PURESTORAGE®
Uncomplicate Data Storage, Forever

WHITE PAPER

Infrastructure as Code Basics for FlashStack

Contents

Introduction

This document is meant to build a baseline of the FlashStack infrastructure components & checks which are included in any IaC-focused guides or code that we produce. This structure is designed to be the framework for a modular IaC codebase that can handle the automated deployment of multiple FlashStack configurations and configuring specific workloads deployed on FlashStack.

What is Infrastructure as Code?

Before we discuss the infrastructure components deployed with Infrastructure as Code (IaC), we need to ensure that the reader understands what Infrastructure as Code is and what we are trying to accomplish. Infrastructure as Code (IaC for short) is about managing infrastructure in a descriptive model. We define the desired state of the platform we are managing and leverage source control for versioning the configurations applied to the environment.

IaC uses tools built to take our declaration of the end state for our configuration, process the current state of the environment, and make the appropriate changes to set the configuration as we declared. These tools deliver an idempotent result, meaning that the result is the same whether we run the code 1 to 1000 times and work to deliver our desired state.

Before IaC, we used an imperative model of defining each step in deploying infrastructure. These steps could be clicks in a GUI, commands at the CLI, or even a script run to make all the changes. Instead, IaC uses a declarative model where we define our configuration, and a tool makes the required changes to get to that final state.

It is a significant change to get out of handling our deployments with manual interaction or even scripts, which still had us responsible for all the logic and decision-making. There are many benefits to leveraging IaC, but the most prominent benefit of this process is that we only need to concern ourselves with the configuration we need within our environment.

With that basic description of IaC, hopefully, everyone reading this document can grasp the basic concept of writing a simple text file that defines some infrastructure's configuration, then telling a tool to deploy that configuration. Our deployments are based on files using variables, allowing any user to define their specific naming and IP addresses. This method of setting configuration specifics also means that users can perform deployments without requiring an understanding of all of the code/tooling used to be capable of modifying the configuration being deployed.

Deploying a FlashStack

Now, we discuss FlashStack specifically and which hardware components we are deploying. To cover the use cases for every FlashStack deployment, automation of the deployment encompasses the network (Nexus switches), the compute environment (Cisco UCS), and the storage (Pure Storage).

In addition, if the use-case or architecture specifies fiber-channel rather than ethernet connectivity to storage, the automation also includes deploying the fiber-channel network (Cisco MDS).

Finally, beyond the deployment of the infrastructure, the automation moves on to the deployment of operating systems/hypervisors and potentially to applications/workloads.

As much as it pains me to admit this based on "blaming the network" for many issues during my career, the basis for automating any deployments of an entire FlashStack or any individual components relies on the network – more specifically, connectivity to our environment. Therefore, we must have basic connectivity for connecting to the hardware components and management interfaces to make any of this work.

Network Configuration - Ethernet

We expect any new deployment has the base configuration of network switches completed, and these switches are accessible remotely with an IP address. Some files containing variables need to be updated, as these control some specific features being enabled and configured. These variables also provide global settings, interfaces/descriptions, VLAN IDs, and IPs other settings (VPCs, NTP, etc.) Once any required variables have been set for our environment, we begin to deploy the configuration by running the tasks for specific functions, such as:

- Configuring MDS features
- Configuring global settings
- Configuring NTP settings
- Configuring interface ports
- Configuring VSANs & enable smart zoning
- Configuring the device aliases
- Configuring zonesets & zones for fiber-channel
- Configuring zonesets & zones for fiber-channel NVMe, if defined
- Activate the zonesets
- Saving the configuration

Upon completing these tasks, all of the MDS features and configurations required for fiber-channel connectivity within the FlashStack platform are in place, and we can proceed to the next step.

Storage Configuration

After the network configuration, the next component to be deployed is the Pure FlashArray, so that our volumes (specifically boot volumes) are accessible for the servers when they are booted to deploy an operating system. The Pure configuration is split into 2 pieces logically: the initial setup of the FlashArray, then the configuration for the hosts/host groups/ volumes. Some specific variables drive the Pure workflow to determine which configuration tasks are run:

- Is the initial configuration being run?
- Is iSCSI being configured?
- Is fiber-channel being configured?
- Is NVMe-RoCE being configured?
- Is NVMe-FC being configured?

As for the deployment of a FlashArray, we only assume that there is a management IP address configured and that we have an API key for the automated deployment. For the initial configuration of a FlashArray, we have variables to set the array name, interfaces, and settings for DNS, NTP, SMTP, proxy, phone home, remote assist, AD, and iSCSI.

As for the specific items that are configured during the initial configuration task, these steps are run:

- Setting the Array Name
- Configuring DNS
- Configuring NTP
- Configuring SMTP (if defined)
- Configuring proxy settings (if defined)
- Configuring alert emails (if defined)
- Configuring phone home
- Configuring remote assist
- Configuring Active Directory roles
- Configuring Active Directory connectivity
- Configuring iSCSI interfaces (if defined)

Once the initial configuration steps are completed, we move to the second configuration phase for the FlashArray, where we provision volumes and create hosts/host groups for server access. There are variables to define the host objects running each protocol for iSCSI, fiber-channel, and NVMe-FC, along with their appropriate initiators (IQN, WWN, or NQN), plus the host groups for each protocol type. There is another set of variables to control the size of boot from SAN volumes, along with the size & number of data volumes (typically datastores).

As for the specific items that are configured during the host & volume configuration task, these steps are run:

- Create data volumes (typically datastore)
- Create swap volumes (if defined)
- Create Boot from SAN volumes for iSCSI (if defined)
- Create Boot from SAN volumes for fiber-channel (if defined)
- Create Boot from SAN volumes for NVMe (if defined)
- Create Host Group for iSCSI (if defined)
- Create Host Group for fiber-channel (if defined)
- Create Host Group for NVMe-FC (if defined)
- Create Host objects for iSCSI (if defined)
- Create Host objects for fiber-channel (if defined)
- Create Host objects for NVMe (if defined)
- Add Host objects to Host Group for iSCSI (if defined)
- Add Host objects to Host Group for fiber-channel (if defined)
- Add Host objects to Host Group for NVMe (if defined)

- Add shared volumes to Host Group for iSCSI (if defined)
- Add shared volumes to Host Group for fiber-channel (if defined)
- Add shared volumes to Host Group for NVMe (if defined)
- Create VMware protocol endpoint for iSCSI (if defined)
- Create VMware protocol endpoint for fiber-channel (if defined)

Once the overall configuration for the FlashArray is completed, we move to the deployment of the UCS compute environment.

Server Configuration

As the UCS platform is based upon policies and templates, there is a significant amount of initial configuration upfront, but then operations are simplified for the platform lifecycle. UCS relies upon many components which need to have their settings or policies defined, and this is handled through multiple tasks which address each of the layers of the Cisco UCS environment:

- UCS Equipment configuration (hardware-focused policies)
- UCS Administration configuration (platform level settings)
- UCS LAN configuration (network-focused policies)
- UCS SAN configuration (external storage-focused policies)
- UCS Server configuration (server template & profile-based policies)

Without diving too far into the details of the steps underneath each layer, which may require more in-depth knowledge of the UCS platform than we cover here, this brief overview only describes the types of configurations handled in each step:

- UCS Equipment – Network uplinks, discovery of chassis/rack servers, UDLD, and neighbor discovery
- UCS Administration – DNS, NTP, & Time zone settings, creating the organization structure, creating a dedicated user account
- UCS LAN – Creation of pools for management IPs, iSCSI IPs (if defined), and mac addresses, creating a QoS system class, creating VLANs, creating policies for network control (CDP & LLDP), workload-specific adapter policies, iSCSI connectivity (if defined), and fiber-channel connectivity (if defined), creating templates for vNICs and iSCSI vNICs (if defined)
- UCS SAN – Creation of pools for IQNs, WWNNs, and WWPNs, creating VSANs for each fabric (if defined for fiber-channel), creating vHBA templates, and enabling VSAN trunking (if required for fiber-channel uplinks)
- UCS Server – Creation of pools for UUIDs and server resource pools, defining host firmware packages, creation of policies for power control, maintenance, adapters, local disks, IPMI access, vMedia, BIOS, iSCSI boot (if defined), and fiber-channel boot (if defined), creation of service profile templates (if defined)

Once these configuration tasks for each layer are completed, we have reached the point where servers leveraging only iSCSI can be powered on, and they can have an operating system or hypervisor installed. If fiber-channel configuration has been defined, then we need another step in the process to configure our MDS switches.

Network Configuration - Fiber-Channel

With fiber-channel connectivity defined for our environment, we need to run through the tasks to configure our fiber-channel networking on Cisco MDS switches. These devices are managed similarly to our Nexus switches for traditional ethernet networking due to being a device running NX-OS. Still, they are not the same or run the same configuration tasks.

Some files containing variables need to be updated, as these control some specific features being enabled and configured. These variables also provide global settings, interfaces/descriptions, port-channels, VSAN IDs, storage WWPNs, and IDs used for other settings (zonesets, zones, aliases, etc.) Once the required variables have been set, we begin to deploy the configuration by running the tasks for specific functions, such as:

- Configuring switch features
- Configuring global settings
- Configuring VLANs
- Configuring NTP settings
- Setting the default gateway for network traffic
- Configuring interface ports
- Configuring VPC settings
- Backing up the configuration

Upon completing these tasks, all of the network features and configurations required by the other components of the FlashStack platform are in place, and we can proceed to the next step.

Workloads

With all these components of the FlashStack brought online, we have completed the base configuration required for us to begin installing an operating system (whether hypervisor or baremetal), and then we can provision workloads afterwards.

VMware

Typically we see VMware as the primary OS installed on the UCS servers within FlashStack. To deploy this hypervisor, we must first deploy ESXi hosts before we deploy vCenter for management.

Variables files define the ESXi username & password, NTP servers, management uplinks, hostnames, IP addresses and subnet masks, VLAN IDs, UCS drivers to be installed, and iSCSI IP addresses, if defined.

As for the specific items configured during the ESXi host configuration task, these steps are run:

- Add the NTP servers
- Modify the ESXi vSwitch
- Create the in-band management portgroup
- Set the ESXi host power management policy
- Upgrade the UCS drivers
- Create iSCSI vSwitches (if defined)
- Create iSCSI portgroups (if defined)

- Create iSCSI vmkernel interfaces (if defined)
- Rescan the iSCSI vHBAs (if defined)

Once the ESXi hosts are built, then vCenter can be deployed on the hosts for management. After vCenter is deployed, we then run some basic configuration steps:

- Create a datacenter
- Create a cluster
- Enable DRS on the cluster
- Enable HA on the cluster
- Create a distributed virtual switch
- Create a vDS portgroup

Once these constructs are complete, we can finalize our configuration to place the hosts into vCenter for management with these last steps:

- Add the ESXi hosts to vCenter and the cluster
- Add the ESXi hosts to the vDS
- Create a vMotion vmkernel interface

After all of these configuration tasks have run, we have completed the virtual environment deployment and can then provision VMs for workloads to be run.

Conclusion

FlashStack is a platform built upon Cisco UCS and Pure Storage, which can include ethernet or fiber-channel networking. One of the most significant components in how both Pure & Cisco manage this hardware is that they are both built upon an API utilized to drive the configuration and operations of these platforms. The basis of an API with full coverage for all functionality of the Pure & Cisco components makes it very easy to automate the operations of that platform, which is one of the benefits of the FlashStack platform for many customers. This is critically important for the automated deployment and automation of the platform, including ongoing operations.

As work to manage operations on-prem and in the cloud becomes increasingly challenging at scale, empowering IT teams and developers to consume resources more easily becomes a critical need. Infrastructure as Code (IaC) is a primary means to simplify these functions, and this is the need that we are working to address this need with our new FlashStack IaC-focused guides and code that we are producing.

About the Author

Joe Houghes is a Senior Solutions Architect in the Portfolio Solutions team within Pure Storage, focused on solutions on the FlashStack platform and automation and integration. He has experience from over 20 years in Information Technology across various customer/vendor organizations with architecture and operations. His expertise covers computing, networking, storage, virtualization, business continuity and disaster recovery, cloud computing technologies, and automation and integration across many applications and vendor platforms.

Pure Storage, Inc.
650 Castro Street, #400
Mountain View, CA 94041

[Publication number and date here]