linux_amd64.zip
  inflating: terraform
[root@sjc02dmz-i14-terraform ~]# ll
total 68080
-rw-----. 1 root root      1812 Jun 15 10:49 anaconda-ks.cfg
-rwxr-xr-x. 1 root root 85545348 May 27 09:38 terraform
-rw-r--r--. 1 root root 34869112 Jun 16 06:23 terraform_0.14.9_linux_amd64.zip
```

4. Move the terraform binary to one of the listed locations. The following command assumes that the binary is currently in your downloads folder and that your PATH includes /usr/local/bin, but you can customize it if your locations are different.

```
[root@sjc02dmz-i14-terraform ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@sjc02dmz-i14-terraform ~]# mv ~/terraform /usr/local/bin/terraform
```

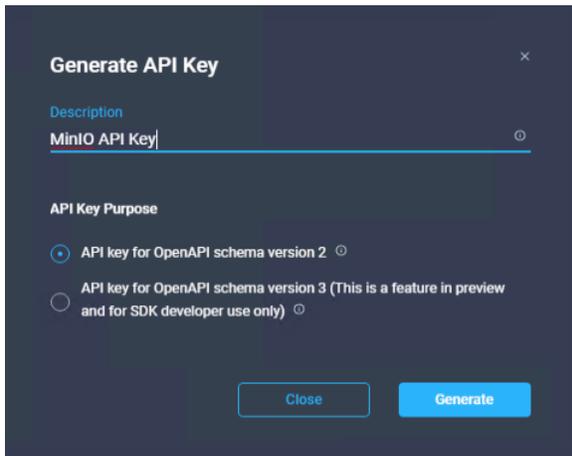
5. Verify the installation:

```
[root@sjc02dmz-i14-terraform ~]# terraform -version
Terraform v0.14.9
```

Generate Cisco Intersight API keys

To start using the provider, the API Key, Secret Key, and Intersight endpoint Uniform Resource Locators (URLs) are required. To generate the API Keys, follow these steps:

1. Log in to your Cisco Intersight virtual appliance.
2. Go to Settings, API Keys, and click Generate API Keys.
3. Enter a description and click Generate.



4. Copy the API key.
5. Save the secret key into a .pem file on your Terraform administration host.

Configure variables

To provision the infrastructure, you need to define variables for various workflows, including:

- Virtual LANs (VLANs)
- Remote server hosting images
- Remote server share
- Remote server OS image
- Remote server HUU image
- Remote server protocol
- Managed object ID for all nodes that needs to be provisioned
- Managed object ID for organization
- Managed object ID for catalog

Before you download and store all necessary images, you need to install and configure a HTTP server on the Terraform administration host. Download the specific RHEL ISO images for each MinIO node from the Terraform host:

```
[root@sjc02dmz-i14-terraform]# dnf -y install httpd genisoimage
[root@sjc02dmz-i14-terraform]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service →
/usr/lib/systemd/system/httpd.service.
```

```
[root@sjc02dmz-i14-terraform]# systemctl start httpd
[root@sjc02dmz-i14-terraform]# firewall-cmd --zone=public --permanent --add-service=http
success
[root@sjc02dmz-i14-terraform]# firewall-cmd --reload
Success
[root@sjc02dmz-i14-terraform]# mkdir -p /var/www/html/images
```

1. Create a Terraform module, which helps to get the Managed Object ID (MOID) for each server, the organization, and the catalog for the OS installation. A module is a container for multiple resources that are used together. Every Terraform configuration has at least one module, known as its *root module*, which consists of the resources defined in the .tf files in the main working directory. A module can call other modules, allowing you to include the resources of the child module in the configuration concisely. You also can call modules multiple times, either within the same configuration or in separate configurations, allowing you to package and re-use resource configurations.
2. Create a subdirectory where you can configure a main.tf and variables.tf file for the module. This step helps ensure that the MOIDs are automatically retrieved for the main configuration file:

```
[root@sjc02dmz-i14-terraform]# mkdir terraform-intersight-moids
[root@sjc02dmz-i14-terraform ~]# cd terraform-intersight-moids
[root@sjc02dmz-i14-terraform terraform-intersight-moids]# vi variables.tf
# Server and Organization names
variable "server_names" {
    type = list
}

variable "organization_name" {}

variable "catalog_name" {}

[root@sjc02dmz-i14-terraform terraform-intersight-moids]# vi main.tf
# Intersight provider Information
terraform {
    required_providers {
        intersight = {
            source = "CiscoDevNet/intersight"
            version = "1.0.12"
        }
    }
}

data "intersight_compute_physical_summary" "server_moid" {
    name = var.server_names[count.index]
    count = length(var.server_names)
}
```

```

output "server_moids" {
  value = data.intersight_compute_physical_summary.server_moid.*.results.0.moid
}

data "intersight_organization_organization" "organization_moid" {
  name = var.organization_name
}

output "organization_moid" {
  value = data.intersight_organization_organization.organization_moid.results[0].moid
}

data "intersight_softwarerepository_catalog" "catalog_moid" {
  name = var.catalog_name
}

output "catalog_moid" {
  value = data.intersight_softwarerepository_catalog.catalog_moid.results[2].moid
}

```

3. Create another subdirectory for the configuration files for the MinIO nodes and configure the main variables file. Split the whole configuration into several tasks to have a better overview and different subdirectories:

```

[root@sjc02dmz-i14-terraform]# mkdir terraform-minio
[root@sjc02dmz-i14-terraform ~]# cd terraform-minio
[root@sjc02dmz-i14-terraform terraform-minio]# mkdir create_policy_profile
deploy_profile install_os
[root@sjc02dmz-i14-terraform terraform-minio]# cd create_policy_profile
[root@sjc02dmz-i14-terraform create_policy_profile]# vi variables.tf
//Define all the basic variables here

variable "api_private_key" {
  default = "/root/terraform-minio/intersight.pem"
}

variable "api_key_id" {
  default = "5e5fb2b17564612d3028b5b4/5e5fbd137564612d3028bcc4/5fa1a9107564612d3007f934"
}

variable "api_endpoint" {
  default = "https://sjc02dmz-intersight.sjc02dmz.net"
}

```

```
variable "management_vlan" {
    default = 300
}

variable "client_vlan" {
    default = 301
}

variable "storage_vlan" {
    default = 302
}

variable "remote-server" {
    default = "sjc02dmz-i14-terraform.sjc02dmz.net"
}

variable "remote-share" {
    default = "/images"
}

variable "remote-os-image-minio" {
    type = list(string)
    default = ["rhel8.2-minio1.iso", "rhel8.2-minio2.iso", "rhel8.2-minio3.iso", "rhel8.2-
minio4.iso"]
}

variable "remote-os-image-link" {
    type = list(string)
    default = ["http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio1.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio2.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio3.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio4.iso"]
}

variable "remote-protocol" {
    default = "softwarerepository.HttpServer"
}

variable "server_names" {
    default = ["sjc02dmz-i14-c240m511", "sjc02dmz-i14-c240m512", "sjc02dmz-i14-c240m513",
"sjc02dmz-i14-c240m514"]
}
```

```

variable "organization_name" {
  default = "Minio"
}

variable "server_profile_action" {
  default = "No-op"
}

variable "catalog_name" {
  default = "appliance-user-catalog"
}

```

Define Cisco Intersight provider

Information about the Cisco Intersight provider is available at:

<https://registry.terraform.io/providers/CiscoDevNet/intersight/latest> To enable and define the Cisco Intersight provider for the MinIO Storage solution, follow these steps, which are also documented at the right site of the webpage under “Use Provider”:

1. On the Terraform administration host, go into one of the subdirectories under terraform-intersight-sds and create a main.tf file:

```

terraform {
  required_providers {
    intersight = {
      source = "CiscoDevNet/intersight"
      version = "1.0.12"
    }
  }
}

provider "intersight" {
  apikey      = var.api_key_id
  secretkey   = var.api_private_key
  endpoint    = var.api_endpoint
}

```

Note: This step will be used for all tasks in each main.tf file.

Understand Cisco Intersight provider and Terraform configuration

The following is an example code snippet for creating a specific virtual network interface card (vNIC) from the infrastructure file (understand and create the main configuration file):

```

resource "intersight_vnic_eth_if" "eth0" { -> Define the resource
  name = "eth0"
  order = 0
  placement { -> Define the placement of the vNIC

```

```

    id      = "MLOM"
    pci_link = 0
    uplink  = 0
  }
  cdn {
    nr_source = "vnic"
  }
  vmq_settings {
    enabled = false
    num_interrupts = 1
    num_vmqs = 1
  }
  lan_connectivity_policy { -> Define LAN Connectivity Policy to use
    moid      = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
    object_type = "vnic.LanConnectivityPolicy"
  }
  eth_network_policy { -> Define the Network Policy to use
    moid = intersight_vnic_eth_network_policy.minio-mgt-network.id
  }
  eth_adapter_policy { -> Define the Adapter Policy to use
    moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
  }
  eth_qos_policy { -> Define the QoS Policy to use
    moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-policy.id
  }
}

```

Each resource is assigned a name, which you can use later for tracking and referencing. This name will not be reflected anywhere in the Cisco Intersight platform. It is only for reference among the .tf files. A resource can point to or reference another resource using the format <resource>.<resource_name>.<property_name>.

Documentation about provider resources and configuration options is available at:

<https://github.com/CiscoDevNet/terraform-provider-intersight/tree/master/website/docs>.

Implement Terraform configuration - init, plan, apply

After creating all the configuration files and the main infrastructure files, the next step is to validate and deploy the configuration.

```
terraform plan
```

The `terraform plan` command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

This command is a convenient way to check whether the execution plan for a set of changes matches the expectations without making any changes to real resources or to the state. For example, you might run `terraform plan` before you commit a change to version control, to create confidence that it will behave as expected.

In the output, the symbols show you the following:

- Resources with a plus sign (+) will be created.
- Resources with a minus sign (-) will be deleted.
- Resources with a tilde (~) will be modified in place.

The `terraform apply` command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a `terraform plan` execution plan.

Configure MinIO infrastructure with Terraform

The configuration to automatically prepare the environment for the following MinIO installation consists of three steps. All these steps were run in subdirectories for a better overview. They can also run in just one configuration file and one directory.

1. Create Policies and Profiles for four Cisco UCS C240 M5 servers.
2. Deploy Profiles.
3. Install custom RHEL 8 ISO images on all nodes.

Create Cisco Intersight Policies and Profiles with Terraform

You can now start creating the policies you need for the MinIO solution and build the server profiles out of the policies. Terraform will build the following policies:

Table 6. Terraform provider Policies and Resource Objects

Policy	Terraform resource object	Comments
Adapter configuration	<code>intersight_adapter_config_policy</code>	Specify the PCI slot ID where the Cisco Virtual Interface Card (VIC) adapter is placed and set Forward Error Correction (FEC) mode for 25-GE connectivity. Configured: <ul style="list-style-type: none">• Slot modular LAN-on-Motherboard (MLOM)• Forward Error Correction (FEC) mode cl74
Ethernet adapter	<code>intersight_vnic_eth_adapter_policy</code>	Specify the adapter properties to improve the throughput over network. Configured: <ul style="list-style-type: none">• Interrupt 11• Completion 9• Rx count 8 and ring size 4096• Tx count 1 and ring size 4096• Really Simple Syndication (RSS) true
Ethernet network	<code>intersight_vnic_eth_network_policy</code>	Specify the network and VLANs used for MinIO, in our cases three networks with different VLANs. Configured: <ul style="list-style-type: none">• Management network VLAN 300• Client network VLAN 301• Storage network VLAN 302

Ethernet Quality of Service (QoS)	intersight_vnic_eth_qos_policy	Specify the Quality of Service with MTU size 9000. Configured: <ul style="list-style-type: none"> • MTU 9000
LAN connectivity	inter-sight_vnic_lan_connectivity_policy intersight_vnic_eth_if	Specify the LAN connectivity with the vNICs. Configured: <ul style="list-style-type: none"> • eth0 (uplink port 0) for Management network • eth1 (uplink port 0), eth2 (uplink port 1) for Client network • eth3 (uplink port 0), eth4 (uplink port 1) for Storage network
Network Time Protocol (NTP)	intersight_ntp_policy	Specify the NTP servers to be used Configured: <ul style="list-style-type: none"> • NTP IP 173.38.201.115
Drive group	intersight_storage_drive_group	Specify the drive group for boot disks (Redundant Array of Independent Disks [RAID] 1). Configured: <ul style="list-style-type: none"> • Slot 13 and 14 RAID 1 for boot (C240 M5) • Virtual disk for boot • ReadWrite, ReadAhead, and WriteBackGoodBBU
Storage	intersight_storage_storage_policy	Specify the Storage Policies with the previous created drive group.
Boot order	intersight_boot_precision_policy	Specify the boot order. Configured: <ul style="list-style-type: none"> • Local disk from MegaRAID • Virtual media from Cisco IMC mapped DVD

According to <https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/unified-computing-system-adapters/white-paper-c11-744754.html#AdapterpolicywithRSS>

After creating the policies, use the same task to create the server profiles for all four MinIO nodes. To start building the policies and profiles, log in to the Terraform administration host and follow these steps:

```
[root@sjc02dmz-i14-terraform ~]# cd terraform-minio/create_policy_profile/
[root@sjc02dmz-i14-terraform create_policy_profile]# terraform init -upgrade
Upgrading modules...
- intersight-moids in ../../terraform-intersight-moids

Initializing the backend...

Initializing provider plugins...
- Finding ciscodevnet/intersight versions matching "1.0.12"...
- Installing ciscodevnet/intersight v1.0.12...
- Installed ciscodevnet/intersight v1.0.12 (signed by a HashiCorp partner, key ID
7FA19DB0A5A44572)

Partner and community providers are signed by their developers.
```

If you'd like to know more about provider signing, you can read about it here:
<https://www.terraform.io/docs/cli/plugins/signing.html>.

Terraform has made some changes to the provider dependency selections recorded in the `.terraform.lock.hcl` file. Review those changes and commit them to your version control system if they represent changes you intended to make.

Terraform has been successfully initialized.

You may now begin working with Terraform. Try running `terraform plan` to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

```
[root@sjc02dmz-i14-terraform create_policy_profile]# terraform plan
... -> We skip the full output since it is very lengthy.
Plan: 22 to add, 0 to change, 0 to destroy.
[root@sjc02dmz-i14-terraform create_policy_profile]# terraform apply
... -> We skip the full output since it is very lengthy.
Plan: 22 to add, 0 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_server_profile.minio[1]: Creating...
intersight_vnic_eth_network_policy.minio-client-network: Creating...
intersight_storage_disk_group_policy.minio-disk-group-boot-policy-c240: Creating...
intersight_vnic_eth_qos_policy.minio-ethernet-qos-1500-policy: Creating...
intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy: Creating...
intersight_server_profile.minio[2]: Creating...
intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy: Creating...
intersight_vnic_eth_network_policy.minio-mgt-network: Creating...
intersight_vnic_eth_network_policy.minio-storage-network: Creating...
intersight_server_profile.minio[0]: Creating...
intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy: Creation complete after 1s
[id=6098e9ea612a22d47257ffed]
intersight_server_profile.minio[3]: Creating...
intersight_server_profile.minio[1]: Creation complete after 1s [id=6098e9e977696e2d30dbc00a]
```

```
intersight_storage_disk_group_policy.minio-disk-group-boot-policy-c240: Creation complete
after 1s [id=6098e9ea656f6e2d30c44c22]
intersight_vnic_eth_network_policy.minio-mgt-network: Creation complete after 1s
[id=6098e9ea612a22d47257fff4]
intersight_server_profile.minio[2]: Creation complete after 1s [id=6098e9ea77696e2d30dbc017]
intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy: Creation complete after 1s
[id=6098e9ea612a22d47257fffc]
intersight_vnic_eth_qos_policy.minio-ethernet-qos-1500-policy: Creation complete after 1s
[id=6098e9ea612a22d472580003]
intersight_server_profile.minio[0]: Creation complete after 1s [id=6098e9ea77696e2d30dbc023]
intersight_vnic_eth_network_policy.minio-client-network: Creation complete after 1s
[id=6098e9ea612a22d472580009]
intersight_vnic_eth_network_policy.minio-storage-network: Creation complete after 1s
[id=6098e9ea612a22d472580010]
intersight_server_profile.minio[3]: Creation complete after 0s [id=6098e9ea77696e2d30dbc02f]
intersight_boot_precision_policy.minio-boot-policy: Creating...
intersight_storage_storage_policy.minio-storage-policy: Creating...
intersight_adapter_config_policy.minio-adapter-config-policy: Creating...
intersight_ntp_policy.minio-ntp-policy: Creating...
intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy: Creating...
intersight_networkconfig_policy.minio-network-policy: Creating...
intersight_storage_storage_policy.minio-storage-policy: Creation complete after 1s
[id=6098e9eb656f6e2d30c44c2d]
intersight_adapter_config_policy.minio-adapter-config-policy: Creation complete after 1s
[id=6098e9eb612a22d472580022]
intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy: Creation complete after
1s [id=6098e9eb612a22d472580032]
intersight_vnic_eth_if.eth4: Creating...
intersight_vnic_eth_if.eth0: Creating...
intersight_vnic_eth_if.eth1: Creating...
intersight_vnic_eth_if.eth2: Creating...
intersight_vnic_eth_if.eth3: Creating...
intersight_vnic_eth_if.eth4: Creation complete after 0s [id=6098e9ec612a22d472580052]
intersight_vnic_eth_if.eth2: Creation complete after 0s [id=6098e9ec612a22d472580058]
intersight_vnic_eth_if.eth3: Creation complete after 0s [id=6098e9ec612a22d47258005d]
intersight_vnic_eth_if.eth1: Creation complete after 1s [id=6098e9ec612a22d472580063]
intersight_vnic_eth_if.eth0: Creation complete after 1s [id=6098e9ec612a22d472580068]
intersight_boot_precision_policy.minio-boot-policy: Creation complete after 2s
[id=6098e9eb6275722d30929e4f]
intersight_ntp_policy.minio-ntp-policy: Creation complete after 2s
[id=6098e9ec6275722d30929e6b]
intersight_networkconfig_policy.minio-network-policy: Creation complete after 2s
[id=6098e9ec6275722d30929e7f]
Apply complete! Resources: 22 added, 0 changed, 0 destroyed.
```

You can now view in Intersight under Policies the newly created policies.

Name	Platform Type	Type	Usage	Last Update
minio-network-policy	UCS Server, UCS Domain	Network Connectivity	4	3 minutes ago
minio-ntp-policy	UCS Server, UCS Domain	NTP	4	3 minutes ago
minio-boot-policy	UCS Server	Boot Order	4	3 minutes ago
minio-lan-connectivity-policy	UCS Server	LAN Connectivity	4	3 minutes ago
minio-adapter-config-policy	UCS Server	Adapter Configuration	4	3 minutes ago
minio-storage-policy	UCS Server	Storage	4	3 minutes ago
minio-disk-group-boot-policy-c240	UCS Server	Disk Group	0	3 minutes ago
minio-storage-network	UCS Server	Ethernet Network	0	3 minutes ago
minio-client-network	UCS Server	Ethernet Network	0	3 minutes ago
minio-ethernet-qos-1500-policy	UCS Server	Ethernet QoS	0	3 minutes ago
minio-ethernet-adapter-policy	UCS Server	Ethernet Adapter	0	3 minutes ago
minio-mgt-network	UCS Server	Ethernet Network	0	3 minutes ago
minio-ethernet-qos-9000-policy	UCS Server	Ethernet QoS	0	3 minutes ago

Under Profiles you can view the four new server profiles for the MinIO nodes.

Name	Status	Target Platform	Server	Last Update
SP-sjc02dmz-i14-c240m514	Not Deployed	UCS Server (Standalone)	sjc02dmz-i14-c240m514	5 minutes ago
SP-sjc02dmz-i14-c240m513	Not Deployed	UCS Server (Standalone)	sjc02dmz-i14-c240m513	5 minutes ago
SP-sjc02dmz-i14-c240m512	Not Deployed	UCS Server (Standalone)	sjc02dmz-i14-c240m512	5 minutes ago
SP-sjc02dmz-i14-c240m511	Not Deployed	UCS Server (Standalone)	sjc02dmz-i14-c240m511	5 minutes ago

The formal task of creating policies and profiles is now finished; in the next step the server profiles get associated with the selected servers.

Associate Cisco Intersight Profiles with Terraform

1. Associate the former created server profiles with Terraform to the physical servers. Use the same variables.tf file from the task before. The configuration file is located in the deploy profile subdirectory.
2. Log into the Terraform administration server and run the following commands:

```
[root@sjc02dmz-i14-terraform ~]# cd terraform-minio/deploy_profile/
[root@sjc02dmz-i14-terraform deploy_profile]# cp ../create_policy_profile/variables.tf .
[root@sjc02dmz-i14-terraform deploy_profile]# terraform init -upgrade
[root@sjc02dmz-i14-terraform deploy_profile]# terraform apply
... -> We skip the full output as it is very lengthy.
Plan: 4 to add, 0 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_server_profile.minio[1]: Creating...
```

```
intersight_server_profile.minio[3]: Creating...
```

```

intersight_server_profile.minio[0]: Creating...
intersight_server_profile.minio[2]: Creating...
intersight_server_profile.minio[3]: Creation complete after 0s
[id=6098e9ea77696e2d30dbc02f]
intersight_server_profile.minio[1]: Creation complete after 0s
[id=6098e9e977696e2d30dbc00a]
intersight_server_profile.minio[0]: Creation complete after 0s
[id=6098e9ea77696e2d30dbc023]
intersight_server_profile.minio[2]: Creation complete after 0s
[id=6098e9ea77696e2d30dbc017]

```

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

3. Verify the deployment by checking the status in Cisco Intersight under Profiles.

Name	Status	Target Platform	Server	Last Update
SP-sjc02dmz-14-c240m511	OK	UCS Server (Standalone)	sjc02dmz-14-c240m511	a few seconds ago
SP-sjc02dmz-14-c240m513	OK	UCS Server (Standalone)	sjc02dmz-14-c240m513	a minute ago
SP-sjc02dmz-14-c240m514	OK	UCS Server (Standalone)	sjc02dmz-14-c240m514	2 minutes ago
SP-sjc02dmz-14-c240m512	OK	UCS Server (Standalone)	sjc02dmz-14-c240m512	2 minutes ago

Prepare custom RHEL ISO images for automated installation

Before installing the OS, you need to create the custom images for each node. The goal is to have four individual installations, each for every MinIO node. For that task, we created custom ISO files, which include a kickstart file with all the specific details for the MinIO node. An example kickstart file for minio1 follows:

```

lang en_US.UTF-8
keyboard --vckeymap=us --xlayouts='us'
timezone --isUtc America/Los_Angeles --ntpservers=173.38.201.115
# System services
services --enabled="chronyd"
rootpw
$6$dA8apVZJJhnc1jrS$IuVqcdAuHQVijluX6S6vw88FYteyog12ZZczrFDRhIROiteIWI41SjPSsgNgIoVGB3YanQGm.
lyWsK7v48P81 --iscrypted
#platform x86, AMD64, or Intel EM64T
cdrom
reboot
#Network Information
network --bootproto=static --device=eth0 --ip=172.16.32.101 --netmask=255.255.255.0 --
gateway=172.16.32.1 --hostname=minio1.sjc02dmz.net --nameserver=192.168.10.51 --noipv6 --
mtu=1500 --onboot=on --activate
network --bootproto=static --device=team1 --ip=172.16.33.101 --netmask=255.255.255.0 --noipv6
--mtu=9000 --onboot=on --activate --teamslaves="eth1,eth2" --teamconfig="{\"runner\":
{\"name\": \"loadbalance\"}}}"
network --bootproto=static --device=team2 --ip=172.16.34.101 --netmask=255.255.255.0 --noipv6
--mtu=9000 --onboot=on --activate --teamslaves="eth3,eth4" --teamconfig="{\"runner\":
{\"name\": \"loadbalance\"}}}"

```

```

bootloader --location=mbr --append="rhgb quiet crashkernel=auto" --boot-drive=/dev/disk/by-
path/pci-0000:18:00.0-scsi-0:2:0:0
clearpart --all --initlabel
zerombr
# Disk partitioning information
part pv.1 --fstype="lvm pv" --ondisk=/dev/disk/by-path/pci-0000:18:00.0-scsi-0:2:0:0 --
size=890000
part /boot --fstype="xfs" --ondisk=/dev/disk/by-path/pci-0000:18:00.0-scsi-0:2:0:0 --size=1024
volgroup minio --pesize=4096 pv.1
logvol /home --fstype="xfs" --size=10240 --name=home --vgname=minio
logvol swap --fstype="swap" --size=4096 --name=swap --vgname=minio
logvol / --fstype="xfs" --size=204800 --name=root --vgname=minio
logvol /var --fstype="xfs" --grow --size=1 --name=var --vgname=minio
logvol /tmp --fstype="xfs" --size=40960 --name=tmp --vgname=minio
auth --passalgo=sha512 --useshadow
firewall --disabled
firstboot --disable
ignoredisk --only-use=/dev/disk/by-path/pci-0000:18:00.0-scsi-0:2:0:0

%packages
@^minimal-environment
chrony
kexec-tools
%end

%addon com_redhat_kdump --enable --reserve-mb='auto'

%end

```

To create a custom ISO for RHEL 8.2, follow these steps:

1. Mount the DVD ISO:

```

[root@sjc02dmz-i14-terraform ~]# mount -o loop /tmp/rhel-8.2-x86_64-dvd.iso /mnt
mount: /mnt: WARNING: device write-protected, mounted read-only.

```

2. Create a directory and copy all the content of the ISO:

```

[root@sjc02dmz-i14-terraform ~]# shopt -s dotglob
[root@sjc02dmz-i14-terraform ~]# mkdir /tmp/rhel8
[root@sjc02dmz-i14-terraform ~]# cp -avRf /mnt/* /tmp/rhel8

```

3. Verify that all hidden files, such as .treeinfo, are there in /tmp/rhel8:

```

[root@sjc02dmz-i14-terraform ~]# cd /tmp/rhel8
[root@sjc02dmz-i14-terraform rhel8]# ls -a
.      addons      EFI      extra_files.json  images      LiveOS      Packages  RPM-GPG-KEY-
redhat-beta  TRANS.TBL

```

```
.. .discinfo EULA GPL isolinux media.repo repodata RPM-GPG-KEY-
redhat-release .treeinfo
```

4. Get the kickstart file and copy it to /tmp/rhel8:

```
[root@sjc02dmz-i14-terraform rhel8]# cp /root/ks-minio1.cfg /tmp/rhel8/
```

5. Confirm the LABEL of the DVD iso, which provides the LABEL information.

```
[root@sjc02dmz-i14-terraform rhel8]# blkid /tmp/rhel-8.2-x86_64-dvd.iso
rhel-8.2-x86_64-dvd.iso: BLOCK_SIZE="2048" UUID="2020-10-09-06-40-37-00" LABEL="RHEL-8-2-0-BaseOS-x86_64" TYPE="iso9660" PTUUID="4c667155" PTTYPE="dos"
```

6. Add the following part in the /tmp/rhel8/isolinux/isolinux.cfg file as follows. Make sure that the part has inst.stage2 and the correct label. Remove “menu default” from “label check” and change timeout to 100.

```
label kickstart
    menu label ^Kickstart Installation of RHEL8.2
    kernel vmlinuz
    menu default
    append initrd=initrd.img inst.stage2=hd:LABEL=RHEL-8-2-0-BaseOS-x86_64
inst.ks=cdrom:/ks-minio1.cfg net.ifnames=0 biosdevname=0
```

7. Save the file and create the ISO as follows. Make sure that -V has the correct LABEL; a mistake will cause the DVD not to work.

```
[root@sjc02dmz-i14-terraform rhel8]# mkisofs -o /tmp/rhel8.2-minio1.iso -b
isolinux/isolinux.bin -J -R -l -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -
boot-info-table -eltorito-alt-boot -e images/efiboot.img -no-emul-boot -graft-points -V
"RHEL-8-2-0-BaseOS-x86_64" .
[root@sjc02dmz-i14-terraform rhel8]# mv ../rhel8.2-minio1.iso /var/www/html/images/
```

8. Repeat steps 1–7 for minio2–4 with the different kickstart file and move them to /var/www/html/images.

Automate install of RHEL OS with Terraform

After creating policies and profiles and assigning and deploying profiles, in the last steps you will do an automated install of RHEL 8.2 on all MinIO nodes. The process contains two actions:

- Create the software repository with the OS image for each node for the environment.
- Trigger all nodes to reboot and boot via vMedia from the OS image.

For that you need to specify one configuration file, main.tf. It creates the software repositories in the Cisco Intersight appliance and reboots the nodes and installs the OS based on the software repository.

To do an automated OS install with Terraform under Cisco Intersight, follow these steps:

1. Log into the Terraform server and go to subdirectory for OS install:

```
[root@sjc02dmz-i14-terraform ~]# cd terraform-minio/install_os/
[root@sjc02dmz-i14-terraform install_os]# cp ../create_policy_profile/variables.tf .
```

2. Run **terraform plan** to see whether your configuration runs through:

```
[root@sjc02dmz-i14-terraform install_os]# terraform init -upgrade
[root@sjc02dmz-i14-terraform install_os]# terraform plan
... -> We skip the full output as it is very lengthy.
Plan: 8 to add, 0 to change, 0 to destroy.
```

Note: You didn't specify an "-out" parameter to save this plan, so Terraform can't guarantee that exactly these actions will be performed if "terraform apply" is subsequently run.

3. If your configuration is good, run the **apply** command to deploy the OS:

```
[root@sjc02dmz-il4-terraform install_os]# terraform apply
... -> We skip the full output as it is very lengthy.
Plan: 8 to add, 0 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[3]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[0]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[1]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[2]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[3]: Creation complete after 1s [id=609936466567612d308c1bd3]
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[2]: Creation complete after 1s [id=609936466567612d308c1bdb]
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[0]: Creation complete after 1s [id=609936466567612d308c1be2]
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[1]: Creation complete after 1s [id=609936466567612d308c1be9]
intersight_os_install.minio[0]: Creating...
intersight_os_install.minio[1]: Creating...
intersight_os_install.minio[3]: Creating...
intersight_os_install.minio[2]: Creating...
intersight_os_install.minio[0]: Still creating... [10s elapsed]
intersight_os_install.minio[1]: Still creating... [10s elapsed]
intersight_os_install.minio[3]: Still creating... [10s elapsed]
intersight_os_install.minio[2]: Still creating... [10s elapsed]
intersight_os_install.minio[0]: Still creating... [20s elapsed]
intersight_os_install.minio[1]: Still creating... [20s elapsed]
intersight_os_install.minio[3]: Still creating... [20s elapsed]
intersight_os_install.minio[2]: Still creating... [20s elapsed]
```

```
intersight_os_install.minio[0]: Still creating... [30s elapsed]
intersight_os_install.minio[1]: Still creating... [30s elapsed]
intersight_os_install.minio[3]: Still creating... [30s elapsed]
intersight_os_install.minio[2]: Still creating... [30s elapsed]
intersight_os_install.minio[0]: Still creating... [40s elapsed]
intersight_os_install.minio[1]: Still creating... [40s elapsed]
intersight_os_install.minio[3]: Still creating... [40s elapsed]
intersight_os_install.minio[2]: Still creating... [40s elapsed]
intersight_os_install.minio[0]: Still creating... [50s elapsed]
intersight_os_install.minio[1]: Still creating... [50s elapsed]
intersight_os_install.minio[3]: Still creating... [50s elapsed]
intersight_os_install.minio[2]: Still creating... [50s elapsed]
intersight_os_install.minio[0]: Still creating... [1m0s elapsed]
intersight_os_install.minio[1]: Still creating... [1m0s elapsed]
intersight_os_install.minio[3]: Still creating... [1m0s elapsed]
intersight_os_install.minio[2]: Still creating... [1m0s elapsed]
intersight_os_install.minio[0]: Still creating... [1m10s elapsed]
intersight_os_install.minio[1]: Still creating... [1m10s elapsed]
intersight_os_install.minio[3]: Still creating... [1m10s elapsed]
intersight_os_install.minio[2]: Still creating... [1m10s elapsed]
intersight_os_install.minio[0]: Still creating... [1m20s elapsed]
intersight_os_install.minio[1]: Still creating... [1m20s elapsed]
intersight_os_install.minio[3]: Still creating... [1m20s elapsed]
intersight_os_install.minio[2]: Still creating... [1m20s elapsed]
intersight_os_install.minio[0]: Still creating... [1m30s elapsed]
intersight_os_install.minio[1]: Still creating... [1m30s elapsed]
intersight_os_install.minio[3]: Still creating... [1m30s elapsed]
intersight_os_install.minio[2]: Still creating... [1m30s elapsed]
intersight_os_install.minio[0]: Still creating... [1m40s elapsed]
intersight_os_install.minio[1]: Still creating... [1m40s elapsed]
intersight_os_install.minio[3]: Still creating... [1m40s elapsed]
intersight_os_install.minio[2]: Still creating... [1m40s elapsed]
intersight_os_install.minio[0]: Still creating... [1m50s elapsed]
intersight_os_install.minio[1]: Still creating... [1m50s elapsed]
intersight_os_install.minio[3]: Still creating... [1m50s elapsed]
intersight_os_install.minio[2]: Still creating... [1m50s elapsed]
intersight_os_install.minio[0]: Still creating... [2m0s elapsed]
intersight_os_install.minio[1]: Still creating... [2m0s elapsed]
intersight_os_install.minio[3]: Still creating... [2m0s elapsed]
intersight_os_install.minio[2]: Still creating... [2m0s elapsed]
intersight_os_install.minio[0]: Still creating... [2m10s elapsed]
intersight_os_install.minio[1]: Still creating... [2m10s elapsed]
```

```
intersight_os_install.minio[3]: Still creating... [2m10s elapsed]
intersight_os_install.minio[2]: Still creating... [2m10s elapsed]
intersight_os_install.minio[0]: Still creating... [2m20s elapsed]
intersight_os_install.minio[1]: Still creating... [2m20s elapsed]
intersight_os_install.minio[3]: Still creating... [2m20s elapsed]
intersight_os_install.minio[2]: Still creating... [2m20s elapsed]
intersight_os_install.minio[1]: Creation complete after 2m29s
[id=60993648930d877110aeb753]
intersight_os_install.minio[0]: Still creating... [2m30s elapsed]
intersight_os_install.minio[3]: Still creating... [2m30s elapsed]
intersight_os_install.minio[2]: Still creating... [2m30s elapsed]
intersight_os_install.minio[0]: Creation complete after 2m32s
[id=60993648930d877110aeb71d]
intersight_os_install.minio[2]: Creation complete after 2m33s
[id=60993648930d877110aeb738]
intersight_os_install.minio[3]: Still creating... [2m40s elapsed]
intersight_os_install.minio[3]: Still creating... [2m50s elapsed]
intersight_os_install.minio[3]: Still creating... [3m0s elapsed]
intersight_os_install.minio[3]: Still creating... [3m10s elapsed]
intersight_os_install.minio[3]: Still creating... [3m20s elapsed]
intersight_os_install.minio[3]: Still creating... [3m30s elapsed]
intersight_os_install.minio[3]: Still creating... [3m40s elapsed]
intersight_os_install.minio[3]: Still creating... [3m50s elapsed]
intersight_os_install.minio[3]: Still creating... [4m0s elapsed]
intersight_os_install.minio[3]: Still creating... [4m10s elapsed]
intersight_os_install.minio[3]: Still creating... [4m20s elapsed]
intersight_os_install.minio[3]: Still creating... [4m30s elapsed]
intersight_os_install.minio[3]: Still creating... [4m40s elapsed]
intersight_os_install.minio[3]: Still creating... [4m50s elapsed]
intersight_os_install.minio[3]: Still creating... [5m0s elapsed]
intersight_os_install.minio[3]: Still creating... [5m10s elapsed]
intersight_os_install.minio[3]: Still creating... [5m20s elapsed]
intersight_os_install.minio[3]: Still creating... [5m30s elapsed]
intersight_os_install.minio[3]: Still creating... [5m40s elapsed]
intersight_os_install.minio[3]: Still creating... [5m50s elapsed]
intersight_os_install.minio[3]: Still creating... [6m0s elapsed]
intersight_os_install.minio[3]: Still creating... [6m10s elapsed]
intersight_os_install.minio[3]: Still creating... [6m20s elapsed]
intersight_os_install.minio[3]: Creation complete after 6m29s
[id=60993646930d877110aeb702]
```

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

The setup of the Cisco UCS environment for the MinIO nodes is now finished and in the next step you will deploy MinIO.

MinIO object storage

MinIO is a high-performance, Kubernetes native object storage suite. With an extensive list of enterprise features, it is scalable, secure, and resilient while remaining remarkably simple to deploy and operate at scale. Software-defined, MinIO runs on any hardware and in any cloud—public, private, or edge. MinIO is the world's fastest object storage and can run the broadest set of workloads in the industry. It is widely considered to be the leader in compatibility with Amazon's S3 application programming interface (API).

High performance

With its focus on high performance, MinIO enables enterprises to support multiple use cases with the same platform. For example, the MinIO performance characteristics allow you to run multiple Spark, Presto, and Hive queries, or to quickly test, train, and deploy artificial intelligence (AI) algorithms, without suffering a storage bottleneck. MinIO object storage is used as the primary storage for cloud-native applications that require higher throughput and lower latency than traditional object storage can provide.

Kubernetes native

MinIO was born into the cloud. Containerization and orchestration are our customer base's default implementations. Our focus on the cloud-native way has resulted in an exceptionally clean Kubernetes implementation. As a result, MinIO can be found on every Kubernetes distribution from the public cloud to the private cloud.

Scalability

MinIO employs the same techniques as the hyper-scalers using simplicity as a building block. By consuming resources in small, discrete chunks, MinIO customers can create multiple instances and pack them densely—up to exabyte scale. Cloud-native applications start small at the prototyping phase, but often grow to multiple racks and sometimes to multiple data centers spread across geographies. MinIO is designed to scale smoothly alongside the demands of the application.

Multitenant

MinIO is designed for the multitenancy world of Kubernetes. With MinIO, tenants are fully isolated from each other with their own instances of MinIO clusters. Each tenant in turn may have multiple users with varying levels of access privileges. Tenant clusters operate independently of each other and, because of the size of the MinIO binary (<100 MB), you can pack them densely for efficiency. You also can employ standard HTTP load balancers or round-robin DNS.

Secure

You can configure MinIO to encrypt data when stored on disk and when transmitted over the network. The state-of-the-art encryption schemes of MinIO support granular object-level encryption using modern, industry-standard encryption algorithms, such as AES-256-GCM, ChaCha20-Poly1305, and AES-CBC. MinIO is fully compatible with S3 encryption semantics, and also extends S3 by including support for non-

Amazon Web Services (AWS) key management services such as HashiCorp Vault, Gemalto KeySecure, and Google Secrets Manager.

MinIO erasure coding and hardware sizing

MinIO protects data with per-object, inline erasure coding that is written in assembly code to deliver the highest performance possible. MinIO uses Reed-Solomon code to stripe objects into data and parity blocks, although you can configure these blocks to any desired redundancy level. Therefore, in a 12-drive setup with 6 parity configurations, an object is striped across as 6 data and 6 parity blocks. Even if you lose as many as 5, or $((n/2)-1)$, drives, parity or data, you can still reconstruct the data reliably from the remaining drives. MinIO implementation helps ensure that objects can be read, or new objects written even if multiple devices are lost or unavailable.

Erasure code protects data without the limitations of RAID configurations or data replicas. For example, RAID-6 protects against only a two-drive failure, whereas erasure code allows MinIO to continue to serve data even with the loss of up to 50 percent of the drives and 50 percent of the servers. Replication results in three or more copies of the object on each of the sites. Erasure code offers a significantly higher level of protection while consuming only a fraction of the storage space as compared to replication.

Finally, MinIO applies erasure code to individual objects, thereby allowing the healing at an object-level granularity. For RAID-protected storage solutions, healing is done at the RAID block layer, and it affects the performance of every file stored on the volume until the healing is completed.

MinIO erasure code calculator

MinIO provides an erasure code calculator that you can use to determine raw and usable capacity across a range of hardware configurations and erasure coding settings. This tool is frequently used for capacity planning. Enter the number of servers in your planned cluster, the number of drives per server, the drive capacity, and select erasure code stripe size and parity settings to calculate the expected usable capacity. A best practice is to expand servers in multiples of four or more.

Erasure Code Calculator

This calculator will help you determine your raw and usable capacity across a range of erasure coding settings.

Number of Servers	8
Number of Drives per Server	16
Drive Capacity	8 TB
Erasure Code Stripe Size (K+M)	16
Erasure Code Parity (M)	4

Usable Capacity	768 TB
Raw Capacity	1.0 PB
Storage Efficiency	75%
Drive Failure(s) Tolerance	32 drives in total (4 out of 16 drives)
Server Failure(s) Tolerance	2 servers in total

Email Me

MinIO installation and configuration

The following procedures document the installation and configuration of MinIO. The installation process involves two unique stages:

1. Prepare the MinIO nodes.
2. Install and configure MinIO.

Prerequisites

After setting up the Cisco UCS environment with the Cisco Intersight appliance, Terraform, and Red Hat Enterprise Linux 8.2, you need to have an active subscription to update all nodes with the latest patches and to install MinIO.

Note: Contact your Red Hat sales representative for a subscription to Red Hat Enterprise Linux.

Further prerequisites for the current installation follow:

- Verify the operating system.
- Register MinIO nodes for RHEL.

Preparation of MinIO nodes

To prepare all nodes, follow these steps:

1. Log in to one MinIO node, register the node, and attach the subscription.

```
[root@minio1 ~]# subscription-manager register
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: XXX
Password: YYY
The system has been registered with ID: 36df4df1-d570-41fb-a52c-f9ef1e6b948e
The registered system name is: minio1.sjc02dmz.net
[root@minio1 ~]# subscription-manager refresh
All local data refreshed
[root@minio1 ~]# subscription-manager list --available
[root@minio1 ~]# subscription-manager attach --pool=XXX
[root@minio1 ~]# subscription-manager repos --disable=*
[root@minio1 ~]# subscription-manager repos --enable=rhel-8-for-x86_64-baseos-rpms --
enable=rhel-8-for-x86_64-appstream-rpms
Repository 'rhel-8-for-x86_64-baseos-rpms' is enabled for this system.
Repository 'rhel-8-for-x86_64-appstream-rpms' is enabled for this system.
[root@minio1 ~]# dnf -y update
```

2. Repeat step 1 for all MinIO nodes.

Install and configure MinIO

In the next step, all systems will prepare to run MinIO. We used an administration host to run the commands.

```
[root@sjc02dmz-rhel ~]# for i in {1..4}; do ssh -t root@minio$i "for j in {a..h}; do wpefs -
af /dev/sd$j; done"; done
[root@sjc02dmz-rhel ~]# clush -g minio "dnf -y install wget"
```

```

[root@sjc02dmz-rhel ~]# clush -g minio "wget https://dl.minio.io/server/minio/release/linux-
amd64/minio"
[root@sjc02dmz-rhel ~]# clush -g minio "chmod +x minio"
[root@sjc02dmz-rhel ~]# clush -g minio "mv minio /usr/local/bin"
[root@sjc02dmz-rhel ~]# clush -g minio "minio --version"
172.16.32.104: minio version RELEASE.2021-08-05T22-01-19Z
172.16.32.103: minio version RELEASE.2021-08-05T22-01-19Z
172.16.32.102: minio version RELEASE.2021-08-05T22-01-19Z
172.16.32.101: minio version RELEASE.2021-08-05T22-01-19Z
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {a..h}; do parted -s -a optimal /dev/sd\${i}
mklabel gpt mkpart primary 0% 100%; done"
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {a..h}; do parted -s -- /dev/sd\${i} align-
check optimal 1; done"
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {1..8}; do mkdir -p /mnt/disk\${i}; done"
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {a..h}; do mkfs.xfs -f /dev/sd\${i}\1; done"
[root@sjc02dmz-rhel ~]# clush -g minio "i=1; for j in {a..h}; do mount /dev/sd\${j}\1
/mnt/disk\${(i++)}; done"
[root@sjc02dmz-rhel ~]# clush -g minio "i=0; for j in {a..h}; do UUID=\$(findmnt -fn -o UUID
/dev/sd\${j}\1) && i=\$(expr \${i} + 1); echo "UUID=\$UUID /mnt/disk\${i} xfs defaults 0 0" |
tee -a /etc/fstab; done"

```

1. Confirm the disk mounts.

```

[root@sjc02dmz-rhel ~]# clush -g minio "df -h | grep /mnt"
172.16.32.101: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.101: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.101: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.101: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.101: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.101: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.101: /dev/sg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.101: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8
172.16.32.103: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.103: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.103: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.103: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.103: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.103: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.103: /dev/sg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.103: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8
172.16.32.104: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.104: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.104: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.104: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.104: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5

```

```

172.16.32.104: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.104: /dev/sdg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.104: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8
172.16.32.102: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.102: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.102: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.102: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.102: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.102: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.102: /dev/sdg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.102: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8

```

2. Install MinIO mc client version so you can later create a user and secret to access the cluster.

```

[root@sjc02dmz-rhel ~]# clush -g minio "wget
https://dl.minio.io/client/mc/release/linux-amd64/mc"
[root@sjc02dmz-rhel ~]# clush -g minio "chmod +x mc"
[root@sjc02dmz-rhel ~]# clush -g minio "mv mc /usr/local/bin"
[root@sjc02dmz-rhel ~]# clush -g minio "mc --version"
172.16.32.104: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.101: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.103: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.102: mc version RELEASE.2021-08-05T17-48-22Z

```

3. Start MinIO on all nodes.

```

[root@sjc02dmz-rhel ~]# clush -g minio "groupadd minio"
[root@sjc02dmz-rhel ~]# openssl passwd -crypt -stdin
minio
g3ztzZZ.Y5NxI
[root@sjc02dmz-rhel ~]# clush -g minio "useradd -m -p g3ztzZZ.Y5NxI -g minio minio"
[root@sjc02dmz-rhel ~]# clush -g minio "chown -R minio:minio /mnt/disk*"
[root@sjc02dmz-rhel ~]# for i in {1..4}; do ssh-copy-id minio@minio$i; done
[root@sjc02dmz-rhel ~]# vi minio_start.sh
#!/bin/bash
# Access Key of the server.
export MINIO_ROOT_USER=admin
# Secret key of the server.
export MINIO_ROOT_PASSWORD=nbv12345
# Volume to be used for Minio server.
export MINIO_VOLUMES=http://minio{1...4}/mnt/disk{1...8}
# Execute MinIO.
exec /usr/local/bin/minio server $MINIO_VOLUMES
[root@sjc02dmz-rhel ~]# clush -g minio -c minio_start.sh --dest=/home/minio
[root@sjc02dmz-rhel ~]# clush -g minio -l minio "/home/minio/minio_start.sh > /dev/null
2>&1 &"

```

Note: Production systems should use system to start MinIO.

```
[root@sjc02dmz-rhel ~]# clush -g minio "ps -ef | grep /usr/local/bin/minio"
172.16.32.101: minio      11825      1 1 07:36 ?          00:00:02 /usr/local/bin/minio
server http://minio{1...4}/mnt/disk{1...8}
172.16.32.101: root      12047     12046  0 07:39 ?          00:00:00 bash -c ps -ef | grep
/usr/local/bin/minio
172.16.32.101: root      12057     12047  0 07:39 ?          00:00:00 grep
/usr/local/bin/minio
172.16.32.104: minio      10830      1 1 07:36 ?          00:00:02 /usr/local/bin/minio
server http://minio{1...4}/mnt/disk{1...8}
172.16.32.104: root      11043     11042  0 07:39 ?          00:00:00 bash -c ps -ef | grep
/usr/local/bin/minio
172.16.32.104: root      11053     11043  0 07:39 ?          00:00:00 grep
/usr/local/bin/minio
172.16.32.103: minio      10888      1 1 07:36 ?          00:00:02 /usr/local/bin/minio
server http://minio{1...4}/mnt/disk{1...8}
172.16.32.103: root      11110     11109  0 07:39 ?          00:00:00 bash -c ps -ef | grep
/usr/local/bin/minio
172.16.32.103: root      11120     11110  0 07:39 ?          00:00:00 grep
/usr/local/bin/minio
172.16.32.102: minio      10861      1 1 07:36 ?          00:00:02 /usr/local/bin/minio
server http://minio{1...4}/mnt/disk{1...8}
172.16.32.102: root      11079     11078  0 07:39 ?          00:00:00 bash -c ps -ef | grep
/usr/local/bin/minio
172.16.32.102: root      11089     11079  0 07:39 ?          00:00:00 grep
/usr/local/bin/minio
```

4. Create a MinIO alias. Then create a user with access and a secret key.

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc alias set cisco http://minio1:9000 admin
nbv12345"
Added `cisco` successfully.
Connection to minio1 closed.
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc admin info cisco"
* minio2:9000
  Uptime: 4 minutes
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 8/8 OK

* minio3:9000
  Uptime: 4 minutes
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 8/8 OK
```

```
* minio4:9000
Uptime: 4 minutes
Version: 2021-08-05T17-48-22Z
Network: 4/4 OK
Drives: 8/8 OK
```

```
* minio1:9000
Uptime: 4 minutes
Version: 2021-08-05T17-48-22Z
Network: 4/4 OK
Drives: 8/8 OK
```

32 drives online, 0 drives offline

Connection to minio1 closed.

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc admin user add cisco cisco Cisco123"
```

Added user `cisco` successfully.

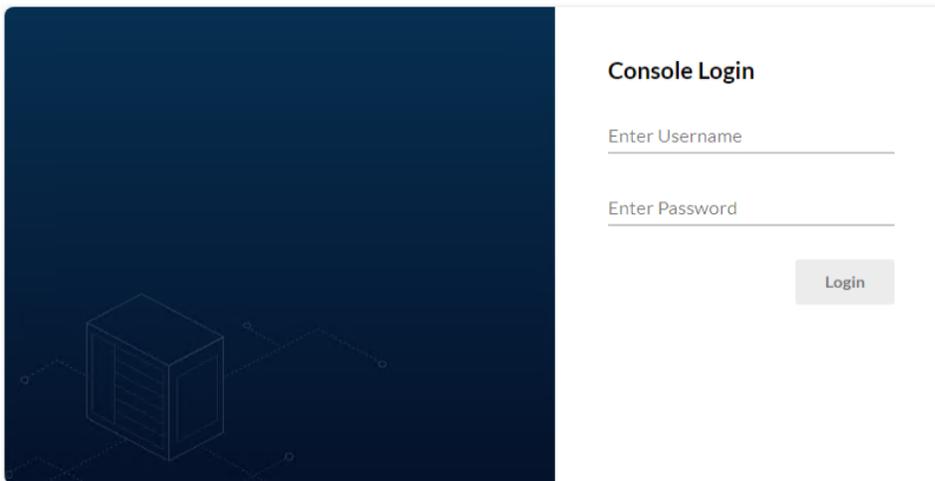
Connection to minio1 closed.

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc admin policy set cisco readwrite
user=cisco"
```

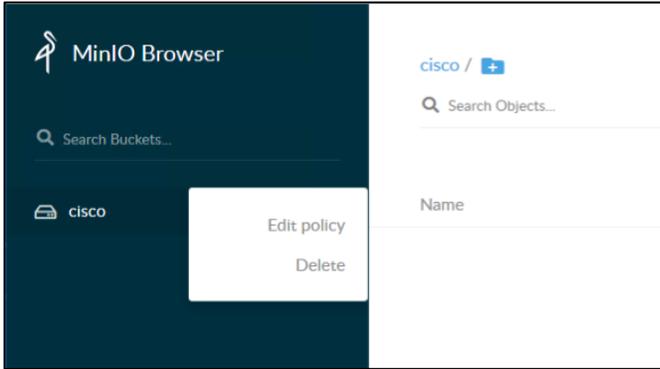
Policy `readwrite` is set on user `cisco`

Connection to minio1 closed.

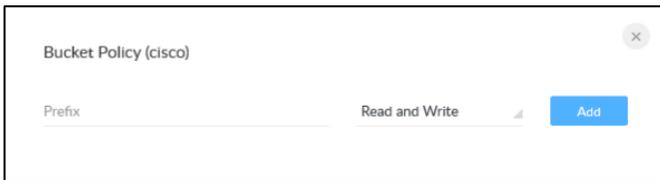
5. Verify the access with the credentials through the MinIO web interface. Go to [http://\[serverip\]:9000](http://[serverip]:9000) from your browser and log in with the configured credentials from step 4.



6. Create a bucket by clicking the bottom red button. Go with the mouse to the bucket and click the three bullet points and edit the policy.



7. Change the policy to Read and Write.



Because MinIO is S3 API compatible, in addition to being able to connect from the mc client, you can connect to MinIO using AWS client tools.

```
[root@sjc02dmz-rhel ~]# aws configure
AWS Access Key ID [*****R7VX]: cisco
AWS Secret Access Key [*****CaLQ]: Cisco123
Default region name [None]:
Default output format [None]:
[root@sjc02dmz-rhel ~]# aws --endpoint-url http://minio1:9000 s3 ls
2021-06-17 01:20:09 cisco
[root@sjc02dmz-rhel ~]# aws --endpoint-url http:// minio1:9000 s3 cp rhel-8.1-x86_64-
dvd.iso s3://cisco
upload: ./rhel-8.2-x86_64-dvd.iso to s3://cisco/rhel-8.2-x86_64-dvd.iso
[root@sjc02dmz-rhel ~]# aws --endpoint-url http:// minio1:9000 s3 ls s3://cisco
2021-05-18 03:39:11 7851737088 rhel-8.2-x86_64-dvd.iso
```

8. The formal setup of the MinIO cluster is now finished and you can move to the operations part, where you can see various daily operations with Cisco Intersight and Terraform.

Day-2 operations for MinIO with Cisco Intersight and Terraform

As mentioned previously, the main intent of this paper is to show how you can expand a running environment with various hardware parts. A couple of scenarios should show the simplicity of doing day-2 operations with Cisco Intersight or Terraform:

- Add disks.
- Add nodes.
- Add network ports.
- Replace a disk.

The following sections describe each scenario with detailed steps from two angles:

- Day-2 operations with Cisco Intersight and the resulting MinIO changes
- Day-2 operations with Terraform provider for Cisco Intersight and the resulting MinIO changes

Add disks

The MinIO solution was built with four Cisco UCS C240 M5L nodes, each node having eight 10-TB disks connected. In the first step of expanding the cluster, you will add four more disks with a total of twelve 10-TB disks for each node. That gives a total of 160-TB raw capacity in addition to the already connected 320-TB raw capacity.

MinIO uses JBOD disks for the whole cluster configuration. That makes scale-up expansions for a running cluster very easy. Here are the steps to add more disks to a running MinIO cluster:

1. Plug the new disks into each node.
2. Check the order of the disks for a sample node. Disks marked in yellow are the newly inserted disks.

```
[root@sjc02dmz-rhel]# ssh -t root@ minio1 "lsblk"
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	9.1T	0	disk	
└─sda1	8:1	0	9.1T	0	part	/mnt/disk1
sdb	8:16	0	9.1T	0	disk	
└─sdb1	8:17	0	9.1T	0	part	/mnt/disk2
sdc	8:32	0	9.1T	0	disk	
└─sdc1	8:33	0	9.1T	0	part	/mnt/disk3
sdd	8:48	0	9.1T	0	disk	
└─sdd1	8:49	0	9.1T	0	part	/mnt/disk4
sde	8:64	0	9.1T	0	disk	
└─sde1	8:65	0	9.1T	0	part	/mnt/disk5
sdf	8:80	0	9.1T	0	disk	
└─sdf1	8:81	0	9.1T	0	part	/mnt/disk6
sdg	8:96	0	9.1T	0	disk	
└─sdg1	8:97	0	9.1T	0	part	/mnt/disk7
sdh	8:112	0	9.1T	0	disk	
└─sdh1	8:113	0	9.1T	0	part	/mnt/disk8
sdi	8:128	0	9.1T	0	disk	
sdj	8:144	0	9.1T	0	disk	
sdk	8:160	0	9.1T	0	disk	
sd1	8:176	0	9.1T	0	disk	
sdm	8:192	0	893.1G	0	disk	
└─sdm1	8:193	0	1G	0	part	/boot
└─sdm2	8:194	0	869.1G	0	part	
└─minio-root	253:0	0	200G	0	lvm	/
└─minio-swap	253:1	0	4G	0	lvm	[SWAP]
└─minio-tmp	253:2	0	40G	0	lvm	/tmp

```

├─minio-var 253:3 0 615.1G 0 lvm /var
└─minio-home 253:4 0 10G 0 lvm /home
sr0          11:0 1 1024M 0 rom
sr1          11:1 1 7.9G 0 rom
sr2          11:2 1 1024M 0 rom
sr3          11:3 1 1024M 0 rom
sdr          65:16 0 894.3G 0 disk
nvme0n1     259:0 0 2.9T 0 disk
nvme1n1     259:1 0 1.5T 0 disk

```

3. Reboot all four nodes to bring the new disks into the right order.

Note: Please note that there is no guarantee Linux will place the disks in the correct order after boot. You will need to use LABEL or Universally Unique Device Identifier (UUID) in /etc/fstab to enforce the correct order placement.

```
[root@sjc02dmz-rhel ~]# clush -g minio reboot
```

4. Configure the new disks to use for MinIO.

```
[root@sjc02dmz-rhel ~]# for i in {1..4}; do ssh -t root@minio$i "for j in {j..1}; do wipefs -af /dev/sd\$j; done"; done
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {i..1}; do parted -s -a optimal /dev/sd\$i mklabel gpt mkpart primary 0% 100%; done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {i..1}; do parted -s -- /dev/sd\$i align-check optimal 1; done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {9..12}; do mkdir -p /mnt/disk\$i; done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {i..1}; do mkfs.xfs -f /dev/sd\$i\1; done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "i=9; for j in {i..1}; do mount /dev/sd\$j\1 /mnt/disk\$(i++); done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "i=8; for j in {i..1}; do UUID=\$(findmnt -fn -o UUID /dev/sd\${j}\1) && i=\$(expr \$i + 1); echo "UUID=\$UUID /mnt/disk\$i} xfs defaults 0 0" | tee -a /etc/fstab; done"
```

```
[root@sjc02dmz-rhel ~]# clush -g minio "df -h | grep /mnt | nl"
```

```

172.16.32.101:      1  /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.101:      2  /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.101:      3  /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.101:      4  /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.101:      5  /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.101:      6  /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.101:      7  /dev/sdg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.101:      8  /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8
172.16.32.101:      9  /dev/sdi1          9.1T   65G   9.1T   1% /mnt/disk9
172.16.32.101:     10  /dev/sdj1          9.1T   65G   9.1T   1% /mnt/disk10
172.16.32.101:     11  /dev/sdk1          9.1T   65G   9.1T   1% /mnt/disk11
172.16.32.101:     12  /dev/sdl1          9.1T   65G   9.1T   1% /mnt/disk12

```

```

172.16.32.103:    1    /dev/sda1          9.1T   65G   9.1T   1%   /mnt/disk1
172.16.32.103:    2    /dev/sdb1          9.1T   65G   9.1T   1%   /mnt/disk2
172.16.32.103:    3    /dev/sdc1          9.1T   65G   9.1T   1%   /mnt/disk3
172.16.32.103:    4    /dev/sdd1          9.1T   65G   9.1T   1%   /mnt/disk4
172.16.32.103:    5    /dev/sde1          9.1T   65G   9.1T   1%   /mnt/disk5
172.16.32.103:    6    /dev/sdf1          9.1T   65G   9.1T   1%   /mnt/disk6
172.16.32.103:    7    /dev/sdg1          9.1T   65G   9.1T   1%   /mnt/disk7
172.16.32.103:    8    /dev/sdh1          9.1T   65G   9.1T   1%   /mnt/disk8
172.16.32.103:    9    /dev/sdi1          9.1T   65G   9.1T   1%   /mnt/disk9
172.16.32.103:   10    /dev/sdj1          9.1T   65G   9.1T   1%   /mnt/disk10
172.16.32.103:   11    /dev/sdk1          9.1T   65G   9.1T   1%   /mnt/disk11
172.16.32.103:   12    /dev/sdl1          9.1T   65G   9.1T   1%   /mnt/disk12
172.16.32.104:    1    /dev/sda1          9.1T   65G   9.1T   1%   /mnt/disk1
172.16.32.104:    2    /dev/sdb1          9.1T   65G   9.1T   1%   /mnt/disk2
172.16.32.104:    3    /dev/sdc1          9.1T   65G   9.1T   1%   /mnt/disk3
172.16.32.104:    4    /dev/sdd1          9.1T   65G   9.1T   1%   /mnt/disk4
172.16.32.104:    5    /dev/sde1          9.1T   65G   9.1T   1%   /mnt/disk5
172.16.32.104:    6    /dev/sdf1          9.1T   65G   9.1T   1%   /mnt/disk6
172.16.32.104:    7    /dev/sdg1          9.1T   65G   9.1T   1%   /mnt/disk7
172.16.32.104:    8    /dev/sdh1          9.1T   65G   9.1T   1%   /mnt/disk8
172.16.32.104:    9    /dev/sdi1          9.1T   65G   9.1T   1%   /mnt/disk9
172.16.32.104:   10    /dev/sdj1          9.1T   65G   9.1T   1%   /mnt/disk10
172.16.32.104:   11    /dev/sdk1          9.1T   65G   9.1T   1%   /mnt/disk11
172.16.32.104:   12    /dev/sdl1          9.1T   65G   9.1T   1%   /mnt/disk12
172.16.32.102:    1    /dev/sda1          9.1T   65G   9.1T   1%   /mnt/disk1
172.16.32.102:    2    /dev/sdb1          9.1T   65G   9.1T   1%   /mnt/disk2
172.16.32.102:    3    /dev/sdc1          9.1T   65G   9.1T   1%   /mnt/disk3
172.16.32.102:    4    /dev/sdd1          9.1T   65G   9.1T   1%   /mnt/disk4
172.16.32.102:    5    /dev/sde1          9.1T   65G   9.1T   1%   /mnt/disk5
172.16.32.102:    6    /dev/sdf1          9.1T   65G   9.1T   1%   /mnt/disk6
172.16.32.102:    7    /dev/sdg1          9.1T   65G   9.1T   1%   /mnt/disk7
172.16.32.102:    8    /dev/sdh1          9.1T   65G   9.1T   1%   /mnt/disk8
172.16.32.102:    9    /dev/sdi1          9.1T   65G   9.1T   1%   /mnt/disk9
172.16.32.102:   10    /dev/sdj1          9.1T   65G   9.1T   1%   /mnt/disk10
172.16.32.102:   11    /dev/sdk1          9.1T   65G   9.1T   1%   /mnt/disk11
172.16.32.102:   12    /dev/sdl1          9.1T   65G   9.1T   1%   /mnt/disk12

```

5. Integrate the new disks into the MinIO service that you created before.

```

[root@sjc02dmz-rhel ~]# ssh -t root@mini01 "mc admin service stop cisco"
Stopped `cisco` successfully.
Connection to mini01 closed.
[root@sjc02dmz-rhel ~]# ssh -t root@mini01 "mc admin info cisco"

```

```

mc: <ERROR> Unable to get service status. Get "http:// minio1:9000/minio/admin/v3/info":
dial tcp minio1:9000: connect: connection refused.
Connection to minio1 closed.
[root@sjc02dmz-rhel ~]# clush -g minio "for i in {9..12}; do chown -R minio:minio
/mnt/disk\${i}; done"
[root@sjc02dmz-rhel ~]# vi minio_start.sh
#!/bin/bash
# Access Key of the server.
export MINIO_ROOT_USER=admin
# Secret key of the server.
export MINIO_ROOT_PASSWORD=nbv12345
# Volume to be used for Minio server.
export MINIO_VOLUMES=http:// minio{1...4}/mnt/disk{1...8}
export MINIO_VOLUMES1=http:// minio{1...4}/mnt/disk{9...12}
# Execute MinIO.
exec /usr/local/bin/minio server $MINIO_VOLUMES $MINIO_VOLUMES1
[root@sjc02dmz-rhel ~]# clush -g minio -c minio_start.sh --dest=/home/minio
[root@sjc02dmz-rhel ~]# clush -g minio -l minio "/home/minio/minio_start.sh > /dev/null
2>&1 &"
[root@sjc02dmz-rhel ~]# ssh -t root@ minio1 "mc admin info cisco"
* minio2:9000
  Uptime: 7 seconds
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 12/12 OK

* minio3:9000
  Uptime: 7 seconds
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 12/12 OK

* minio4:9000
  Uptime: 7 seconds
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 12/12 OK

* minio1:9000
  Uptime: 7 seconds
  Version: 2021-08-05T17-48-22Z
  Network: 4/4 OK
  Drives: 12/12 OK

```

48 drives online, 0 drives offline

Connection to minio1 closed.

You have now integrated the new disks and increased the disk count from 32 disks to 48 disks.

Add nodes

For the following part, you will differentiate between nodes being added by the Cisco Intersight appliance and nodes being added by Terraform provider for Intersight. Common to both paths is claiming the additional targets and assigning them to the right organization before you start rolling out the profiles and installing the OS. That basic process is not described here.

Add nodes with Cisco Intersight appliance

Adding nodes through the Cisco Intersight appliance is a three-step process:

- Clone the current profile to two new profiles and deploy them to assigned servers.
- Create Operating System images in the software repository.
- Install operating software for each server.

At first, you will start cloning a current profile to two new profiles. Here are the steps to clone the profile:

1. Go to **Profiles**, click the three bullet points on the right side for any profile and select **Clone**.
2. Select the two servers you want to assign the new profiles.

Step 1
General
Select and assign server(s) to a clone or specify the number of clones that you want to assign them to later.

Original UCS Server Profile

Organization: Minio Status: Out of Sync

Name: SP-sjc02dmz-i14-c240m511

Target Platform: UCS Server (Standalone)

Server Assignment

<input type="checkbox"/>	Name	User Label	Health	Model	UCS Domain	Serial Nu...
<input type="checkbox"/>	sjc02dmz-i14-c240m514		Critical	UCSC-C240-M5L		WZP21450ZRG
<input type="checkbox"/>	sjc02dmz-i14-c240m512		Critical	UCSC-C240-M5L		WZP21450ZS8
<input type="checkbox"/>	sjc02dmz-i14-c240m513		Critical	UCSC-C240-M5L		WZP21450ZRQ
<input type="checkbox"/>	sjc02dmz-i14-c240m511		Critical	UCSC-C240-M5L		WZP21460GQA
<input checked="" type="checkbox"/>	sjc02dmz-i14-c240m515		Critical	UCSC-C240-M5L		WZP21460GQ9
<input checked="" type="checkbox"/>	sjc02dmz-i14-c240m516		Critical	UCSC-C240-M5L		WZP21450ZRA

Selected 2 of 6

3. Edit the name of the clone profile and deploy it on the selected servers.

4. Go to Software Repository and click Add Operating System Image in the Operating System Images tab.
5. Select the right **Organization**, the **Protocol**, and the **File Location**.

6. Fill out the details on the next page and click **Add**.

7. Repeat the same steps for the remaining server.
8. In the final step, you need to install the OS on each server. Go to the **Servers** tab on the left, select the three bullets on the right where you want to install the OS, and select **Install Operating System**.
9. Select **Next** and then the image you want to install.
10. Select **Next** and then **Embedded**.
11. Select **Next, Next**, and then again **Next**.
12. Select **Install**.
13. Repeat the same steps for the remaining server.

Two additional servers are now preconfigured with Cisco Intersight for MinIO.

Add nodes with Terraform provider for Cisco Intersight appliance

Adding nodes through Terraform for Cisco Intersight is a four-step process:

- Change the variables.tf file.
- Re-run the policy plan and create the additional profiles.
- Deploy the profiles.
- Install the OS on both servers.

In this example, you will use only two servers for the expansion, but usually a minimum of four servers is required to expand a cluster. Follow these steps:

1. Change the variables.tf file and copy it to all directories with the other Terraform plans.

```
[root@sjc02dmz-i14-terraform create_policy_profile]# vi variables.tf
...
variable "remote-os-image-minio" {
    type = list(string)
    default = ["rhel8.2-minio1.iso", "rhel8.2-minio2.iso", "rhel8.2-minio3.iso", "rhel8.2-
minio4.iso", "rhel8.2-minio5.iso", "rhel8.2-minio6.iso"]
}

variable "remote-os-image-link" {
    type = list(string)
    default = ["http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio1.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio2.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio3.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio4.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio5.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio6.iso"]
}

variable "server_names" {
    default = ["sjc02dmz-i14-c240m511", "sjc02dmz-i14-c240m512", "sjc02dmz-i14-c240m513",
"sjc02dmz-i14-c240m514", "sjc02dmz-i14-c240m515", "sjc02dmz-i14-c240m516"]
}
```

...

```
[root@sjc02dmz-i14-terraform create_policy_profile]# cp variables.tf ../deploy_profile/  
[root@sjc02dmz-i14-terraform create_policy_profile]# cp variables.tf ../install_os/
```

2. Run Terraform to create two new profiles. Because we already used the same Terraform state files, Terraform knows that four profiles are already created, so it creates just another two new profiles.

```
[root@sjc02dmz-i14-terraform create_policy_profile]# terraform apply
```

... -> **We skip the full output as it is very lengthy.**

```
Plan: 2 to add, 12 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_server_profile.minio[5]: Creating...
```

```
intersight_server_profile.minio[4]: Creating...
```

```
intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy: Modifying...  
[id=609b9840612a22d472585c71]
```

```
intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy: Modifications complete  
after 1s [id=609b9840612a22d472585c71]
```

```
intersight_server_profile.minio[5]: Creation complete after 1s  
[id=60be0ca377696e2d3092b01a]
```

```
intersight_server_profile.minio[4]: Creation complete after 1s  
[id=60be0ca377696e2d3092b025]
```

```
intersight_storage_storage_policy.minio-storage-policy: Modifying...  
[id=609b9840656f6e2d30c48f56]
```

```
intersight_adapter_config_policy.minio-adapter-config-policy: Modifying...  
[id=609b9840612a22d472585c9c]
```

```
intersight_networkconfig_policy.minio-network-policy: Modifying...  
[id=609b98416275722d3092fc4e]
```

```
intersight_ntp_policy.minio-ntp-policy: Modifying... [id=609b98406275722d3092fc32]
```

```
intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy: Modifying...  
[id=609b9840612a22d472585c8c]
```

```
intersight_boot_precision_policy.minio-boot-policy: Modifying...  
[id=609b98416275722d3092fc63]
```

```
intersight_adapter_config_policy.minio-adapter-config-policy: Modifications complete  
after 0s [id=609b9840612a22d472585c9c]
```

```
intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy: Modifications  
complete after 0s [id=609b9840612a22d472585c8c]
```

```
intersight_storage_storage_policy.minio-storage-policy: Modifications complete after 1s  
[id=609b9840656f6e2d30c48f56]
```

```
intersight_vnic_eth_if.eth2: Modifying... [id=609b9841612a22d472585cdf]
```

```
intersight_vnic_eth_if.eth1: Modifying... [id=609b9841612a22d472585cf4]
```

```
intersight_vnic_eth_if.eth4: Modifying... [id=609b9841612a22d472585cee]
```

```
intersight_vnic_eth_if.eth3: Modifying... [id=609b9841612a22d472585ce8]
intersight_vnic_eth_if.eth0: Modifying... [id=609b9841612a22d472585cd6]
intersight_networkconfig_policy.minio-network-policy: Modifications complete after 1s
[id=609b98416275722d3092fc4e]
intersight_vnic_eth_if.eth1: Modifications complete after 0s
[id=609b9841612a22d472585cf4]
intersight_vnic_eth_if.eth4: Modifications complete after 0s
[id=609b9841612a22d472585cee]
intersight_vnic_eth_if.eth2: Modifications complete after 0s
[id=609b9841612a22d472585cdf]
intersight_vnic_eth_if.eth0: Modifications complete after 0s
[id=609b9841612a22d472585cd6]
intersight_vnic_eth_if.eth3: Modifications complete after 0s
[id=609b9841612a22d472585ce8]
intersight_boot_precision_policy.minio-boot-policy: Modifications complete after 1s
[id=609b98416275722d3092fc63]
intersight_ntp_policy.minio-ntp-policy: Modifications complete after 2s
[id=609b98406275722d3092fc32]
```

Apply complete! Resources: 2 added, 12 changed, 0 destroyed.

3. Deploy the two new profiles and assign them to the specific servers.

```
[root@sjc02dmz-i14-terraform create_policy_profile]# cd ../deploy_profile/
[root@sjc02dmz-i14-terraform deploy_profile]# terraform apply
... -> We skip the full output as it is very lengthy.
Plan: 2 to add, 4 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_server_profile.minio[4]: Creating...
intersight_server_profile.minio[5]: Creating...
intersight_server_profile.minio[3]: Modifying... [id=609b984077696e2d30dc0d9d]
intersight_server_profile.minio[0]: Modifying... [id=609b984077696e2d30dc0da9]
intersight_server_profile.minio[2]: Modifying... [id=609b984077696e2d30dc0db5]
intersight_server_profile.minio[1]: Modifying... [id=609b984077696e2d30dc0d91]
intersight_server_profile.minio[5]: Creation complete after 1s
[id=60be0ca377696e2d3092b01a]
intersight_server_profile.minio[4]: Creation complete after 1s
[id=60be0ca377696e2d3092b025]
intersight_server_profile.minio[3]: Modifications complete after 1s
[id=609b984077696e2d30dc0d9d]
```

```
intersight_server_profile.minio[0]: Modifications complete after 1s
[id=609b984077696e2d30dc0da9]

intersight_server_profile.minio[2]: Modifications complete after 1s
[id=609b984077696e2d30dc0db5]

intersight_server_profile.minio[1]: Modifications complete after 1s
[id=609b984077696e2d30dc0d91]
```

Apply complete! Resources: 2 added, 4 changed, 0 destroyed.

4. Deploy the two servers to install the RHEL Operating System with Terraform.

```
[root@sjc02dmz-i14-terraform deploy_profile]# cd ../install_os/
[root@sjc02dmz-i14-terraform install_os]# terraform apply
... -> We skip the full output as it is very lengthy.
Plan: 4 to add, 0 to change, 0 to destroy.
```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[5]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[4]: Creating...
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[5]: Creation complete after 1s [id=60be39576567612d302389e2]
intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[4]: Creation complete after 1s [id=60be39576567612d302389ea]
intersight_os_install.minio[4]: Creating...
intersight_os_install.minio[5]: Creating...
intersight_os_install.minio[4]: Creation complete after 0s [id=60be39588286999c5611ab73]
intersight_os_install.minio[5]: Creation complete after 0s [id=60be39588286999c5611ab7d]
```

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

The formal deployment of the new nodes is now finished, and you can move over to the further integration of both nodes into the MinIO cluster.

5. Skip the RHEL OS preparation, starting directly with the MinIO integration:

```
[root@sjc02dmz-rhel ~]# for i in {5,6}; do ssh -t root@minio$i "for j in {a..l}; do
wipefs -af /dev/sd\${j}; done"; done
[root@sjc02dmz-rhel ~]# clush -g minio-new "dnf -y install wget"
[root@sjc02dmz-rhel ~]# clush -g minio-new "wget
https://dl.minio.io/server/minio/release/linux-amd64/minio"
[root@sjc02dmz-rhel ~]# clush -g minio-new "chmod +x minio"
```

```

[root@sjc02dmz-rhel ~]# clush -g minio-new "mv minio /usr/local/bin"
[root@sjc02dmz-rhel ~]# clush -g minio "minio --version"
172.16.32.105: minio version RELEASE.2021-08-05T17-48-22Z
172.16.32.106: minio version RELEASE.2021-08-05T17-48-22Z
172.16.32.101: minio version RELEASE.2021-08-05T17-48-22Z
172.16.32.104: minio version RELEASE.2021-08-05T17-48-22Z
172.16.32.102: minio version RELEASE.2021-08-05T17-48-22Z
172.16.32.103: minio version RELEASE.2021-08-05T17-48-22Z

[root@sjc02dmz-rhel ~]# clush -g minio-new "for i in {a..l}; do parted -s -a optimal /dev/sd\${i} mklabel gpt mkpart primary 0% 100%; done"

[root@sjc02dmz-rhel ~]# clush -g minio-new "for i in {a..l}; do parted -s -- /dev/sd\${i} align-check optimal 1; done"

[root@sjc02dmz-rhel ~]# clush -g minio-new "for i in {1..12}; do mkdir -p /mnt/disk\${i}; done"

[root@sjc02dmz-rhel ~]# clush -g minio-new "for i in {a..l}; do mkfs.xfs -f /dev/sd\${i}\1; done"

[root@sjc02dmz-rhel ~]# clush -g minio-new "i=1; for j in {a..l}; do mount /dev/sd\${j}\1 /mnt/disk\${(i++)}; done"

[root@sjc02dmz-rhel ~]# clush -g minio-new "i=0; for j in {a..l}; do UUID=\$(findmnt -fn -o UUID /dev/sd\${j}\1) && i=\$(expr \${i} + 1); echo "UUID=\$UUID /mnt/disk\${i} xfs defaults 0 0" | tee -a /etc/fstab; done"

```

6. Confirm the disk mounts.

```

[root@sjc02dmz-rhel ~]# clush -g minio-new "df -h | grep /mnt"
172.16.32.106: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.106: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.106: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.106: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.106: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.106: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.106: /dev/sg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.106: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8
172.16.32.106: /dev/sdi1          9.1T   65G   9.1T   1% /mnt/disk9
172.16.32.106: /dev/sdj1          9.1T   65G   9.1T   1% /mnt/disk10
172.16.32.106: /dev/sdk1          9.1T   65G   9.1T   1% /mnt/disk11
172.16.32.106: /dev/sdl1          9.1T   65G   9.1T   1% /mnt/disk12
172.16.32.105: /dev/sda1          9.1T   65G   9.1T   1% /mnt/disk1
172.16.32.105: /dev/sdb1          9.1T   65G   9.1T   1% /mnt/disk2
172.16.32.105: /dev/sdc1          9.1T   65G   9.1T   1% /mnt/disk3
172.16.32.105: /dev/sdd1          9.1T   65G   9.1T   1% /mnt/disk4
172.16.32.105: /dev/sde1          9.1T   65G   9.1T   1% /mnt/disk5
172.16.32.105: /dev/sdf1          9.1T   65G   9.1T   1% /mnt/disk6
172.16.32.105: /dev/sg1          9.1T   65G   9.1T   1% /mnt/disk7
172.16.32.105: /dev/sdh1          9.1T   65G   9.1T   1% /mnt/disk8

```

```

172.16.32.105: /dev/sdi1          9.1T   65G   9.1T   1% /mnt/disk9
172.16.32.105: /dev/sdj1          9.1T   65G   9.1T   1% /mnt/disk10
172.16.32.105: /dev/sdk1          9.1T   65G   9.1T   1% /mnt/disk11
172.16.32.105: /dev/sdl1          9.1T   65G   9.1T   1% /mnt/disk12

```

7. Install the latest MinIO mc client version on all hosts.

```

[root@sjc02dmz-rhel ~]# clush -g minio-new "wget
https://dl.minio.io/client/mc/release/linux-amd64/mc"
[root@sjc02dmz-rhel ~]# clush -g minio-new "chmod +x mc"
[root@sjc02dmz-rhel ~]# clush -g minio-new "mv mc /usr/local/bin"
[root@sjc02dmz-rhel ~]# clush -g minio "mc --version"
172.16.32.101: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.102: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.104: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.103: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.105: mc version RELEASE.2021-08-05T17-48-22Z
172.16.32.106: mc version RELEASE.2021-08-05T17-48-22Z

```

8. Create a user and group on the new nodes, edit the MinIO start file, and start the cluster again.

```

[root@sjc02dmz-rhel ~]# clush -g minio-new "groupadd minio"
[root@sjc02dmz-rhel ~]# openssl passwd -crypt -stdin
minio
g3ztzZZ.Y5NxI
[root@sjc02dmz-rhel ~]# clush -g minio-new "useradd -m -p g3ztzZZ.Y5NxI -g minio minio"
[root@sjc02dmz-rhel ~]# clush -g minio-new "chown -R minio:minio /mnt/disk*"
[root@sjc02dmz-rhel ~]# for i in {5,6}; do ssh-copy-id minio@minio$i; done
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc admin service stop cisco"
Stopped `cisco` successfully.
Connection to minio1 closed.
[root@sjc02dmz-rhel ~]# vi minio_start.sh
#!/bin/bash
# Access Key of the server.
export MINIO_ROOT_USER=admin
# Secret key of the server.
export MINIO_ROOT_PASSWORD=nbv12345
# Volume to be used for Minio server.
export MINIO_VOLUMES=http://minio{1...4}/mnt/disk{1...8}
export MINIO_VOLUMES1=http://minio {1...4}/mnt/disk{9...12}
export MINIO_VOLUMES2=http://minio {5...6}/mnt/disk{1...12}
# Execute MinIO.
exec /usr/local/bin/minio server $MINIO_VOLUMES $MINIO_VOLUMES1 $MINIO_VOLUMES2
[root@sjc02dmz-rhel ~]# clush -g minio -c minio_start.sh --dest=/home/minio
[root@sjc02dmz-rhel ~]# clush -g minio -l minio "/home/minio/minio_start.sh > /dev/null
2>&1 &"

```

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio "mc admin info cisco"
```

- minio2:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

- minio3:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

- minio4:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

- minio5:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

- minio6:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

- minio1:9000
Uptime: 33 seconds
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK

72 drives online, 0 drives offline

Connection to minio1 closed.

Add network ports

In certain situations, it makes sense to upgrade the network by adding more connections to each host. One of the reasons to add more disks is to get both more capacity and more throughput. In our example, you have upgraded the network connections from 25 Gbps to each Nexus N93180YC-FX to 50 Gbps by adding another 25-Gbps connection (refer to Figure 4).

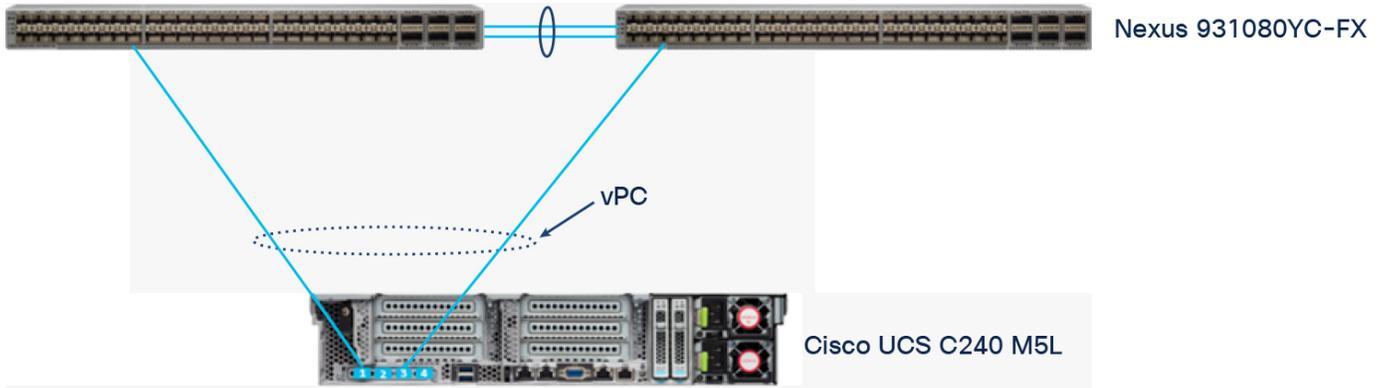


Figure 4.
Single connection to each Nexus switch

Because it is possible to have a default port-channel on the Cisco UCS VIC 1400 Series adapters on the first two ports and the last two ports, you just need to build a static port-channel on each Nexus switch with both connections (refer to Figure 5).

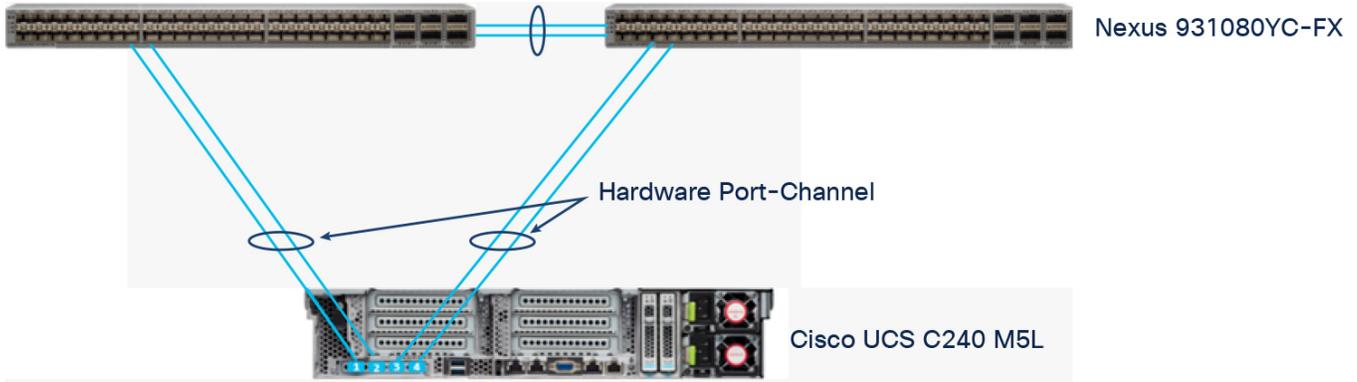


Figure 5.
Hardware port-channel with two connections to each Nexus Switch

After adding the additional connections, you do not need to change the configuration on the Linux site when using bonding or teaming of the interfaces. The vNICs automatically apply the second line to each Nexus switch.

Replace a disk in a storage node

Sometimes a disk fails in the environment, and you need to replace the disk in the MinIO cluster. In the current setup with MinIO you need to do only a few steps in the Cisco Intersight appliance because you are replacing a JBOD disk. As soon as you have replaced the disk, you have just a few steps to prepare the disk for the MinIO cluster.

At first, check whether the disk has an Unconfigured Good or JBOD state and if needed, change it to JBOD.

1. Click the specific server with the replaced disk and check whether all front disks are shown in green.
2. Then click the Inventory tab and go to Storage Controller -> Controller MRAID (RAID).
3. Click Physical Drives and check the state of the replaced disk. In this case the disk has an Unconfigured Good state.

Name	Disk Firmware Version	Size (MiB)	Model	Serial	Vendor	Protocol	Type	Drive State
Disk 1	A3Z4	9536718 MB	HUH721010AL42C0	1SJRAD3Z	HGST	SAS	HDD	JBOD
Disk 2	A3Z4	9536718 MB	HUH721010AL42C0	1SJPMK6Z	HGST	SAS	HDD	JBOD
Disk 3	A3Z4	9536718 MB	HUH721010AL42C0	1SJPLJBZ	HGST	SAS	HDD	JBOD
Disk 4	A3Z4	9536718 MB	HUH721010AL42C0	1SJN05WZ	HGST	SAS	HDD	JBOD
Disk 5	A3Z4	9536718 MB	HUH721010AL42C0	1SJPHB2Z	HGST	SAS	HDD	JBOD
Disk 6	A3Z4	9536718 MB	HUH721010AL42C0	1SJ22MZ	HGST	SAS	HDD	JBOD
Disk 7	A3Z4	9536718 MB	HUH721010AL42C0	1SJM73KZ	HGST	SAS	HDD	JBOD
Disk 8	A3Z4	9536718 MB	HUH721010AL42C0	1SJNJWZ	HGST	SAS	HDD	JBOD
Disk 9	A3Z4	9536718 MB	HUH721010AL42C0	1SJR4D6Z	HGST	SAS	HDD	JBOD
Disk 10	A3Z4	9536718 MB	HUH721010AL42C0	1SJLMJSZ	HGST	SAS	HDD	JBOD
Disk 11	A3Z4	9536718 MB	HUH721010AL42C0	1SJNEPDZ	HGST	SAS	HDD	JBOD
Disk 12	A3Z4	9536718 MB	HUH721010AL42C0	7JKTE1DC	HGST	SAS	HDD	Unconfigured Good
Disk 13	GXT51F3Q	914573 MB	SAMSUNG MZ7LM96...	S3LHNXLJ701235	ATA	SATA	SSD	Online
Disk 14	GXT51F3Q	914573 MB	SAMSUNG MZ7LM96...	S3LHNXLJ701253	ATA	SATA	SSD	Online

4. Now click the three dots on the right for disk 12 and select -> Set State.
5. Select JBOD and click Set.
6. The disk is now set to JBOD and you can start configuring the disk for MinIO.

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio1 "mc admin info cisco"
```

```
* minio2:9000
  Uptime: 10 minutes
  Version: 2021-08-05T17-48-22Z
  Network: 6/6 OK
  Drives: 12/12 OK

* minio3:9000
  Uptime: 10 minutes
  Version: 2021-08-05T17-48-22Z
  Network: 6/6 OK
  Drives: 12/12 OK

* minio4:9000
  Uptime: 10 minutes
  Version: 2021-08-05T17-48-22Z
  Network: 6/6 OK
  Drives: 12/12 OK

* minio5:9000
```

```
Uptime: 10 minutes
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK
```

```
* minio6:9000
Uptime: 10 minutes
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 11/12 OK
```

```
* minio1:9000
Uptime: 10 minutes
Version: 2021-08-05T17-48-22Z
Network: 6/6 OK
Drives: 12/12 OK
```

71 drives online, 1 drive offline

7. Now find the newly added disk on the specific node and verify it with lsblk.

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio6 "for i in {a..l}; do findmnt -fn -o UUID /dev/sd\${i}\1; done"
```

```
b87dc9e0-7a91-4d18-b6c0-379bbee0c595
87e6e0f6-49a2-4f23-9ee1-4b0b100ef8b9
12dc2cfe-363a-4d0c-8157-5ec83c62a24a
abe0f5ad-bdbd-4e36-b0a4-50853910e65c
```

```
43192b9a-0458-43b8-a1b5-3eecf4e9830e
d04c757c-16d6-4d25-aa94-8b825414cec1
e5ccda6d-8d80-4688-ad90-d18b3d86bfe8
ce1f173f-4380-489a-968c-db99d6261ed8
d0467c82-9d9b-4327-a68d-6aeb040aeeb9
2b105d85-6f89-4284-99cc-1d2f841099d8
85c68914-1c9f-4b78-b290-7f70f4bc465b
```

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio6 lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	9.1T	0	disk	
└─sda1	8:1	0	9.1T	0	part	/mnt/disk1
sdb	8:16	0	9.1T	0	disk	
└─sdb1	8:17	0	9.1T	0	part	/mnt/disk2
sdc	8:32	0	9.1T	0	disk	
└─sdc1	8:33	0	9.1T	0	part	/mnt/disk3
sdd	8:48	0	9.1T	0	disk	

```

└─sdd1      8:49  0  9.1T  0 part /mnt/disk4
sdf         8:80  0  9.1T  0 disk
└─sdf1     8:81  0  9.1T  0 part /mnt/disk6
sdg         8:96  0  9.1T  0 disk
└─sdg1     8:97  0  9.1T  0 part /mnt/disk7
sdh         8:112 0  9.1T  0 disk
└─sdh1     8:113 0  9.1T  0 part /mnt/disk8
sdi         8:128 0  9.1T  0 disk
└─sdi1     8:129 0  9.1T  0 part /mnt/disk9
sdj         8:144 0  9.1T  0 disk
└─sdj1     8:145 0  9.1T  0 part /mnt/disk10
sdk         8:160 0  9.1T  0 disk
└─sdk1     8:161 0  9.1T  0 part /mnt/disk11
sdl         8:176 0  9.1T  0 disk
└─sdl1     8:177 0  9.1T  0 part /mnt/disk12
sdm         8:192 0 893.1G 0 disk
└─sdm1     8:193 0   1G  0 part /boot
└─sdm2     8:194 0 869.1G 0 part
    └─minio-root 253:0  0  200G 0 lvm /
    └─minio-swap 253:1  0   4G  0 lvm [SWAP]
    └─minio-tmp  253:2  0  40G  0 lvm /tmp
    └─minio-var  253:3  0 615.1G 0 lvm /var
    └─minio-home 253:4  0  10G  0 lvm /home
sr0         11:0  1 1024M 0 rom
sr1         11:1  1  8.9G 0 rom
sr2         11:2  1 1024M 0 rom
sr3         11:3  1 1024M 0 rom
sdr         65:16 0 894.3G 0 disk
sds         65:32 0  9.1T  0 disk
nvme0n1     259:0  0  2.9T  0 disk
nvme1n1     259:1  0  1.5T  0 disk

```

- Disk `/dev/sde` is missing and disk `/dev/sds` was added by the OS. Delete the appropriate UUID entry of the missing disk in `/etc/fstab` and reboot the system to correct the disk entries and verify. The newly added disk is now `/dev/sdl`.

```

[root@sjc02dmz-rhel ~]# ssh -t root@172.16.32.106 lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0   9.1T  0 disk
└─sda1               8:1    0   9.1T  0 part /mnt/disk1
sdb                  8:16   0   9.1T  0 disk
└─sdb1              8:17   0   9.1T  0 part /mnt/disk2
sdc                  8:32   0   9.1T  0 disk
└─sdc1              8:33   0   9.1T  0 part /mnt/disk3

```

```

sdd                8:48    0    9.1T    0 disk
└─sdd1             8:49    0    9.1T    0 part /mnt/disk4
sde                8:64    0    9.1T    0 disk
└─sde1            8:65    0    9.1T    0 part /mnt/disk6
sdf                8:80    0    9.1T    0 disk
└─sdf1            8:81    0    9.1T    0 part /mnt/disk7
sdg                8:96    0    9.1T    0 disk
└─sdg1            8:97    0    9.1T    0 part /mnt/disk8
sdh                8:112   0    9.1T    0 disk
└─sdh1            8:113   0    9.1T    0 part /mnt/disk9
sdi                8:128   0    9.1T    0 disk
└─sdi1            8:129   0    9.1T    0 part /mnt/disk10
sdj                8:144   0    9.1T    0 disk
└─sdj1            8:145   0    9.1T    0 part /mnt/disk11
sdk                8:160   0    9.1T    0 disk
└─sdk1            8:161   0    9.1T    0 part /mnt/disk12
sd1                8:176   0    9.1T    0 disk
sdm                8:192   0 893.1G  0 disk
├─sdm1             8:193   0     1G    0 part /boot
└─sdm2             8:194   0 869.1G  0 part
    ├─minio-root   253:0    0   200G  0 lvm  /
    ├─minio-swap   253:1    0     4G    0 lvm  [SWAP]
    ├─minio-tmp    253:2    0    40G    0 lvm  /tmp
    ├─minio-var    253:3    0 615.1G  0 lvm  /var
    └─minio-home   253:4    0    10G    0 lvm  /home
sr0                11:0    1   1024M  0 rom
sr1                11:1    1    8.9G   0 rom
sr2                11:2    1   1024M  0 rom
sr3                11:3    1   1024M  0 rom
sdr                65:16   0 894.3G  0 disk
nvme1n1            259:0    0    1.5T   0 disk
nvme0n1            259:1    0    2.9T   0 disk

```

9. Now prepare the new disk for MinIO.

```

[root@sjc02dmz-rhel ~]# ssh -t minio6 "parted -s -a optimal /dev/sd1 mklabel gpt mkpart
primary 0% 100%"
[root@sjc02dmz-rhel ~]# ssh -t minio6 "parted -s -- /dev/sd1 align-check optimal 1"
[root@sjc02dmz-rhel ~]# ssh -t minio6 "mkfs.xfs -f /dev/sd11"
[root@sjc02dmz-rhel ~]# ssh -t minio6 "mount /dev/sd11 /mnt/disk5"
[root@sjc02dmz-rhel ~]# ssh -t minio6 "UUID=$(findmnt -fn -o UUID /dev/sd11) && echo
\"UUID=\$UUID /mnt/disk5 xfs defaults 0 0\" | tee -a /etc/fstab"
[root@sjc02dmz-rhel ~]# ssh minio@172.16.32.106 "/home/minio/minio_start.sh > /dev/null
2>&1 &"

```

```
[root@sjc02dmz-rhel ~]# ssh -t root@minio "mc admin info cisco"
```

```
* minio2:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
* minio3:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
* minio4:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
* minio5:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
* minio6:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
* minio1:9000
```

```
Uptime: 2 hours
```

```
Version: 2021-08-05T17-48-22Z
```

```
Network: 6/6 OK
```

```
Drives: 12/12 OK
```

```
72 drives online, 0 drives offline
```

```
Connection to minio1 closed.
```

The replacement of the failed disk was successful and the MinIO cluster is active again.

Appendix

Terraform MOID variables.tf file

```
# Server and Organization names
variable "server_names" {
  type = list
}

variable "organization_name" {}
```

```
variable "catalog_name" {}
```

Terraform MOID main.tf file

```
# Intersight Provider Information
terraform {
  required_providers {
    intersight = {
      source = "CiscoDevNet/intersight"
      version = "1.0.12"
    }
  }
}

data "intersight_compute_physical_summary" "server_moid" {
  name = var.server_names[count.index]
  count = length(var.server_names)
}

output "server_moids" {
  value = data.intersight_compute_physical_summary.server_moid.*.results.0.moid
}

data "intersight_organization_organization" "organization_moid" {
  name = var.organization_name
}

output "organization_moid" {
  value = data.intersight_organization_organization.organization_moid.results[0].moid
}

data "intersight_softwarerepository_catalog" "catalog_moid" {
  name = var.catalog_name
}
```

```
}
```

```
output "catalog_moid" {  
  value = data.intersight_softwarerepository_catalog.catalog_moid.results[2].moid  
}
```

Terraform main variables.tf file

```
//Define all the basic variables here
```

```
variable "api_private_key" {  
  default = "/root/terraform-intersight-sds/intersight.pem"  
}
```

```
variable "api_key_id" {  
  default = "5e5fb2b17564612d3028b5b4/5e5fbd137564612d3028bcc4/5fa1a9107564612d3007f934"  
}
```

```
variable "api_endpoint" {  
  default = "https://sjc02dmz-intersight.sjc02dmz.net"  
}
```

```
variable "management_vlan" {  
  default = 300  
}
```

```
variable "client_vlan" {  
  default = 301  
}
```

```
variable "storage_vlan" {  
  default = 302  
}
```

```
variable "remote-server" {  
  default = "sjc02dmz-il4-terraform.sjc02dmz.net"  
}
```

```
variable "remote-share" {  
  default = "/images"  
}
```

```
variable "remote-os-image-minio" {  
  type = list(string)
```

```

    default = ["rhel8.2-minio1.iso", "rhel8.2-minio2.iso", "rhel8.2-minio3.iso", "rhel8.2-
minio4.iso", "rhel8.2-minio5.iso", "rhel8.2-minio6.iso"]
}

variable "remote-os-image-link" {
    type = list(string)

    default = ["http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio1.iso",
"http://sjc02dmz-i14-terraform.sjc02dmz.net/images/rhel8.2-minio2.iso", "http://sjc02dmz-i14-
terraform.sjc02dmz.net/images/rhel8.2-minio3.iso", "http://sjc02dmz-i14-
terraform.sjc02dmz.net/images/rhel8.2-minio4.iso", "http://sjc02dmz-i14-
terraform.sjc02dmz.net/images/rhel8.2-minio5.iso", "http://sjc02dmz-i14-
terraform.sjc02dmz.net/images/rhel8.2-minio5.iso"]
}

variable "remote-protocol" {
    default = "softwarerepository.HttpServer"
}

variable "server_names" {
    default = ["sjc02dmz-i14-c240m511", "sjc02dmz-i14-c240m512", "sjc02dmz-i14-c240m513",
"sjc02dmz-i14-c240m514", "sjc02dmz-i14-c240m515", "sjc02dmz-i14-c240m516"]
}

variable "organization_name" {
    default = "Minio"
}

variable "server_profile_action" {
    default = "No-op"
}

variable "catalog_name" {
    default = "appliance-user-catalog"
}

```

Terraform main.tf file for creating policies and profiles

```

# Intersight Provider Information
terraform {
    required_providers {
        intersight = {
            source = "CiscoDevNet/intersight"
            version = "1.0.12"
        }
    }
}

```

```

}

provider "intersight" {
  apikey      = var.api_key_id
  secretkey   = var.api_private_key
  endpoint    = var.api_endpoint
}

module "intersight-moids" {
  source      = "../../terraform-intersight-moids"
  server_names = var.server_names
  organization_name = var.organization_name
  catalog_name      = var.catalog_name
}

resource "intersight_server_profile" "minio" {
  count = length(var.server_names)
  name = "SP-${var.server_names[count.index]}"
  organization {
    object_type = "organization.Organization"
    moid        = module.intersight-moids.organization_moid
  }
  assigned_server {
    moid        = module.intersight-moids.server_moids[count.index]
    object_type = "compute.RackUnit"
  }
  action = var.server_profile_action
}

resource "intersight_networkconfig_policy" "minio-network-policy" {
  name          = "minio-network-policy"
  description   = "DNS Configuration Policy for CIMC"
  organization {
    object_type = "organization.Organization"
    moid        = module.intersight-moids.organization_moid
  }
  preferred_ipv4dns_server = "192.168.10.51"
  alternate_ipv4dns_server = ""
  dynamic "profiles" {
    for_each = intersight_server_profile.minio
    content {
      moid = profiles.value["moid"]
    }
  }
}

```

```

        object_type = "server.Profile"
    }
}

resource "intersight_adapter_config_policy" "minio-adapter-config-policy" {
    name          = "minio-adapter-config-policy"
    description   = "Adapter Configuration Policy for Minio"
    organization {
        object_type = "organization.Organization"
        moid       = module.intersight-moids.organization_moid
    }
    settings {
        slot_id = "MLOM"
        dce_interface_settings {
            fec_mode = "c174"
            interface_id = "0"
        }
        dce_interface_settings {
            fec_mode = "c174"
            interface_id = "1"
        }
        dce_interface_settings {
            fec_mode = "c174"
            interface_id = "2"
        }
        dce_interface_settings {
            fec_mode = "c174"
            interface_id = "3"
        }
        eth_settings {
            lldp_enabled = true
        }
        fc_settings {
            fip_enabled = false
        }
        port_channel_settings {
            enabled = "true"
        }
    }
    dynamic "profiles" {
        for_each = intersight_server_profile.minio
    }
}

```

```

    content {
        moid = profiles.value["moid"]
        object_type = "server.Profile"
    }
}

resource "intersight_vnic_eth_adapter_policy" "minio-ethernet-adapter-policy" {
    name = "minio-ethernet-adapter-policy"
    description = "Ethernet Adapter Policy for Minio"
    rss_settings = true
    organization {
        object_type = "organization.Organization"
        moid = module.intersight-moids.organization_moid
    }
    vxlan_settings {
        object_type = "vnic.VxlanSettings"
        enabled = false
    }
    nvgre_settings {
        enabled = false
        object_type = "vnic.NvgreSettings"
    }
    arfs_settings {
        object_type = "vnic.ArfsSettings"
        enabled = true
    }
    roce_settings {
        object_type = "vnic.RoceSettings"
        enabled = false
    }
    interrupt_settings {
        coalescing_time = 125
        coalescing_type = "MIN"
        nr_count          = 11
        mode              = "MSIx"
        object_type = "vnic.EthInterruptSettings"
    }
    completion_queue_settings {
        object_type = "vnic.CompletionQueueSettings"
        nr_count    = 9
        ring_size   = 1
    }
}

```

```

}
rx_queue_settings {
    object_type = "vnic.EthRxQueueSettings"
    nr_count    = 8
    ring_size   = 4096
}
tx_queue_settings {
    object_type = "vnic.EthTxQueueSettings"
    nr_count    = 1
    ring_size   = 4096
}
tcp_offload_settings {
    object_type = "vnic.TcpOffloadSettings"
    large_receive = true
    large_send    = true
    rx_checksum   = true
    tx_checksum   = true
}
}

resource "intersight_vnic_eth_network_policy" "minio-mgt-network" {
    name = "minio-mgt-network"
    description = "Mgt Network for Minio"
    organization {
        object_type = "organization.Organization"
        moid = module.intersight-moids.organization_moid
    }
    vlan_settings {
        object_type = "vnic.VlanSettings"
        default_vlan = var.management_vlan
        mode         = "TRUNK"
    }
}

resource "intersight_vnic_eth_network_policy" "minio-client-network" {
    name = "minio-client-network"
    description = "Client Network for Minio"
    organization {
        object_type = "organization.Organization"
        moid = module.intersight-moids.organization_moid
    }
    vlan_settings {

```

```
    object_type = "vnic.VlanSettings"
    default_vlan = var.client_vlan
    mode        = "TRUNK"
  }
}
```

```
resource "intersight_vnic_eth_network_policy" "minio-storage-network" {
  name = "minio-storage-network"
  description = "Storage Network for Minio"
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
  vlan_settings {
    object_type = "vnic.VlanSettings"
    default_vlan = var.storage_vlan
    mode        = "TRUNK"
  }
}
```

```
resource "intersight_vnic_eth_qos_policy" "minio-ethernet-qos-9000-policy" {
  name          = "minio-ethernet-qos-9000-policy"
  description = "Ethernet quality of service for Minio"
  mtu           = 9000
  rate_limit   = 0
  cos          = 0
  trust_host_cos = false
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
}
```

```
resource "intersight_vnic_eth_qos_policy" "minio-ethernet-qos-1500-policy" {
  name          = "minio-ethernet-qos-1500-policy"
  description = "Ethernet quality of service for Minio"
  mtu           = 1500
  rate_limit   = 0
  cos          = 0
  trust_host_cos = false
  organization {
    object_type = "organization.Organization"
  }
}
```

```

    moid = module.intersight-moids.organization_moid
  }
}

resource "intersight_vnic_lan_connectivity_policy" "minio-lan-connectivity-policy" {
  name = "minio-lan-connectivity-policy"
  description = "LAN Connectivity Policy for Minio"
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
  dynamic "profiles" {
    for_each = intersight_server_profile.minio
    content {
      moid = profiles.value["moid"]
      object_type = "server.Profile"
    }
  }
}

resource "intersight_vnic_eth_if" "eth0" {
  name = "eth0"
  order = 0
  placement {
    object_type = "vnic.PlacementSettings"
    id = "MLOM"
    pci_link = 0
    uplink = 0
  }
  cdn {
    nr_source = "vnic"
  }
  vmq_settings {
    enabled = false
    num_interrupts = 1
    num_vmqs = 1
  }
  lan_connectivity_policy {
    moid = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
    object_type = "vnic.LanConnectivityPolicy"
  }
  eth_network_policy {

```

```

    moid = intersight_vnic_eth_network_policy.minio-mgt-network.id
}
eth_adapter_policy {
    moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
}
eth_qos_policy {
    moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-1500-policy.id
}
}

resource "intersight_vnic_eth_if" "eth1" {
    name = "eth1"
    order = 1
    placement {
        object_type = "vnic.PlacementSettings"
        id = "MLOM"
        pci_link = 0
        uplink = 0
    }
    cdn {
        nr_source = "vnic"
    }
    vmq_settings {
        enabled = false
        num_interrupts = 1
        num_vmqs = 1
    }
    lan_connectivity_policy {
        moid = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
        object_type = "vnic.LanConnectivityPolicy"
    }
    eth_network_policy {
        moid = intersight_vnic_eth_network_policy.minio-client-network.id
    }
    eth_adapter_policy {
        moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
    }
    eth_qos_policy {
        moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy.id
    }
}
}

```

```

resource "intersight_vnic_eth_if" "eth2" {
  name = "eth2"
  order = 2
  placement {
    object_type = "vnic.PlacementSettings"
    id = "MLOM"
    pci_link = 0
    uplink = 1
  }
  cdn {
    nr_source = "vnic"
  }
  vmq_settings {
    enabled = false
    num_interrupts = 1
    num_vmqs = 1
  }
  lan_connectivity_policy {
    moid = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
    object_type = "vnic.LanConnectivityPolicy"
  }
  eth_network_policy {
    moid = intersight_vnic_eth_network_policy.minio-client-network.id
  }
  eth_adapter_policy {
    moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
  }
  eth_qos_policy {
    moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy.id
  }
}

```

```

resource "intersight_vnic_eth_if" "eth3" {
  name = "eth3"
  order = 3
  placement {
    object_type = "vnic.PlacementSettings"
    id = "MLOM"
    pci_link = 0
    uplink = 0
  }
  cdn {

```

```

    nr_source = "vnic"
}
vmq_settings {
    enabled = false
        num_interrupts = 1
    num_vmqs = 1
}
lan_connectivity_policy {
    moid      = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
    object_type = "vnic.LanConnectivityPolicy"
}
eth_network_policy {
    moid = intersight_vnic_eth_network_policy.minio-storage-network.id
}
eth_adapter_policy {
    moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
}
eth_qos_policy {
    moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy.id
}
}

resource "intersight_vnic_eth_if" "eth4" {
    name = "eth4"
    order = 4
    placement {
        object_type = "vnic.PlacementSettings"
        id      = "MLOM"
        pci_link = 0
        uplink = 1
    }
    cdn {
        nr_source = "vnic"
    }
    vmq_settings {
        enabled = false
            num_interrupts = 1
        num_vmqs = 1
    }
    lan_connectivity_policy {
        moid      = intersight_vnic_lan_connectivity_policy.minio-lan-connectivity-policy.id
        object_type = "vnic.LanConnectivityPolicy"
    }
}

```

```
}
eth_network_policy {
  moid = intersight_vnic_eth_network_policy.minio-storage-network.id
}
eth_adapter_policy {
  moid = intersight_vnic_eth_adapter_policy.minio-ethernet-adapter-policy.id
}
eth_qos_policy {
  moid = intersight_vnic_eth_qos_policy.minio-ethernet-qos-9000-policy.id
}
}
```

```
resource "intersight_ntp_policy" "minio-ntp-policy" {
  name      = "minio-ntp-policy"
  description = "NTP Policy for Minio"
  enabled = true
  ntp_servers = [
    "173.38.201.115"
  ]
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
  dynamic "profiles" {
    for_each = intersight_server_profile.minio
    content {
      moid = profiles.value["moid"]
      object_type = "server.Profile"
    }
  }
}
```

```
resource "intersight_storage_drive_group" "minio-drive-group-boot-c240" {
  name      = "minio-drive-group-boot-c240"
  description = "Drive Group Boot for Minio"
  raid_level = "Raid1"
  use_jbods = true
  manual_drive_group {
    span_groups {
      slots = "13-14"
    }
  }
}
```

```

virtual_drives {
  object_type      = "storage.VirtualDriveConfig"
  boot_drive       = false
  expand_to_available = true
  name             = "Boot_VD"
  virtual_drive_policy {
    drive_cache    = "Default"
    access_policy  = "ReadWrite"
    read_policy    = "ReadAhead"
    write_policy   = "WriteBackGoodBbu"
  }
}
storage_policy {
  moid = intersight_storage_storage_policy.minio-storage-policy.moid
}
}

```

```

resource "intersight_storage_storage_policy" "minio-storage-policy" {
  name                = "minio-storage-policy"
  description         = "Storage Policy for Minio"
  use_jbod_for_vd_creation = true
  unused_disks_state = "Jbod"
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
  dynamic "profiles" {
    for_each = intersight_server_profile.minio
    content {
      moid = profiles.value["moid"]
      object_type = "server.Profile"
    }
  }
}
}

```

```

resource "intersight_boot_precision_policy" "minio-boot-policy" {
  name                = "minio-boot-policy"
  description         = "Boot Policy for Minio"
  configured_boot_mode = "Legacy"
  enforce_uefi_secure_boot = false
  organization {
    object_type = "organization.Organization"
  }
}

```

```

    moid = module.intersight-moids.organization_moid
  }
  boot_devices {
    enabled      = true
    name         = "disk"
    object_type  = "boot.LocalDisk"
    additional_properties = jsonencode({
      Slot = "MRAID"
    })
  }
  boot_devices {
    enabled      = true
    name         = "vmedia"
    object_type  = "boot.VirtualMedia"
    additional_properties = jsonencode({
      Subtype = "cimc-mapped-dvd"
    })
  }
  dynamic "profiles" {
    for_each = intersight_server_profile.minio
    content {
      moid = profiles.value["moid"]
      object_type = "server.Profile"
    }
  }
}

```

Terraform main.tf file for deploying profiles

```

# Intersight Provider Information
terraform {
  required_providers {
    intersight = {
      source = "CiscoDevNet/intersight"
      version = "1.0.12"
    }
  }
}

provider "intersight" {
  apikey      = var.api_key_id
  secretkey   = var.api_private_key
  endpoint    = var.api_endpoint
}

```

```

module "intersight-moids" {
  source          = "../../terraform-intersight-moids"
  server_names    = var.server_names
  organization_name = var.organization_name
  catalog_name    = var.catalog_name
}

resource "intersight_server_profile" "minio" {
  count = length(var.server_names)
  name = "SP-${var.server_names[count.index]}"
  organization {
    object_type = "organization.Organization"
    moid        = module.intersight-moids.organization_moid
  }
  assigned_server {
    moid        = module.intersight-moids.server_moids[count.index]
    object_type = "compute.RackUnit"
  }
  action = "Deploy"
}

```

Terraform main.tf file for installing Operating System

```

# Intersight Provider Information
terraform {
  required_providers {
    intersight = {
      source = "CiscoDevNet/intersight"
      version = "1.0.12"
    }
  }
}

provider "intersight" {
  apikey      = var.api_key_id
  secretkey   = var.api_private_key
  endpoint    = var.api_endpoint
}

module "intersight-moids" {
  source          = "../../terraform-intersight-moids"
  server_names    = var.server_names
  organization_name = var.organization_name
}

```

```
catalog_name = var.catalog_name
}
```

```
resource "intersight_softwarerepository_operating_system_file" "rhel-custom-iso-with-
kickstart-minio" {
  count = length(var.server_names)
  nr_version = "Red Hat Enterprise Linux 8.2"
  description = "RHEL 8.2 installer ISO with embedded kickstart MinIO"
  name = "ISO-${var.server_names[count.index]}"
  nr_source {
    additional_properties = jsonencode({
      LocationLink = var.remote-os-image-link[count.index]
    })
    object_type = var.remote-protocol
  }
  vendor = "Red Hat"
  catalog {
    moid = module.intersight-moids.catalog_moid
  }
}
```

```
resource "intersight_os_install" "minio" {
  count = length(var.server_names)
  name = "minio-os-${var.server_names[count.index]}"
  server {
    object_type = "compute.RackUnit"
    moid = module.intersight-moids.server_moids[count.index]
  }
  image {
    object_type = "softwarerepository.OperatingSystemFile"
    moid = intersight_softwarerepository_operating_system_file.rhel-custom-iso-with-kickstart-
minio[count.index].moid
  }
  answers {
    nr_source = "Embedded"
  }
  description = "OS install"
  install_method = "vMedia"
  organization {
    object_type = "organization.Organization"
    moid = module.intersight-moids.organization_moid
  }
}
```

}

Conclusion

Day-2 operations for scale-out storage are getting more and more important. On-premises IT operations teams need to manage compute, network, and storage infrastructure. The Cisco Intersight appliance and Terraform provider for Cisco Intersight can greatly improve such operations. Customers get all the benefits of software-as-a-service (SaaS) delivery and full lifecycle management of distributed infrastructure and workloads across data centers, remote sites, branch offices, and edge environments. This service empowers customers to analyze, update, fix, and automate the environment in ways that were not previously possible. As a result, organizations can achieve significant Total Cost of Ownership (TCO) savings and deliver applications faster in support of new business initiatives.

In combination with MinIO, customers get an optimum of cloud-scale storage together with an easy-to-manage environment. The simplicity of Day-2 operations with Cisco Intersight and Terraform provider for Cisco Intersight makes MinIO even better to run and operate.

For more information

For additional information, refer to the following:

- Cisco Intersight data sheet: <https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/intersight/datasheet-c78-739433.html#FlexibleDeploymentOptions>
- Terraform provider for Cisco Intersight: <https://registry.terraform.io/providers/CiscoDevNet/intersight/latest>
- MinIO: <https://min.io/>
- Github Repository: https://github.com/ucs-compute-solutions/Intersight_Terraform_MinIO

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 <https://www.cisco.com/go/offices>.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)