How Alaska Airlines Automates Cisco UCS
Introduction

Alaska Airlines business units regularly evaluate whether application workloads should be placed into the public cloud or in on-premises data centers. Alaska Airlines evaluates application placement based on a number of factors, including cost, capacity, and automation tooling. Offering an alternative to public cloud is key to delivering a successful on-premises data center. In today’s era of the software-defined data center, engineers must learn to build infrastructure that is as dynamic and flexible as the public cloud.

Alaska Airlines, with contributions from Cisco and VMware, has developed a solution to auto scale on-premises virtualized infrastructure in response to the requirement for IT to keep pace with the speed of business. One difficult aspect of on-premises data center architecture is that you often design for the highest spike of required capacity and build to that benchmark. By instead designing to scale on demand, we can even out the peaks and drive down the total cost of infrastructure ownership.

The focus of the solution presented here is the accessibility, speed, and reliability of VMware ESXi host provisioning, using the Cisco Unified Computing System™ (Cisco UCS®) and the VMware hypervisor. The solution documented here aims to fully describe all the components used in this method of infrastructure deployment. The core hardware being deployed consists of Cisco® UCS blade servers, managed by Cisco UCS Manager. The software components include Cisco UCS Central Software, Cisco UCS Director, VMware vSphere, and VMware vRealize Operations Manager (vROps).

The key to this solution is the software-defined nature of Cisco UCS hardware. All the elements of the hardware can be defined based on policies and pools; it grants exceptional flexibility for automation and scalability. The solution offered here can be implemented at small and large enterprises alike.

All components of the solution are described; however, a general knowledge of both VMware and Cisco products is assumed. This document does not describe how to set up vSphere or how to deploy Cisco UCS Central. Please see the appropriate product documentation for more general guidelines.

The document is targeted toward engineers and architects at medium and large enterprise organizations who have at least an intermediate level of vSphere and Cisco UCS experience.

Business objectives

Alaska Airlines, like many businesses and industries, has seasonal demands such as holidays and summer travel, and these generate peak loads on our infrastructure. During our annual Cyber Monday sales events, we often increase our web server load by a factor of 5. Alaska Airlines IT is now pivoting to support these peak economic periods with elastic capacity on demand. Although elements of the website can and have been moved into the public cloud, a significant number of the applications that support the site remain in on-premises data centers. As a result, IT needs to provide the same auto scaling capabilities of public cloud for on-premises workload locations.

In addition to scaling on demand and doing so at the speed of business, our business stakeholders consuming on-premises IT services expect a level of reliability equal to or better than that found in public cloud services. The reliability requirement is often measured by how often Service-Level Agreement (SLA) goals are met and the consistency of application performance.
Solution overview

We will demonstrate how to monitor VMware vSphere cluster capacity and generate an automated workflow stream using vROps and Cisco UCS Director. This workflow allocates a new Cisco UCS blade server, provisions it using service profiles, installs and configures ESXi, and finally provisions it into the proper vCenter cluster.

The solution overview in Figure 1 illustrates the four initial steps in the solution workflow. Starting with a capacity trigger from vROps, the solution creates the entire process of taking that trigger, deciphering where the capacity is required, and kicking off the workflow in Director to build up the required compute capacity in that target vSphere cluster. It then deploys and configures the ESXi host and validates the compliance with the associated host profile before placing it online in the cluster and accepting live workload. All of this happens without any manual intervention.

Since this process is machine repeatable, scaling out or scaling in to meet dynamic business requirements is possible at high speeds. When scaling out, once the host is online, VMware Distributed Resource Scheduler (DRS) takes over and distributes the workload as needed. Because the model supports peak periods, it is equally important to scale down host resources at other times and return servers to a resource pool.

Properly built automation provides perfect deployments every time with zero configuration drift and eliminates the possibility of human error. Shifting to this automated process has resulted in a reduction in service-impacting events for Alaska. Automation is solving the consistency problem by leveraging Cisco UCS service profile templates, host profiles, storage presentation, and drivers/patch levels, all of which use the exact same build process. In addition, application performance is kept consistent by keeping capacity properly sized (based on the monitoring trigger).

While achieving the business objectives through automation was the main driver for this solution, there are other benefits to IT as well:

- **Recovery times**: In the event of hardware-based faults, the stateless nature of the hosts allows for triggering the automation workflow and allocating a new server from a resource pool.
- **Patching and upgrades**: Because the hosts are stateless in this solution, if a configuration change is needed, it can be done simply by modifying the image or the host profile and rebooting a host.
- **Security and auditing**: Software-driven configuration is easier to audit and control, making periodic compliance checking more manageable.
Growth and lifecycle management

We first implemented this solution to support a new data center project where we needed to build out a large number of hosts in a short amount of time across multiple data centers. We then adopted this solution in our legacy data centers to support a hardware lifecycle refresh.

Not only does the solution allow for elastic growth based on monitoring triggers with no human intervention but it also provides a framework that allows manual builds at scale with a single command. The old scripted deployments could be done at a rate of two hosts per hour, under perfect conditions. Trying to speed up the process resulted in more deployment failures. In contrast, using the automation solution, hosts could be built at a rate of 50 per hour, and that number could be significantly higher if the hardware were required and available.

In the next section, we take the three main components of the solution, review how they are built, and highlight the considerations involved in the configuration.

Technical overview

vROps monitoring

The first component in our auto scale solution is the monitoring of the vSphere environment. To determine if a vSphere cluster needs to increase or decrease capacity we rely on capacity monitoring capabilities in vROps. Thresholds were created in vROps based on CPU and memory metrics within a cluster for both scale-up and scale-down events. When a threshold is crossed, a defined vROps alert generates a Representational State Transfer (REST) API call to the API gateway indicating the environment affected.

When configuring a REST notification trigger three components need to be defined:

- Symptom definitions
- Alert definitions
- Notification settings

Symptom definitions

For the auto scale automation, vROps will be looking for a specific condition to be met before issuing a scale-up REST notification. Since we are concerned with the host resources available to virtual machines, this can be broken down into two different symptom definitions: CPU capacity remaining and memory capacity remaining (Figure 2).

Figure 2. vROps symptom definitions
At Alaska Airlines, symptom conditions are defined to have been met when the appropriate resource has less than 15 percent capacity remaining in the cluster. To configure two system definitions in vROps, use these steps:

a. Navigate in vROps to Alerts ➔ Symptom Definitions and click the plus icon to add a new symptom definition. Configure a new alert symptom with Cluster Compute Resource as the base object.

b. Navigate to CPU ➔ Capacity Usage % and from the Info drop-down select Critical when metric is less than or equal to 15 percent, as shown in Figure 3. Adjust wait cycles as desired. The wait cycle is how often the polling cycle occurs in vCenter. The default in vROps is 5 minutes per wait cycle.

Figure 3. vROps trigger for CPU capacity remaining

![Cluster Compute Resource: CPU/Capacity Remaining (%)](image)

- **Static Threshold:**
  - ucxd-autoscale-cpu-capacity: Critical when metric is less than or equal to 15
  
- **Advanced:**
  - Wait Cycle: 1
  - Cancel Cycle: 1
  - Evaluate on instanced metric:


c. Navigate in vROps to Alerts ➔ Symptom Definitions, and click the plus icon to add a new symptom definition. Configure a new alert symptom with Cluster Compute Resource as the base object.

d. Navigate to Memory ➔ Capacity Usage % and from the Info drop-down select Critical when metric is less than or equal to 15 percent, as shown in Figure 4. Adjust wait cycles as desired. The wait cycle is how often the polling cycle occurs in vCenter. The default in vROps is 5 minutes per wait cycle.

Figure 4. vROps trigger for memory capacity remaining

![Cluster Compute Resource: Memory/Capacity Remaining (%)](image)

- **Static Threshold:**
  - ucxd-autoscale-memory-capacity: Critical when metric is less than or equal to 15
  
- **Advanced:**
  - Wait Cycle: 1
  - Cancel Cycle: 1
  - Evaluate on instanced metric:
Alert definitions
After the symptoms are defined, an alert definition is needed so that vROps knows where to look for the symptoms to be present and how to categorize them.

1. Create a new alert definition and add in the symptom definitions created previously.
2. Add the base object as vCenter Adapter → Cluster Compute Resource.
3. Set the following values, as shown in Figure 5.
   - **Base Object Type**: Cluster Compute Resource
   - **Impact**: Health
   - **Criticality**: Critical
   - **Alert Type**: Virtualization/Hypervisor: Capacity
   - **Base Objects Exhibits**: Any
4. Add both symptom types to the symptom definitions as match = Any.

As shown in Figure 5, we have defined the ‘ucsd-autoscale’ alert to be active if either the ‘ucsd-autoscale-cpu-capacity’ symptom or the ‘ucsd-autoscale-memory-capacity’ symptom is present. Once the alert has been triggered on a host cluster by one of these conditions it will remain active until both are below the configured threshold (15 percent capacity remaining).
**Figure 5.** vROps alert definitions

<table>
<thead>
<tr>
<th>Alert Definition Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name:</strong> ucsp-autoscale</td>
</tr>
<tr>
<td><strong>Base Object Type:</strong> Cluster Compute Resource</td>
</tr>
<tr>
<td><strong>Impact:</strong> Health</td>
</tr>
<tr>
<td><strong>Criticality:</strong> Critical</td>
</tr>
<tr>
<td><strong>Alert Type:</strong> Virtualization/Hypervisor: Capacity</td>
</tr>
</tbody>
</table>

**Symptoms**

- **Self-Cluster Compute Resource**
  
  **This symptom set is true when:**
  - Base object exhibits **Any** of the following symptoms:
    
    - 1. ucsp-autoscale-cpu-capacity
    - 2. ucsp-autoscale-memory-capacity

  **Drag another symptom here to add more symptoms.**
**Notification settings**

Before configuring the notification for our REST API call in vROps, an outbound REST notification plug-in instance needs to be created pointing to our API gateway (Figure 6).

1. To configure, go to the Administration section and select Management of vROps.
2. Configure the following settings:
   - **Plugin Type**: Rest Notification Plugin
   - **Instance Name**: The name of the instance as it will appear in vROps
   - **URL**: The full URL to the API gateway (see the API gateway section for more details)
   - **Username**: Username for Flask service running on API gateway
   - **Password**: Password for Flask service running on API gateway
   - **Content type**: Set to ‘application/json’ since the API gateway will be parsing for JSON

Figure 6. vROps alert outbound settings
After the outbound REST plug-in instance has been configured and tested in vROps, the new alert notification rule can be created.

1. In the Alerts section of vROps select Notification Settings.
2. Configure the following settings, as shown in Figure 7:
   - **Name**: The name of the notification rule as it will appear in vROps
   - **Method**: Rest Notification Plugin and the newly created outbound REST instance selected from the drop-down
   - **Filtering criteria**:
     - **Scope**: Object Type and Cluster Compute Resource
     - **Notification Trigger**: Alert Definition and the auto scale alert definition created previously

![vROps alert notification settings](image)

**API gateway**

The API gateway is a Red Hat Enterprise Linux 7.4 virtual machine running the Flask web service. Alaska Airlines is using the Flask service to address limitations in the vROps API plug-in, which cannot generate the data payload needed by Cisco UCS Director to initiate the auto scale workflow.

As shown in the three workflow tasks detailed in Figure 8 below, the Flask service receives a REST API call from vROps and then parses the JSON data to determine which vCenter environment and vSphere cluster has reached a capacity threshold. Based on the JSON data passed from vROps, the Flask service can determine several parameters based on predetermined naming standards. A query is made to Infoblox to determine the next available hostname iteration. Once that is done, a Python program is called to validate data from vROps, construct a JSON payload, and then initiate a REST API call to Director. That call starts the auto scale workflow.
As an example, Alaska Airlines uses the following naming standards, which allows for easy parsing of data:

- **vSphere cluster naming standard:**
  
  \{(SiteName)}-{(EnvironmentType)}-{(ClusterType)}
  
  Example: DC1-PRD-GEN

- **The Service Profile Template naming standard** is similar:
  
  LS-SPT-{(SiteName)}-{(EnvironmentType)}-{(ClusterType)}
  
  Example: LS-SPT-DC1-PRD-GEN

**Figure 8.** Data transform process

Several values in the vROps payload are used to generate the correct Director workflow parameters:

- **Alert Status:** When the alert threshold is initially reached, the “Alert Status” value is set to "Active". This is the only alert status we need to perform an action on; we need to ignore the other possible values (such as “Cancelled”).

- **Resource Name:** This is the name of the VMware host cluster that has reached the critical threshold. We will use this resource name to create the payload that is sent to Director to submit a host build workflow.

- **Alert Name:** This is the alert name we are using to trigger an automated host build workflow.

**Example vROps notification**

```plaintext
/*
   'status': 'ACTIVE',
   'startDate': 1522435619340,
   'criticality': 'ALERT_CRITICALITY_LEVEL_CRITICAL',
   'resourceId': 'fffffff-dddc-4ff3-9999-ebfffafff4c1',
   'Messages': '',
   'Type': 'ALERT_TYPE_VIRTUALIZATION_PROBLEM',
   'subType': 'ALERT_SUBTYPE_CAPACITY_PROBLEM',
   'Health': 4,
   'AlertName': 'ucsd-autoscale',
   'resourceName': 'DC1-PRD-GEN'
 */
```
After the API gateway has determined this is a valid scale-up request on the provided VMware host cluster, it executes a Python script to create a JSON payload for Director. The payload consists of the following parameters:

- **Cisco UCS Director workflow name**: The name of the workflow to be submitted (static).
- **Site name**: The data center environment.
- **vCenter**: The name of the vCenter instance the cluster belongs to
- **Cluster type**: The type of workload the cluster performs (general, DMZ, database, etc.)
- **Environment**: The environment type (production, QA, test)
- **VMhost name**: The hostname for the host being built, determined by querying Infoblox
- **Template**: The Cisco UCS Central service profile template to be used (This template has many settings and specifications that are used to build out the host, such as MAC address pool, hardware type, World Wide Name [WWN] address pools, etc.)

Figure 9 shows the steps used to create the Director payload.

**Figure 9.** Flask payload generation

![Figure 9. Flask payload generation](image)

**Note:** The Alaska Airlines auto scale solution uses custom Python code for the API gateway, but other programming languages could be used.

The custom Python code contains functions for:

- Basic authentication mechanisms
- Setting up the Flask application to listen on the [http://myapigateway:8080/endpoint/ucsdirector](http://myapigateway:8080/endpoint/ucsdirector) URL for POST requests generated by vROps
- A ‘parse’ function to format the incoming JSON payload into a usable dictionary format
- Checking the validity of the incoming notification based on our specifications
- Executing a custom ‘create_payload’ function that creates the JSON payload required to submit a scale-up workflow request to Director based on the cluster name included in the vROps alert (see the create_payload code below)
- As shown in the code samples below, submit a REST API workflow service request to the Director server, including the JSON payload for the new host build

The following code snippet includes the firing mechanism for the Director service request.
REST API call to Director

```python
# Parse the JSON data submitted by vROps
a = parse(request)

# Check validity and submit UCS Director workflow request
if ((a['AlertName'] == webhookalertname) and (a['status'] == 'ACTIVE')):
    payload = create_payload(a['resourceName'])
    uri = 'https://' + ucsdhost + '/app/api/rest?formatType=json&opName=userAPISubmitWorkflowServiceRequest&opData=
    if test:
        print payload
        return ('UCSD Autoscale Test', 200, None)
    else:
        print payload
        return ucsdcall(uri, 'post', json.dumps(payload, separators=(',', ':')), headers, check=False)
else:
    return ('No Action Required', 200, None)
```

Example JSON for Director payload

```json
/*
    {'param2': -1,
     'param1':
      {'list': [
        {'name': 'input_0_Site5369120',
         'value': 'dc1'},
        {'name': 'input_1_vCenter121332',
         'value': 'vcdclprd'},
        {'name': 'input_10_Cluster5995348',
         'value': 'gen'},
        {'name': 'input_10_Environment8337882',
         'value': 'prd'},
        {'name': 'input_3_Cluster8995855',
         'value': 'dcl-pod1-vcdclprd@dcl0DC1-PRD-GEN'},
        {'name': 'input_7_Host_Name67104',
         'value': 'dclprdesxigen14'},
        {'name': 'input_8_UCSC_Template15226',
         'value': '1;org-root/org-VMware;org-root/org-VMware/ls-spt-dcl-prd-gen'}
      ]},
    'param0': 'Scale_UP'
}*/```
Cisco UCS Director

The third major component of the solution, Cisco UCS Director, is the primary orchestrator and handles the connection between all the elements of the infrastructure. Our solution is built on Cisco UCS Central, multiple vSphere installations, NetApp storage arrays with Brocade switches for Fibre Channel connectivity, Infoblox for Internet Protocol Address Manager (IPAM), and an external Director PowerShell Agent to handle some of the final tasks through vSphere's PowerCLI, as shown in Figure 10.

Figure 10. Cisco UCS Director component connections

Cisco UCS Central, vSphere, and NetApp are all added as managed objects in Cisco UCS Director. The Brocade and Infoblox connections are unmanaged and referenced only through scripts within the workflows. The PowerShell Agent is used to execute vSphere PowerCLI commands, and is a component of Director that can be deployed when configuring the Director appliance. It is called out in Figure 10 because it is an external Windows Server that runs the PowerShell Agent service on behalf of Director.

How you choose to organize your technology stacks into pods within Director is flexible; this solution does not rely on it being built in any specific way. However, the naming of the various components definitely matters, as most of those names are used when deriving values used in the scripts and workflow execution forms. Note that Director (version 6.5) does not allow renaming most elements once they have been added to inventory, so we recommend establishing your naming philosophy in advance to avoid having to delete and reconnect components.

This section focuses on a few specific elements of the Cisco UCS design:

- Naming conventions are critical and they are used in multiple components, including Director, Central, and VMware components. Names are either programmatically derived based on associated elements or are used to derive values used in later steps of the process.
- Using a matrix to derive the values for the various pools will greatly simplify and standardize your deployment. A predefined matrix will also make it easier to grow and adapt your environment for the future.
- Most of the policies defined in Cisco UCS will follow best practice guidelines; however, you need to ensure some specific ones are created, including the following:
  - BIOS policy specific to VMware components and virtualization
  - Boot policy for Preboot Execution Environment (PXE) booting into VMware Auto Deploy services
  - Server pool qualification policy to define how the server pools are allocated
• Similarly, the templates used in this solution are the same templates you’d need for any best practice implementation of Cisco UCS Central; however, there are some specific notes related to a couple of these templates:
  ◦ virtual Network Interface Card (vNIC) and virtual Host Bus Adapter (vHBA) templates
  ◦ Interface placement for service profile templates

Naming conventions
Before getting into the various components that are handled by Cisco UCS Director, it is necessary to first understand the importance of naming conventions. Names drive the level of complexity for automation in many cases. Concise rules for naming allow for simpler derivatives within the code of workflows. The majority of named elements are derived from a simple table of lookup values based on the following examples:

• environment: prd, tst, dev, stg
• datacenter: dc1, dc2, dc3, dc4
• cluster: gen, ora, sql, dmz
• interface: mgt, sto, sec, opn
• hardware: gen, db
• side: a, b

Those definitions may be heavily impacted by the network and storage configurations of the target environment. They are simply examples, which are then used to derive the names for the following components:

• vCenter: vc + [datacenter] + [environment]
• vSphere host profiles: hp + [datacenter] + [cluster] + [iteration]
• ESXi hostname: [datacenter] + [environment] + esxi + [cluster] + [iteration]
• Central service profile templates: spt + [datacenter] + [environment] + [cluster]
• Central vNIC templates: vnt + [datacenter] + [environment] + [interface] + [side]
• Central pools:
  ◦ Universal Unique Identifier (UUID) pool name: uuid + [datacenter] + [environment] + [cluster]
  ◦ MAC pool name: mac + [datacenter] + [environment] + [cluster] + [interface] + [side]
  ◦ World Wide Node Name (WWNN) pool name: wwnn + [datacenter] + [environment] + [cluster]
  ◦ World Wide Port Name (WWPN) pool name: wwpn + [datacenter] + [environment] + [cluster] + [side]
  ◦ KVM pool name: kvm + [datacenter] + [environment] + [cluster]
  ◦ Server pool name: svr + [datacenter] + [environment] + [hardware]
Cisco UCS Central pools

While the solution can be built using Cisco UCS Manager, it is best to centrally manage the configuration in Cisco UCS Central. This allows for greater consistency and easier auditing and management of polices and pools. If a license for Central is not available, this solution can still be accomplished using Manager, but plan for more overhead, additional code, and extra validation steps in your workflows.

As with Cisco UCS Director, there are no special requirements for setting up domain groups or where policies live. You can choose to keep it flat, put everything in one organization, segregate into multiple sub organizations, set up single or multiple domain groups per data center, or use some combination of those depending what makes sense for your environment. Keep in mind that server pools do not span domain groups, so consider that when planning out the structure of your Central deployments.

Pools can be designed using whatever strategy you choose. The key is that by generating them programmatically, you can easily predict what a future host's addressing will be. This method simplifies the code required to create things like storage fabric zoning and identifying the interface to use when detecting when a new host enters a target cluster.

We strongly recommend provisioning pools using UCS PowerTool, to ensure that pool names are created without error. As with Director objects, once created, a pool cannot be renamed; instead it must be deleted and recreated if it needs a different name, which can be problematic if active profiles have already been built from that pool.

For simplicity, most pools contain 254 objects, even though currently 64 is the maximum needed in any given pool because in vSphere 6.5, the maximum number of ESXi hosts in a cluster is 64. If future versions of vSphere allow for more hosts, this design will still accommodate up to 254 hosts without having to deploy additional pools.

Note that the number is 254 instead of 255 because we're intentionally skipping the first iteration of 00, and starting at 01. Ideally this will match whatever your hostname iterations are. Central assigns addresses sequentially from a pool, so hostname01 should have a MAC address that also ends in 01. If your hostname strategy starts at 00, adjust your pools to also start at 00 (which would give you 255 objects instead of 254).

Pools should be derived based on the established naming conventions. For MAC addresses, it is considered best practice to keep the first three positions at the default of 00:25:B5 for Cisco hardware identification. The remainder of the MAC address can be derived based on the matrix in Table 1.

Table 1. MAC address pool matrix

<table>
<thead>
<tr>
<th>Data center</th>
<th>Upper 4th</th>
<th>vCenter</th>
<th>Lower 4th</th>
<th>vcluster</th>
<th>Upper 5th</th>
<th>Interface</th>
<th>Lower 5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc1</td>
<td>1</td>
<td>dev</td>
<td>1</td>
<td>Gen</td>
<td>1</td>
<td>mgt-a</td>
<td>1</td>
</tr>
<tr>
<td>dc2</td>
<td>2</td>
<td>tst</td>
<td>2</td>
<td>Ora</td>
<td>2</td>
<td>mgt-b</td>
<td>2</td>
</tr>
<tr>
<td>dc3</td>
<td>3</td>
<td>prd</td>
<td>3</td>
<td>Sql</td>
<td>3</td>
<td>sto-a</td>
<td>3</td>
</tr>
<tr>
<td>dc4</td>
<td>4</td>
<td></td>
<td></td>
<td>Dmz</td>
<td>4</td>
<td>opn-a</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>opn-b</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sec-a</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sec-b</td>
<td>8</td>
</tr>
</tbody>
</table>
For example, the MAC address pool named mac-dc2devgenmgmt-b would have a pool of 00:25:B5:21:12:01 - 00:25:B5:21:12:FF. (Note that the first address in the pool is 01, rather than 00.)

Similarly, for WWPNs, we recommend keeping the first five positions at the default of 20:00:00:25:B5 for Cisco hardware identification. The remainder of the WWPN address can be derived based on the matrix in Table 2.

Table 2. WWPN pool matrix

<table>
<thead>
<tr>
<th>Data center</th>
<th>Upper 6th</th>
<th>vCenter</th>
<th>Lower 7th</th>
<th>vcluster</th>
<th>Upper 8th</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc1</td>
<td>1</td>
<td>dev</td>
<td>1</td>
<td>Gen</td>
<td>1</td>
</tr>
<tr>
<td>dc2</td>
<td>2</td>
<td>tst</td>
<td>2</td>
<td>Ora</td>
<td>2</td>
</tr>
<tr>
<td>dc3</td>
<td>3</td>
<td>prd</td>
<td>3</td>
<td>Sql</td>
<td>3</td>
</tr>
<tr>
<td>dc4</td>
<td>4</td>
<td></td>
<td></td>
<td>Dmz</td>
<td>4</td>
</tr>
</tbody>
</table>

For example, the WWPN address pool named wwpn-dc4prddmz-a would have a pool of 20:00:00:25:B5:43:4A:01 - 20:00:00:25:B5:43:4A:FF, and wwpn-dc4prddmz-b would have a pool of 20:00:00:25:B5:43:4B:01 - 20:00:00:25:B5:43:4B:FF.

Server pools are the other big pool to configure. Each Service Profile Template (SPT) has a server pool associated with it, and each Cisco UCS domain will have one server pool per hardware model. So if you have a Cisco UCS domain with two different blade configurations (for example, one model that is used for databases and another model for everything else), you will want two server pools defined. The pool is defined using a server pool qualification policy, as explained below.

The other pools used by this solution are:

- **UUID**: We recommend defining this using the same philosophy as the MAC and WWPN pools highlighted above.
- **WWNN**: Again, use the same philosophy as above.
- **IP**: If you plan to enable KVM access on your blades, you will need to define IP pools. Doing so is optional but recommended, and it can be done in any way that makes sense for your environment.

Cisco UCS Central policies

Multiple policies are required to complete this solution, but how you define your policies is mostly up to you. You will need a policy for each of the following:

- **BIOS**: The VMware recommended policy should work fine, but that may vary depending on your hardware and environment specifics. This policy configures things such as C-state and virtualization VT flags for your CPUs.
- **Boot**: A simple policy required to PXE boot the hosts. In our environment, we use diskless blades, so the policy includes only two items: LAN boot and SD card. We explicitly define the vNIC name in the LAN boot section, which means a policy is required for each vSphere cluster you wish to automate.
- **Ethernet adapter**: We use the default global VMware policy as configured out of the box, with no special modifications.
- **Fibre Channel adapter**: We also use the default global VMware Fibre Channel adapter policy with no special modifications here.
- **Host firmware package**: Although it is not explicitly required for this solution, we strongly recommend complementing VMware’s stateless cache deployment with a separate host firmware package policy for each vSphere cluster. That way, you can easily align firmware updates with host upgrades (drivers and ESXi versions).

- **Server pool qualification**: This policy is used to define the server pools, for example, blades with 6-core CPUs versus blades with 12-core CPUs. There is a one-to-one relationship between server pool qualification and server pools.

Other policies may be required in your environment but they are not relevant to this discussion.

**Cisco UCS Central templates**

Once all the pools and policies are created, the required templates can be built. Three types of templates must be configured:

- vNIC templates
- vHBA templates
- Service profile templates

**vNIC templates**

These are built as updating templates, and we create redundant templates for each traffic type in each vSphere cluster: one for A-side traffic and one for B-side. Note that we do not leverage Cisco UCS redundancy or failover. If we select those options, it will obfuscate any link failures or maintenance activities to vSphere, which can cause some unintended results related to path selection and VMware High Availability on an ESXi host. Instead, we want vSphere to detect and handle any potential link down state, so we leave redundancy set to None and failover set to Disabled in the vNIC templates.

Figure 11 shows an example template. Because we have specific business requirements to separate certain traffic, we create multiple sets of interfaces per host. In our case, we build a set for host management traffic that includes vMotion, a set for IP storage traffic for things like NFS and iSCSI presented to guest virtual machines, a set for protected credit card data traffic, and a final set for everything else, which is where most virtual machine traffic is transmitted.

These templates are where we specify which MAC address pools to use and which VLANs are assigned to each set of interfaces. We also define a few specific policies, such as quality of service (not required in this solution), or network control if you want to enable Cisco Discovery Protocol (CDP). The reason these are updating templates is so that adding a VLAN, for example, can be done on the fly, without reapplying the template or requiring a reboot.
Figure 11.  Cisco UCS Central vNIC template
vHBA templates
These templates are similar to the vNIC templates, except that they define the Fibre Channel connectivity. These are also updating templates, with redundancy set to None, and they are mapped to a WWPN pool. Likewise, a set is built for each cluster: one for A-side and one for B-side fabrics. Unlike the vNIC templates, there is only the one set, since we’re maintaining only one pair of Fibre Channel connections. Figure 12 shows an example template.

Figure 12. Cisco UCS Central vHBA template
Service profile templates

Service profile templates are the core design element that is leveraged by Cisco UCS Director (or any automation tool pointed at Central). This template pulls together all the other templates, pools, and policies, and defines how each blade server is configured and deployed. Unlike the other two types of template, these are not updating templates, as that would require the blade to reboot any time a change was made to the template. Any changes made to a service profile template will result in deleting the old profile and spinning up a new one based on the changed template.

The service profile template specifies everything from which server pool to build from to what the boot and BIOS policies are, and which vNIC and vHBA templates to use. The template defines all the things that make up the hardware configuration of the blade. Figure 13 shows an example template.

One key to service profile template design is the interface placement configuration. Because we’re relying on VMware host profiles, the order of the PCI devices is important. Ensure that every service profile is deployed identically by manually specifying the interface placement order. This way, we can be sure that vmnic0, for example, will always be the same uplink across every host using the same host profile. Before we started adding this to our service profile template, we would find that even though the host profile showed compliant the VLANs being presented were sometimes different. This discrepancy occurred because on the Cisco UCS backend, vmnic0 might have been mapped to a different uplink due to different interface placement.

We utilize a policy of “linear ordered” for the virtual slot mapping scheme, using “assigned-only” for each of the four virtual slot selection preferences. We then manually order the interfaces for each A-side and B-side fabric across each of the four vCONs. The step-by-step process for how to do this can be found here.
**Figure 13.** Cisco UCS Central service profile template

<table>
<thead>
<tr>
<th>Template Usage</th>
<th>Settings</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 Associated</td>
<td>User Label</td>
<td></td>
</tr>
<tr>
<td>0 Unassociated</td>
<td>Template Instantiation Mode</td>
<td>Initial</td>
</tr>
<tr>
<td>0 Config Error</td>
<td>Desired Power State Check On Association</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>Compatibility Check On Migration Using Server Pool</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

### ID Pools
- Assigned
  - UUID Pool: unknown
  - In-Band Management IP Pool (IPv4): Not Assigned
  - Out-of-Band Management IP Pool (IPv4): Not Assigned
  - In-Band Management IP Pool (IPv6): Not Assigned
  - ION Pool: Not Assigned

### Connectivity
- Assigned
  - LAN Connectivity (Manual): 8 vNICs, 0 iSCSI vNICs
  - SAN Connectivity (Manual): 2 vHBAs
  - Dynamic Connectivity: Not Assigned
  - Management VLAN: Not Assigned

### Servers
- Assigned
  - Server Pool: mgp.server
  - Server Pool Qualification Policy: mgp.server

### Storage
- Assigned
  - Local Disk Configuration Policy: SD
  - Storage Profile: Not Assigned

### Policies
- Assigned
  - Maintenance: global-default
  - Host Firmware Package: mgp.vmware
  - BIOS: global-vmware
  - Boot: mgp.vmware
  - Host Interface Placement: ESK
  - vMedia: Not Assigned
  - Scrub: global-default
  - Power Control: global-default
  - iDRAC Access: Not Assigned
  - Serial Over LAN: Not Assigned
  - Statistics Threshold: global-default
vCenter and cluster design
vCenter clusters are generally defined by hardware type, network segmentation, or workload type. Auto Deploy and Cisco UCS Director workflows can be configured to identify where to put new systems based on the naming standards.

Example: MSSQL cluster has its own hardware model. Auto Deploy rule sets can be defined by IP/MAC address range and hardware type. This determines the server pool that Cisco UCS Central pulls from when creating the service profile. Director then places the service profile into the appropriate cluster, and Auto Deploy applies the host profile for the targeted vCenter cluster.

We recommend implementing a fully automated disaster recovery system in all clusters to take advantage of the elasticity and hands-off deployment/tear down of ESXi hosts.

How you choose to organize your clusters is up to you. The following key components are required for the solution we’re implementing:

- A Trivial File Transfer Protocol (TFTP) server is required for each vCenter deployment.
- Likewise, an image profile repository is required for each vCenter deployment. This is managed by the Auto Deploy services.
- Custom deploy rules are required for each host profile in our environments.
- To successfully deploy stateless cache hosts, a number of specific configurations must be included in (and sometimes excluded from) each host profile.

TFTP
Each vCenter will require a TFTP server to handle the Auto Deploy architecture. VMware provides out-of-the-box TFTP services via the vCenter Server Appliance (VCSA) package, or you can simply install your preferred TFTP software on another server that both vCenter and the ESXi hosts have network access to. Each vCenter has a unique set of Auto Deploy files that need to be injected into the TFTP server; those are found under the root of vCenter ➔ Configuration ➔ Auto Deploy ➔ Download TFTP boot Zip hyperlink. This deployment still requires the Flash web client; it cannot yet be managed through the HTML5 interface.

Auto Deploy management
vCenter will require the Image Builder and Auto Deploy services to be enabled and started to provide stateless boot services to the hardware layer and to manage the Auto Deploy image profiles and deploy rules.

- **Image profiles**: Configure appropriate image profiles for Cisco hardware and remove unnecessary vSphere installation bundles. Custom drivers and future driver updates are also implemented here.
- **Deploy rules**: Rules should have a one-to-one relationship with host profiles. Multiple host profiles can exist for a single cluster (separated by hardware type, if necessary), but each host profile needs to have an Auto Deploy rule. Configure deploy rules with the following information:
  - Rule name, preferably referencing the cluster name using the rule
  - MAC address range, derived from the MAC pool used for the target cluster
  - Hardware identifier of blade type (for example, UCS-B200-M4)
  - Auto Deploy image profile to use
  - VMware host profile for the cluster and hardware type
  - Target cluster to deploy the new host into
Be sure to activate all rules after configurations are completed.

**Stateless cache**
Stateless cache allows ESXi to be loaded into memory and saves a cached copy to local drives or SD cards. The stateless configuration means that each time a host is rebooted it is essentially a new host booting off the Auto Deploy service, and there is no concern about configuration artifacts that may build up over time in a stateful configuration. Also note that because of the cached mode, if connectivity or other issues prevent a new image being deployed, it can still boot from the cached copy, greatly increasing resilience in the case of a wider environment failure, for example if a power outage knocked out power in the data center.

**Host profiles**
Ensure the Auto Deploy stateless cache settings are configured under Advanced Configuration Settings ➔ System Image Cache Configuration ➔ System Image Cache Configuration. Host profiles contain settings such as storage LUN or NFS mounts, vSwitch associations (either distributed virtual switch or virtual standard switch with its port groups), and most importantly, host port group settings (where each VMkernel adapter is defined). These VMkernel adapter settings are referenced when Cisco UCS Director creates the answer file during deployment.

With stateless or stateless cache systems, nothing is stored on the local disk that persists through reboots. Core-dump and ESXi logs must be redirected to SAN storage and/or syslog targets and network core-dump services in vCenter to retain diagnostic data that persists through reboots. Zdump crash files are redirected to the vCenter Network Coredump service when the VMware vSphere ESXi Dump Collector service is enabled and started.

Under the Host profile section of vCenter, edit the host profile as follows:

1. Ensure the name matches the standards, and put in a description, if desired, on the first page.
2. The second page is where all the settings are stored. For this Cisco UCS environment make the following configurations:
   1. Uncheck the entire CIM Indication Subscriptions tree from enforcement if present.
      Expand the Advanced Configuration Settings, subsequent Advanced Options, and Advanced Configuration options trees.
      Uncheck ScratchConfig.ConfiguredScratchLocation. Because each host is unique we cannot enforce this option.

      Set up Secure Shell (SSH) and Direct Console User Interface (DCUI) timeouts, and syslog configurations:
      - (Optional) UserVars.ESXiShellTimeOut = 300
      - (Optional) UserVars.ESXiShellInteractiveTimeOut = 300
      - (Optional) All syslog.loggers.x.size = 10240
      - Syslog.global.logDir = [VMFS datastore] ESXiLogs
      - Syslog.global.logDirUnique = Enabled (Check the box)

      (Optional) Expand Power System Configuration and ensure Power System is set to CPU policy High Performance.
Expand System Image Cache Configuration and ensure the setting is Enable Stateless Caching to USB disk on the host.

Under General System Settings, ensure Coredump Partition Settings is disabled.

Expand the Network Configuration tree and then open the Host Port Group tree. Under each VMkernel adapter type (there should be four in the current model), expand the tree for each VMkernel adapter:

- Select the IP address settings for vMotion1, vMotion2, Management Network and Storage VMkernel adapters.
- Under the IPv4 drop-down, select User specified IPv4 address to be used while applying the configuration.

Expand netStack instance ➔ defaultTcpipStack ➔ DNS Configuration, and ensure the DNS IP addresses are updated (DNS servers):

- Domain name is <domain>.
- Search domain is <domain>.
- Host name is set to User Specified host name to be used while applying the configuration.

Click IP route configuration and ensure the default gateway is correct.

- Expand IP route configuration and ensure the IP routes are set up properly for the environment. (If a host profile alarms on incorrect number or type of IPv4 routes, this section is what needs to be checked for validity.)

Click Network Coredump settings, enable core dump, specify vmk0 as the Network Interface Card (NIC) to use, and specify <IP address of vCenter> over port 6500.

In the Security and Services section, highlight the Security Configuration tree. That will expose an Administrator Password section. Open the drop-down menu, choose Configure a Fixed Administrator Password, and set a root password as desired.

Expand the Storage Configuration tree, and uncheck Software FCoE Configuration, Virtual SAN Configuration, and iSCSI Initiator Configuration.

Expand the Pluggable Storage Architecture (PSA) Configuration tree, and then expand the PSA Device Setting tree.

- Ensure that each data store has an entry with the appropriate settings in all three locations under PSA: Device Sharing, Device Setting, Device Configuration.

Expand the Native Multi-Pathing (NMP) tree and then expand Storage Array Type Plug-in (SATP) configuration.

Expand the SATP default PSP configuration tree and highlight the VMW_SATP_ALUA policy to see the settings. Modify the *PSP name to VMW_PSP_RR. (This sets all new storage LUNs to Round Robin by default, versus Most Recently Used, which is the policy default set by ESXi.)

Expand the PSP and SATP configuration for NMP devices tree. Then expand the PSP configuration. (Ensure both sub-trees have the data store disk IDs with appropriate settings.)

3. Complete the host profile edit and finish the wizard.
Hardware choices

To simplify host profile management, we strongly recommend having only one hardware type per cluster. Clusters can be mixed with different hardware types/brands/models, but each hardware type then needs a unique host profile. That also means those host profiles have to be carefully maintained to ensure 100 percent consistency between them.

Example: Two hardware types in a single cluster need two Auto Deploy rules referencing two different host profiles. They will contain nearly the same configuration with only the hardware model types and PCI bus enumeration varying between the hardware types.

For accurate capacity planning and management, each server pool in Cisco UCS Central should consist of only one hardware capacity model. For example, you may have one pool of servers that have a high core count, but lower speed CPUs, and another pool of servers that have high-speed CPUs but a lower core count. Having a separate pool for each type allows a service profile template to directly associate with a hardware type and the use case that platform is dedicated to.

Operational workflow—Cisco UCS Director

The Cisco UCS Director workflow is initiated once it receives a REST API call from the API gateway service. The workflow has a number of both built-in and custom tasks that execute serially. A few tasks could be configured to execute simultaneously to save time; however, the majority of the tasks have prerequisites, such as inputs that are generated from a previous task. Much of the processing time of the workflow is spent waiting for the host to come online.

The Director workflow progresses through each task shown in Table 3.

Table 3. Logical Steps of the UCS Director Workflow Tasks

<table>
<thead>
<tr>
<th>Step</th>
<th>Workflow task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get host information</td>
<td>Get ESXi management and storage networks plus SAN arrays</td>
</tr>
<tr>
<td>2</td>
<td>Infoblox ESXi Management IP</td>
<td>Call Infoblox to assign ESXi Management IPs and update DNS</td>
</tr>
<tr>
<td>3</td>
<td>Infoblox ESXi Storage IP</td>
<td>Call Infoblox to assign ESXi Storage IPs and update DNS</td>
</tr>
<tr>
<td>4</td>
<td>Assign vMotion IP</td>
<td>Self-assign vMotion IP addresses</td>
</tr>
<tr>
<td>5</td>
<td>SAN zone creation</td>
<td>Create zones on the Brocade for the new host</td>
</tr>
<tr>
<td>6</td>
<td>Check Cisco UCS pool inventory</td>
<td>Call Central for server pool spare</td>
</tr>
<tr>
<td>7</td>
<td>Create service profile</td>
<td>Call Central for server profile creation, and then deploy</td>
</tr>
<tr>
<td>8</td>
<td>Waiting for server power on</td>
<td>Pauses workflow as Cisco UCS configures and boots server</td>
</tr>
<tr>
<td>9</td>
<td>PXE boot and VMware Auto Deploy</td>
<td>This step happens outside Director workflow as server boots</td>
</tr>
<tr>
<td>10</td>
<td>Check host online</td>
<td>Monitor vCenter for host registration</td>
</tr>
<tr>
<td>11</td>
<td>Customize ESXi host</td>
<td>Use PowerShell to insert variable data into answer file</td>
</tr>
<tr>
<td>12</td>
<td>Host configuration compliance validation</td>
<td>vCenter validates compliance and exits maintenance mode</td>
</tr>
<tr>
<td>13</td>
<td>Verify configuration errors</td>
<td>JavaScript that confirms steps 11–12 are error free</td>
</tr>
<tr>
<td>14</td>
<td>Get server KVM IP</td>
<td>Query Central for pool-assigned KVM IP</td>
</tr>
<tr>
<td>15</td>
<td>Infoblox KVM IP</td>
<td>Call Infoblox to reserve KVM IP with DNS entry</td>
</tr>
<tr>
<td>16</td>
<td>Director Service Request (SR) documentation</td>
<td>Populate SR with workflow data and generate email</td>
</tr>
</tbody>
</table>
Step 1: Get host information

Overview: Look up networks for storage and management, and look up storage arrays

Director task type: Custom

Description: The first task in the workflow is to use the cluster name provided by the API gateway and look up the management network, storage network, and storage arrays associated with that cluster. For each of the three outputs, it looks up the information using a predefined List Of Values (LOV) stored in the custom workflow inputs in Cisco UCS Director. This makes adding new environments simply a matter of creating another record in the LOV tables for the new environment.

The custom workflow input tables have two columns, one for the label and the other for the value. Table 4 shows an example.

Table 4. Example of a custom workflow input table (LOV)

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC1-TS-GEN</td>
<td>10.10.12.0/24</td>
</tr>
<tr>
<td>DC2-TS-SQL</td>
<td>10.20.14.0/24</td>
</tr>
<tr>
<td>DC2-PR-DMZ</td>
<td>10.20.18.0/24</td>
</tr>
<tr>
<td>DC3-PR-GEN</td>
<td>10.30.122.0/23</td>
</tr>
</tbody>
</table>

There is a separate custom workflow input table each for:

- **Management networks:** Used for VMkernel connectivity dedicated to VMware Management and vMotion traffic
- **Storage networks:** Used for VMkernel connectivity dedicated to storage (NFS/iSCSI) traffic
- **Storage arrays:** Used to identify the aliases for each storage array to connect to for that cluster (used for Brocade zoning) and NetApp iGroup scripting

The sample script below takes two inputs (Table 5), such as DC3-PR as the site, and the cluster as GEN, and combines them to create the lookup value of DC3-PR-GEN. The result of the lookup returns an output (Table 6) of the network name as a value, in this case, 10.30.122.0/23 as the Management Network. The code uses the same input to also return the Storage Network and Storage Arrays values (Table 6) from additional LOV tables. These outputs are stored and used in subsequent tasks within the workflow.

Table 5. Infoblox workflow environment lookup inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site Name</td>
<td>Logical Datacenter Name (explicit)</td>
</tr>
<tr>
<td>Cluster Name</td>
<td>vSphere Cluster Name (explicit)</td>
</tr>
</tbody>
</table>

Table 6. Infoblox workflow environment lookup outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management network</td>
<td>Network name in CIDR format for vmk0, the ESXi management network</td>
</tr>
<tr>
<td>Storage network</td>
<td>Network name in CIDR format for vmk1, the IP storage network (NFS)</td>
</tr>
<tr>
<td>Storage arrays</td>
<td>List of values corresponding to the storage array names, used when creating the zoning in the Brocade Fibre Channel switches</td>
</tr>
</tbody>
</table>
Environment lookups

// Create the variable to match the Label in the LOV
var code = (input.Site + "-" + input.Cluster).toLowerCase();

// Function which performs the lookup in the LOV
function checkmgmtLOV(customlov, code) {
    logger.addInfo("Checking in : "+customlov);
    logger.addInfo("Checking for : "+code);
    var input = CustomActionUtil.getWFCustomLOVInput(customlov);
    var customLOVList = CustomActionUtil.getWFCustomLOVPairs(input.getCustomValuesProvider());

    if(customLOVList==null) {
        logger.addError("No Value find in the Custom Input LOV" + customlov);
    } else {
        for(var i=0; i<customLOVList.size(); i++) {
            if((customLOVList[i].getLovLabel()).toLowerCase().equals(code)) {
                logger.addWarning("Value exists for :
" + customLOVList[i].getLovLabel() + " with value "+customLOVList[i].getLovValue());
                return customLOVList[i].getLovValue();
                break;
            }
        }
        logger.addError("No Value find in the Custom Input LOV: " + customlov + " for "+input.Site + "-" + input.Cluster);
    }
}

// Creates the variable which holds the output for the Management Network value
var mgmt_net = checkmgmtLOV("OC_mgmt_subnet", code);
output.mgmt_net = mgmt_net;

// Creates the variable which holds the output for the Storage Network value
var storage_net = checkmgmtLOV("OC_storage_subnet", code);
output.storage_net = storage_net;
ctxt.setTaskStatusMessage("Mgmt: "+mgmt_net + " Storage: "+storage_net);

// Creates the variable which holds the output for the Storage Arrays value
var targets = checkmgmtLOV("OC_targets", code);
output.targets = targets;
Step 2: Infoblox ESXi management IP
Overview: Reserve the ESXi Management IP in Infoblox

Director task type: Custom

Description: In this task, an IP address for the ESXi management interface needs to be assigned from the IPAM system, Infoblox.

The first element of IPAM integration includes claiming an IP address for each VMkernel interface and creating an A record for the ESXi management interface. This custom task has inputs for management network, hostname, domain, and the Infoblox server and credentials (Table 7). The outputs are IP address, comment, and registered host (Table 8).

Table 7. Infoblox workflow management inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management network</td>
<td>CIDR network name derived from the LOV in the previous task</td>
</tr>
<tr>
<td>Host name</td>
<td>Name of the Cisco UCS service profile, which matches the hostname for the DNS record created in Infoblox</td>
</tr>
<tr>
<td>Domain</td>
<td>Name of the domain to use as part of the Fully Qualified Domain Name (FQDN) for the A record of the management IP address</td>
</tr>
<tr>
<td>Infoblox server</td>
<td>IP address of the Infoblox API interface</td>
</tr>
<tr>
<td>Infoblox Username</td>
<td>Username of the Infoblox service account</td>
</tr>
<tr>
<td>Infoblox password</td>
<td>Password of the Infoblox service account (obfuscated by the password user input data type)</td>
</tr>
</tbody>
</table>

Table 8. Infoblox workflow management outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>Actual IP address identified and reserved by the custom task</td>
</tr>
<tr>
<td>Registered host</td>
<td>FQDN of the hostname, used in the A record</td>
</tr>
<tr>
<td>Comment</td>
<td>Predefined comment written to the service request details</td>
</tr>
</tbody>
</table>

Step 3: Infoblox storage IP
Overview: Reserve the ESXi storage IP in Infoblox

Director task type: Custom

Description: This workflow task reuses the task from Step 2. The same code is used for the management IP and the storage IP, but the execution varies depending on the inputs. The primary difference is that for the storage IP, an A record is not created in DNS, since we don’t need DNS resolution for the storage network interface.

Alternate inputs are passed to the code that triggers allocation of the storage IP. The inputs are storage network, hostname, domain, and the Infoblox server and credentials (Table 9). The outputs are IP address, comment, and registered host (Table 10).
Table 9. Infoblox workflow storage inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage network</td>
<td>CIDR network name derived from the LOV in the GetHostNetworks task</td>
</tr>
<tr>
<td>Hostname</td>
<td>Name of the Cisco UCS service profile, which matches the hostname for the DNS record created in Infoblox</td>
</tr>
<tr>
<td>Domain</td>
<td>Name of the domain to use as part of the FQDN for the A record of the management IP address</td>
</tr>
<tr>
<td>Infoblox server</td>
<td>IP address of the Infoblox API interface</td>
</tr>
<tr>
<td>Infoblox username</td>
<td>Username of the Infoblox service account</td>
</tr>
<tr>
<td>Infoblox password</td>
<td>Password of the Infoblox service account (obfuscated by the password user input data type)</td>
</tr>
</tbody>
</table>

Table 10. Infoblox workflow storage outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>Actual IP address identified and reserved by the custom task</td>
</tr>
<tr>
<td>Registered host</td>
<td>FQDN of the hostname, used in the A record</td>
</tr>
<tr>
<td>Comment</td>
<td>Predefined comment written to the SR details</td>
</tr>
</tbody>
</table>

Step 4: Assign vMotion IP

**Overview:** Self-assign vMotion IP for ESXi host.

**Director task type:** Custom

**Description:** The vMotion network is an unrouted /23 network, so the IP addresses do not need to be reserved in Infoblox. However, this step comes after the Infoblox reservations because the fourth octet of the management IP is the same as the fourth octet of the two vMotion IPs to keep things consistent and eliminate the need for any additional reservations or tracking. For example, if the assigned management IP is 10.1.1.52, then the A-side vMotion IP becomes 192.168.100.52, and the B-side vMotion IP becomes 192.168.101.52. The workflow for vMotion IPs has only one input, for the management IP (Table 11), and two outputs, the A-side and B-side vMotion IP addresses (Table 12).

The reason for two vMotion uplinks in vCenter – and therefore two IP addresses – is to maximize and balance vMotion traffic across both fabric interconnects. This effectively doubles the amount of simultaneous vMotion tasks from 8 to 16 per ESXi host, resulting in faster evacuation times. This design decision should be based on expected traffic loads and uplink speeds.

The reference to A-side and B-side is how we identify the upstream connections. Connections made through fabric interconnect A are referred to as A-side in our code and documentation. Likewise, connections through fabric interconnect B are referred to as B-side.

We create a pair of vNICs per function. In our case it’s 8 total vNICs per host: two for management/vmotion, two for storage, two for non-secured VM traffic, and two for secured VM traffic. This practice is explained earlier in the Cisco UCS → vNIC templates section.

Table 11. vMotion IP inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management IP address</td>
<td>IP address of the VMkernel management interface, as identified earlier in the InfobloxMgmtIP task</td>
</tr>
</tbody>
</table>
### vMotion IP outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vMotion A</td>
<td>vMotion IP for A-side interface</td>
</tr>
<tr>
<td>vMotion B</td>
<td>vMotion IP for B-side interface</td>
</tr>
</tbody>
</table>

#### vMotion IP assignment

```javascript
importPackage(java.lang);
importPackage(java.util);

var mgmt_ip = input.ipaddress;
var vmotionip1 = "192.168.100." + mgmt_ip.slice(mgmt_ip.lastIndexOf(".")+1);
output.vm1_ip = vmotionip1;
logger.addInfo("vMotion1 IP address: " + vmotionip1);

var vmotionip2 = "192.168.101." + mgmt_ip.slice(mgmt_ip.lastIndexOf(".")+1);
output.vm2_ip = vmotionip2;
logger.addInfo("vMotion2 IP address: " + vmotionip2);
ctxt.setTaskStatusMessage("vMotion IPs: " + vmotionip1 + " & " + vmotionip2);
```

### Step 5: Create Brocade zoning

**Overview:** SSH script to provision the Brocade zones and aliases

**Director task type:** Custom

**Description:** The storage zoning targets Brocade switch addresses, and login credentials are stored in a Cisco UCS Director custom input table. The Director workflow determines which of these to use based on the service profile template. The workflow then executes a custom SSH task to set up the zoning aliases for the new host on the Brocade switches and enable the configuration.

In addition to IP storage, each host is connected to storage by Fibre Channel as well. The custom task to create Brocade aliases and zones has inputs for site, hostname, vHBA1, vHBA2, and storage targets. The only output is a comment for tracking the Director SR number. The task is executed as a shell script over SSH. To account for multiple pairs of Brocade switches, we execute a “case” statement to determine which switch to connect to and execute the commands.

**Brocade zoning script**

```javascript
var site = (input.site).toLowerCase();
var hostname = input.hostname;
var targets = (input.targets).toLowerCase();

switch (site) {
  case 'dc1':
    output.switch_A = "dc1brocade01.alaskaair.com"; // (10.10.10.21)
    output.switch_B = "dc1brocade02.alaskaair.com"; // (10.10.10.22)
    break;
```
case 'dc2':
    output.switch_A="dc2brocade01.alaskaair.com"; //(10.12.10.21)
    output.switch_B="dc2brocade02.alaskaair.com"; //(10.12.10.22)
    break;

case 'dc3':
    output.switch_A="dc3brocade01.alaskaair.com"; //(10.20.10.21)
    output.switch_B="dc3brocade02.alaskaair.com"; //(10.20.10.22)
    break;

case 'dc4':
    output.switch_A="dc4brocade01.alaskaair.com"; //(10.21.10.21)
    output.switch_B="dc4brocade02.alaskaair.com"; //(10.21.10.22)
    break;

default:
    ctxt.setFailed("No Site defined in the Brocade Custom task");
    ctxt.exit();
    break;

}ctxt.setTaskStatusMessage("Brocades: "+output.switch_A +" & "+output.switch_B);
output.aliname_A="vHBA_"+hostname+_a"
output.aliname_B="vHBA_"+hostname+_b"
if (targets.contains("nimble")){
    output.zonename_A=hostname+_Nimble_A"
    output.zonename_B=hostname+_Nimble_B"
}else{
    output.zonename_A=hostname+_NetApp_A"
    output.zonename_B=hostname+_NetApp_B"
}
output.target_A=input.targets+";"+output.aliname_A;
output.target_B=input.targets+";"+output.aliname_B;

logger.addWarning("Brocades: "+output.switch_A +" & "+output.switch_B);
logger.addWarning("Alias: "+output.aliname_A +" & "+output.aliname_B);
logger.addWarning("Zone Name: "+output.zonename_A +" & "+output.zonename_B);
logger.addWarning("Targets: "+output.target_A +" & "+output.target_B);
Step 6: Check Cisco UCS server pool inventory

Overview: Checks UCS Central for available server pool inventory

Director task type: Built-in

Description: This is a built-in workflow task, with no outputs. The inputs are to identify which UCS Central server to perform the collection action against in order to determine which resources need to be inventoried (Table 13). In this case, the resources selected are service profiles, local service profile templates, and local service profiles.

Table 13. RequestInventoryCollection inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central identity</td>
<td>Specifies the UCS Central entity defined in Director</td>
</tr>
<tr>
<td>Central resources</td>
<td>Specifies which resources to perform the inventory refresh action against</td>
</tr>
</tbody>
</table>

Step 7: Create service profile

Overview: Creates service profile from template and assigns to Cisco UCS blade server

Director task type: Built-in

Description: A new service profile is generated using the service profile template. The service profile template defines several configuration items on the host, including the pool of physical servers the host is created from, the MAC addresses, WWPN addresses, and network connectivity templates.

This step initiates the allocation and configuration of a Cisco UCS blade server and starts the provisioning process.

Table 14. Cisco UCS Service Profile inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>Host name derived from Infoblox query during the API gateway step</td>
</tr>
<tr>
<td>Service profile template</td>
<td>Name of the service profile template derived from naming conventions</td>
</tr>
<tr>
<td>Service profile naming convention</td>
<td>Required input by the built in task, set to manual</td>
</tr>
<tr>
<td>Destination organization</td>
<td>Name of the UCS organization being provisioned into</td>
</tr>
</tbody>
</table>

Step 8: Wait for server power on

Overview: Pauses workflow for 5 minutes for server provisioning

Director task type: Built-in

Description: This task has no inputs or outputs; it simply uses a built-in function to pause the workflow for a specified amount of time. This exists to give Cisco UCS Central time to create the service profile and associate it to a blade, for the blade to power up and go through the post operation, and finally, if needed, for the firmware policy specified in the associated service profile to be applied. Note that if a firmware update does occur during this step, it will last longer than the five minutes defined in this task.

Nothing breaks if it takes longer than five minutes. The reason this task exists is simply to pause the workflow while the server goes through the boot up process. If this task did not exist, the next task would simply start sooner, and would go through more retries while the server boots up.
**Step 9: Server PXE boot and VMware Auto Deploy**

**Overview:** Server PXE boots and leverages VMware Auto Deploy

**Director task type:** Not applicable

**Description:** After the service profile has been applied to the blade, the physical server is powered on and goes through a PXE boot process using VMware Auto Deploy. The VMware Auto Deploy rules are configured to associate a specific pool of MAC addresses to a specific cluster based on the Cisco UCS schema described above. This is the same pool of MAC addresses that was pulled from when deploying the host using the service profile template in Central.

Once this step is complete proceed to step 10.

**Step 10: Check host online**

**Overview:** Monitor for ESXi host registration in vCenter

**Director task type:** Custom

**Description:** A custom host detection task polls the vCenter cluster inventory looking for the new host. Since the new host is added to the cluster by Auto Deploy via a DHCP address for the initial configuration, the inventory polling service uses the MAC addresses passed in from the service profile creation step to determine which host has been deployed by the workflow.

The host discovery is performed by a custom Cisco UCS Director workflow task written in JavaScript. The script pulls an inventory of all the hosts in the input cluster, iterates through each one, and checks if any of the MAC addresses match the MAC address on vNIC1 (Table 15). As shown in Figure 14, if a match is found, the task is successful and the DHCP host IP address is returned.

After successfully detecting the host, the custom workflow task passes host identity data, such as the name and IP address assigned via DHCP, on to the next task in the chain as workflow task outputs (Table 16).

*Figure 14. vCenter host discovery logical process flow*
Table 15. vCenter host discovery inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter cluster</td>
<td>vCenter cluster name</td>
</tr>
<tr>
<td>VNIC1 MAC</td>
<td>MAC address derived from the Cisco UCS service profile creation</td>
</tr>
</tbody>
</table>

Table 16. vCenter host discovery outputs

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host IP</td>
<td>IP address of the host discovered in the vCenter host cluster</td>
</tr>
<tr>
<td>VMware account name</td>
<td>Cisco UCS Director account connection used to query vCenter</td>
</tr>
<tr>
<td>vCenter server</td>
<td>Name of the vCenter server queried</td>
</tr>
<tr>
<td>Site name</td>
<td>Director site queried</td>
</tr>
<tr>
<td>Cluster name</td>
<td>vCenter cluster queried</td>
</tr>
</tbody>
</table>

Step 11: Customize ESXi host

Overview: Calls an external PowerShell program that configures ESXi host

Director task type: Built-in

This step of the workflow executes a custom script to apply the following host profile customizations to the newly deployed host:

- Hostname
- Management network IP and subnet mask
- Storage network IP and subnet mask
- vMotion IPs and network masks

After applying the configuration, the host is rebooted to complete the remediation process, at which point the initial DHCP address used during PXE boot is released and host comes online with the statically assigned address.

The ExecutePowershellCommand task is a built-in Cisco UCS Director task that executes a custom PowerShell script remotely from a Windows server. This task connects to a Director PowerShell Agent (Table 17), which is software installed on a Windows server to facilitate the remote execution of scripts. This connection is configured and established on the Director server using LDAP authentication, and it can be used to execute PowerShell commands in any number of workflows.

Table 17. Host profile customization inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PowerShell Agent user</td>
<td>Domain user account used to execute the script on the remote windows server</td>
</tr>
<tr>
<td>PowerShell Agent password</td>
<td>Domain account password used to execute the script on the remote windows server</td>
</tr>
</tbody>
</table>

Table 18. Host profile customization output

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POWERSHELL_COMMAND_RESULT</td>
<td>Reports the success or failure of the PowerShell command execution</td>
</tr>
</tbody>
</table>
In the configuration of the built-in `ExecutePowershellCommand` task, we specify which agent server we wish to use, as well as the authentication details for the service account and the PowerShell command we wish to execute. The required values from the task configuration are:

- **Label**: The name of the task as it will appear in the workflow.
- **PowerShell Agent**: The PowerShell agent connection configured on the remote Windows server.
- **Target machine IP**: The IP or FQDN of the server the script is being executed on. This is the PowerShell Agent server itself.
- **Commands/script**: This is the command string used to execute the PowerShell script including the required parameters. Some of these parameters are referenced from the output of previous tasks and are called with the syntax `${WorkflowTaskName.OutputVariableName}`. The command example is shown below.

**Example `ExecutePowershellCommand` command syntax**

```
D:\myscripts\answerfilecust.ps1 ${CheckHostOnline.vserver} ${PSA User} ${PSA Password} "${CheckHostOnline.clusterName}" ${SPT.SP_VNIC1_MAC} ${vMotionIP.vm1_ip} ${vMotionIP.vm2_ip} ${InfobloxStorageIP.IP} ${storagemask} ${InfobloxMGMTIP.IP} ${mgmtmask} ${SPT.SERVICE_PROFILE_NAME}
```

- **CheckHostOnline.vserver**: Target vCenter server of discovered host
- **PowerShell Agent User**: PowerShell agent user defined in task input
- **PowerShell Agent Password**: PowerShell agent password defined in task input
- **CheckHostOnline.clusternname**: Target cluster of discovered host
- **SPT.SP_VNIC1_MAC**: VNIC1 MAC address derived from the creation of the service profile
- **vMotionIP.vm1_ip**: IP of vMotion NIC 1
- **vMotionIP.vm2_ip**: IP of vMotion NIC 2
- **InfobloxStorageIP.IP**: IP of storage NIC
- **Storagemask**: Storage NIC subnet mask
- **InfobloxMGMTIP.IP**: IP of management NIC
- **Mgmtmask**: Management NIC subnet mask
- **SPT.SERVICE_PROFILE_NAME**: Hostname derived from the creation of the service profile

The result of the PowerShell execution (Table 18) tells UCS Director if the task succeeded or failed.
Step 12: Host configuration compliance validation

Overview: Calls an external PowerShell program that will configure ESXi host

Director task type: Built-in

Description: After the host profile association and reboot, the host rejoins the cluster with the finalized hostname and IP configuration. If any host profile compliance issues occur, the host does not exit maintenance mode and the workflow logs an error message.

Host profile compliance is used to determine if a host is ready to serve active workflow. In the host profile, we have defined dozens of configuration policies that should be applied to the host. Many of these configuration items are unique based on the cluster the host is deployed to. Some examples of these are:

- Portgroups for virtual machine traffic (VLANs)
- Storage paths
- Syslog locations
- Host management network configuration
- vMotion network configuration
- Storage network configuration

If vCenter determines the host is compliant with the host profile policy, the host configuration has been successful and the host exits maintenance mode. After the host exits maintenance mode, it begins serving production workloads automatically. Note that the host compliance validation occurs automatically and is not initiated or managed by the Cisco UCS Director workflow.

If the host is not compliant with the host profile, it rejoins the cluster but does not exit maintenance mode. Host profile compliance issues usually require manual investigation, so if this step of the Director workflow is not completed successfully a notification will be sent out so the issue can be remediated. Below are some examples of host profile compliance issues that could be encountered:

- **Missing storage paths:** There may have been an issue with the storage zoning, so the host is not seeing all the expected datastores.

- **Issues with the stateless cache:** If the stateless cache has gone stale, it will cause a failure and won’t join the cluster. This condition is usually resolved by formatting the SD cards and rebooting.

- **Incorrect number of uplinks:** The number of vNICs in the service profile template must match the number of configured uplinks in the host profile.
Step 13: Verify configuration errors

Overview: Calls JavaScript program that check that Steps 11 and 12 exited error free

Director task type: Custom

Description: This step of the workflow will validate whether the PowerShell script in Step 11 executed successfully (Table 19). If that step fails, it indicates an issue with the host profile customization, and the service request terminates in a failed state.

Table 19. PowerShell Agent error-checking inputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POWERSHELL_COMMAND_RESULT</td>
<td>Reports the success or failure of the PowerShell command execution from the Step 11 task</td>
</tr>
</tbody>
</table>

This custom task contains a simple JavaScript check to verify success.

PowerShell Agent error-checking script

```javascript
var txt = input.InputString;

if(txt.contains("Error")){
    logger.addError("Error: In PowerShell Task");
    ctxt.setFailed(txt);
    ctxt.exit();
}
```

Step 14: Get KVM IP

Overview: JavaScript that calls Cisco UCS Central to determine the KVM IP address

Director task type: Custom

Description: This custom task runs a JavaScript that builds and executes a REST API call to the Cisco UCS Central appliance to determine the configured KVM IP address pulled from a pool to be used on the host's service profile (Table 20). The workflow then uses this IP and hostname (Table 21) in a custom script in the next task to register a KVM host record in Infoblox.

Table 20. Inputs for the Get KVM IP custom task

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCS Central service profile identity</td>
<td>Reference to the service profile that has been deployed by the current workflow</td>
</tr>
</tbody>
</table>

Table 21. Outputs for the Get KVM IP custom task

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-of-band IP</td>
<td>IP address that was assigned to the service profile for out-of-band management, defined by the service profile template</td>
</tr>
<tr>
<td>Host name</td>
<td>Hostname of the ESXi host to be used in Infoblox DNS record creation in the next task</td>
</tr>
</tbody>
</table>
Step 15: Infoblox KVM IP

Overview: Registers KVM IP from Step 14 with Infoblox

Director task type: Custom

Description: Similar to the custom script for reserving the management and storage IPs, this script takes the KVM IP identified from Step 14 and registers it in Infoblox (Table 22) as an A record. The record is created with a custom DNS name based on the requirements of our environment and therefore relies on the domain input to specify the custom zone.

This task is optional and can fail without causing the failure of the overall workflow. The requirement being met here is simply to create an A record for operations users to access a host’s out-of-band management interface based on its name.

Table 22. Inputs for the Infoblox KVM IP custom task

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM IP address</td>
<td>Out-of-band management IP identified in the previous task</td>
</tr>
<tr>
<td>Hostname</td>
<td>Name of the Cisco UCS service profile, which matches the hostname for the DNS record created in Infoblox</td>
</tr>
<tr>
<td>Domain</td>
<td>Name of the domain to use as part of the FQDN for the A record of the management IP address</td>
</tr>
<tr>
<td>Infoblox server</td>
<td>IP address of the Infoblox API interface</td>
</tr>
<tr>
<td>Infoblox username</td>
<td>Username of the Infoblox service account</td>
</tr>
</tbody>
</table>

Table 23. Outputs from the Infoblox KVM IP custom task

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered host</td>
<td>FQDN of the DNS record that was registered</td>
</tr>
</tbody>
</table>

Step 16: Add notes to service request

Overview: Writes data to Cisco UCS Director service request

Director task type: Custom

Description: This step in the workflow adds comments to the service request indicating the site, cluster, and hostname (Table 24) of the host deployed. This can then be viewed in the historical log of executed service requests and are intended to ensure a service request for a specific host is easily identifiable. Without these notes, each service request must be opened and inspected to find these details.

Table 24. Service request comments input

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiteName</td>
<td>Data center site derived from host discovery</td>
</tr>
<tr>
<td>ClusterName</td>
<td>vCenter cluster derived from host discovery</td>
</tr>
<tr>
<td>Service profile name</td>
<td>Service profile derived from the service profile creation</td>
</tr>
</tbody>
</table>

This custom task contains a custom script to add the initiator comments to the service request.
Adding service request comments script

```java
importPackage(java.util);
importPackage(java.lang);
importPackage(com.cloupia.lib.util);
importPackage(com.cloupia.model.cIM);
importPackage(com.cloupia.service.cIM.inframgr);

function addComments(comment){
    var srId = ctxt.getSrId();
    if (input.SR_Comments=="" || input.SR_Comments==null || (input.SR_Comments).isEmpty()){
        var comm = input.site+"@"+input.cluster + "@" + input.host;
    }else{
        var comm = input.site+"@"+input.cluster + "@" + input.host + "@"+input.SR_Comments;
    }
    var comments = String(comm);
    ctxt.setTaskStatusMessage(comments);
    logger.addInfo("Added Initiator Comments: " + comments);
    var sr = InfraPersistenceUtil.getServiceRequest(Integer.parseInt(srId));
    sr.setComments(comments);
    var result = InfraPersistenceUtil.modifyServiceRequest(sr);
    logger.addWarning("Added Initiator Comments to SR_ID: " + result);
}
addComments(input.SR_Comments);
```

Documentation and Notifications

Upon successful completion of the workflow, the last step is to update the configuration management database and, if desired, send notifications to administrators and other stakeholders.

Challenges and limitations

There is no one-size-fits-all solution for anything in IT, and this document is no exception. The following are some of the things to consider when determining if this solution is right for your environment.

- **Connectivity:** The network architecture will be a major factor in implementing any kind of resource pool-based design. The limiting factor here is that if your networks are segmented in such a way that fabric interconnect domains serve independent network environments, you may lose flexibility of resource allocation. In larger environment, this factor is less of an issue, but in some smaller environments it could be functionally prohibitive. Network connectivity limitations don’t affect the value of configuration control or speed of provisioning, but it may not be possible to fluidly move capacity between environments.
- **Hardware and host profiles**: Because vSphere host profiles generally have a one-to-one ratio to hardware types, if your environment does not have a standard hardware configuration, it may be difficult to manage the large number of hardware types and their associated host profiles. This potential issue is strongly mitigated if each vSphere cluster contains only a single hardware model. It is worth noting that host profiles are very difficult to implement well without the benefits of Cisco UCS service profiles.

- **Environment Size**: The frequency of new host builds is only one factor in determining the value of an automation solution, but it is a notable one. If your environment only adds a new host a few times per year, you will obviously not see the same value from automating that process.

Keep in mind that even if this complete solution does not meet your business requirements, some elements may still be worth implementing. Also, be aware that some pieces may need considerable modification to function for your environment.

**For more information**

- **Cisco UCS PowerTool Suite Community**: https://communities.cisco.com/docs/DOC-37154
- **Cisco UCS PowerTool Suite download**: https://software.cisco.com/download/home/286305108/type/284574017/release/2.3.1
- **Cisco UCS PowerTool Command Reference**: https://communities.cisco.com/docs/DOC-57485
- **Infoblox API examples**: https://community.infoblox.com/t5/API-Integration/The-definitive-list-of-REST-examples/td-p/1214
- **NetApp OnTap 9.x Documentation**: http://docs.netapp.com/ontap-9/index.jsp
- **VMware vSphere and ESXi documentation**: https://docs.vmware.com/en/VMware-vSphere/index.html
- **VMware PowerCLI**: https://code.vmware.com/web/dp/tool/vmware-powercli/10.0.0
Appendix

Product listing
The following hardware and software products and versions were used to deploy this solution:

- Cisco UCS Director 6.5.0.2 (65940)
- Cisco UCS Central 1.5(1a)
- VMware vCenter 6.5 U1 (5973321)
- VMware ESXi 6.5.0d (5310538)
- VMware vRealize Operations Manager 6.6.1.6163035
- Infoblox (ib-1420 model, 8.0.5-351775 software)
- Brocade fabric-os v8.2.0 (model x4s)
- NetApp Data ONTAP 9.1p8

API server software configuration

- Red Hat Enterprise Linux Server release 7.4
- Python 2.7.5
- Flask 0.12.2
- Requests 2.6.0
- Coverage 3.6b3
- Kafka-python 1.4.1

Python module requirements

- certifi >= 2016.2.28
- Flask >= 0.10.1
- requests >= 2.9.1
- kafka-python >= 1.1.3