

FLASHSTACK CI FOR SPLUNK REFERENCE ARCHITECTURE

A Framework for Deploying Splunk® Enterprise at Scale

June 2017



TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
GOALS AND OBJECTIVES	3
AUDIENCE	4
REFERENCE ARCHITECTURE DESIGN PRINCIPLES	4
FLASHSTACK INTRODUCTION	5
TECHNOLOGY OVERVIEW	6
PURE STORAGE FLASHARRAY//M	6
FLASHARRAY//M TECHNICAL SPECIFICATIONS	7
PURITY OPERATING ENVIRONMENT	7
PURE1	8
EXPERIENCE EVERGREEN STORAGE	8
CISCO UNIFIED COMPUTING SYSTEM	8
RED HAT ENTERPRISE LINUX 7	11
SPLUNK	11
SPLUNK AND CISCO UCS – BENEFITS	12
SPLUNK ENTERPRISE	13
SPLUNK ARCHITECTURE	13
HOW THE INDEXER STORES DATA	14
STORAGE REQUIREMENTS IN A CLUSTERED ENVIRONMENT	16
SOLUTION DESIGN	18
DESIGN TOPOLOGY	18
PHYSICAL TOPOLOGY	19
LOGICAL CONFIGURATION	23
TEST CONFIGURATION	24
TEST OVERVIEW	24
TEST SETUP	24
TEST RESULTS	27
PERFORMANCE TESTS	28
TEST RESULTS	28
SHARED STORAGE BENEFITS	30

BEST PRACTICES FOR SPLUNK ON PURE FLASHARRAY	31
CONCLUSION	33
REFERENCES	33
APPENDIX A: SPLUNK COMPONENTS	34
APPENDIX B: RAPID DEPLOYMENT OF SPLUNK HOSTS	35
APPENDIX C: CLUSTER SHELL UTILITY INSTALLATION	45
APPENDIX D: SPLUNK VOLUMES SETUP	50
APPENDIX E: UCS CLI TO START AND SHUTDOWN SERVERS	53
ABOUT THE AUTHOR	54

DOCUMENT HISTORY

Version	Changes
1.0 (June 2017)	Initial release

EXECUTIVE SUMMARY

Splunk® has become a mission critical application. Thousands of organizations are gaining insight from their machine data and transaction logs using Splunk, and many more are planning to deploy Splunk. No matter what stage you're in, having guidelines to follow can help improve the Splunk experience. Since a mission critical data application deserves a mission critical data platform, Pure Storage® built the solution on the Pure FlashStack™ converged infrastructure solution. FlashStack is a joint offering from Cisco® and Pure Storage.

This paper is intended to provide a framework for designing and sizing a high-performance, scalable, and resilient Splunk platform. Pure Storage is a leading all-flash array provider focused on reducing storage complexity while improving Splunk performance, resiliency, and efficiency.

To assure that your Splunk platform is sized appropriately, Pure Storage tested Splunk Enterprise on Pure's FlashStack platform.

The top takeaways from these efforts are that FlashStack for Splunk can:

- Lower management costs by up to 90%
- Reduce data center footprint, power, and cooling by up to 10x compared to legacy storage solutions

FlashStack for Splunk describes architecture and deployment procedures for Splunk Enterprise at scale based on a distributed deployment model. This reference architecture also provides guidelines for right-sizing Splunk storage requirements. We detail configuration of the hardware and software components involved, provide various testing results, and offer implementation and best practices guidance.

GOALS AND OBJECTIVES

The goal of this reference architecture is to showcase the scalability, performance, manageability, and simplicity of the Pure FlashStack solution for deploying Splunk Enterprise at scale. One of the challenges of an enterprise grade Splunk deployment is the tendency to try to fit the application into existing infrastructure during the “pilot” phase. As the benefits of Splunk start to take hold within the organization, this “pilot” deployment model can quickly become undersized in terms scale and performance.

Whether you are an experienced Splunk ninja who is looking to build a truly robust infrastructure or new to Splunk, you want to design a solution that will not go through peaks and valleys of performance and scale as Splunk takes hold in your organization. Following are the goals of this reference architecture:

- Provide a design guide for setting up Splunk Enterprise on FlashStack
- Demonstrate seamless capacity growth to support Splunk Enterprise
- Detail storage sizing guidelines and best practices

AUDIENCE

The target audience for this document includes, but is not limited to, system administrators, storage administrators, IT managers, system architects, sales engineers, field consultants, professional services, and partners who are looking to design and deploy Splunk Enterprise on a FlashStack platform. A working knowledge of Splunk, Linux®, server, storage, and networks is assumed but is not a prerequisite to read this document.

REFERENCE ARCHITECTURE DESIGN PRINCIPLES

The guiding principles for implementing this reference architecture are:

Simple – Avoid unnecessary and/or complex configurations or techniques that make the results look better than a normal out-of-box environment.

Repeatable – Create a scalable building block that can be easily replicated at any customer site. Publish the versions of firmware under test and uncover any issues in the lab before customers deploy the solution.

Available – Complement Splunk high availability architecture with Pure's non-disruptive upgrade (NDU) capability.

Efficient – Take advantage of Splunk's power while leveraging Pure's low latency, high IOPS, and throughput, space, power, and cooling efficiencies.

Scalable – Linear scalability of Splunk deployment within the FlashStack architecture using best-in-class flash storage.

FLASHSTACK INTRODUCTION

The Pure FlashStack solution is a converged infrastructure (CI) solution that delivers the benefits of fully-tested converged infrastructure built on best-of-breed components from Cisco and Pure Storage. FlashStack provides a converged infrastructure solution that is simple, flexible, efficient, and costs less than legacy converged infrastructure solutions based on traditional disk.

The FlashStack CI solution is available from accredited FlashStack Partners who help provide an excellent converged infrastructure ownership experience. FlashStack Partners have the knowledge and experience necessary to help streamline the sizing, procurement, and delivery of your entire system.

Key Benefits of the FlashStack solution are:

1. Consistent Performance and Scalability

- Consistent sub-millisecond latency with 100% flash storage

- Consolidate 100s of enterprise-class applications in a single rack

- Scale easily, without disruption

- Repeatable growth through multiple FlashStack CI deployments

2. Operational Simplicity

- Fully tested, validated, and documented for rapid deployment

- Reduced management complexity

- No storage tuning or tiers necessary

- Auto-aligned 512b architecture eliminates storage alignment headaches

3. Lowest TCO

- Dramatic savings in power, cooling, and space with 100% Flash

- Industry leading data reduction

- FlashArray controller upgrades included with maintenance every three years via the Pure Storage Evergreen™ program

4. Enterprise Grade Resiliency

- Highly available architecture and redundant components

- Non-disruptive operations

- Upgrade and expand without downtime or performance loss

- Native data protection: snapshots and replication



Figure 1. FlashStack

TECHNOLOGY OVERVIEW

The IT industry has been transforming rapidly onto converged infrastructure, which enables faster provisioning, scalability, lower data center (DC) costs, simpler management infrastructure, and future proofing against technology advancement. The Pure FlashStack solution provides best-of-breed technology to reap the benefits of CI. This section details the various infrastructure components that are part of the FlashStack solution.

PURE STORAGE FLASHARRAY//M

FlashArray//M expands upon the FlashArray's modular, stateless architecture, designed to enable expandability and upgradability for multiple product generations. FlashArray//M leverages a chassis-based design with customizable modules that enable both capacity and performance to be improved independently over time with advances in compute and flash – meeting your business needs today and tomorrow. FlashArray//M delivers the following:

- **Consistent Performance** – FlashArray delivers consistent <1ms average latency. Performance is optimized for real-world applications workloads that are dominated by I/O sizes of 32K or larger vs. 4K/8K hero performance benchmarks. Full performance is maintained even under failures/updates.
- **Lower Cost than Disk** – Inline de-duplication and compression deliver 2 – 10x greater space savings across a broad set of I/O workloads including databases, virtual machines, and virtual desktop infrastructure. With VDI workloads data reduction is typically > 10:1.
- **Mission-Critical Resiliency** – FlashArray//M has proven >99.9999% delivered availability, as measured across the Pure Storage installed base, inclusive of non-disruptive upgrades and maintenance, and without performance impact.
- **Disaster Recovery Built-In** – FlashArray offers native, fully-integrated, data reduction-optimized backup and disaster recovery at no additional cost. Set-up disaster recovery with policy-based automation within minutes. Recover instantly from local, space-efficient snapshots or remote replicas.
- **Simplicity Built-In** – FlashArray offers game-changing management simplicity that makes storage installation, configuration, provisioning, and migration a snap. No more managing performance, RAID, tiers, or caching. Achieve optimal application performance without any tuning at any layer. Manage FlashArray the way you like it: Web-based GUI, CLI, VMware vCenter, Windows PowerShell, Python, REST API, or OpenStack.

FLASHARRAY//M TECHNICAL SPECIFICATIONS



	//M10*	//M20*	//M50*	//M70*
Capacity	Up to 25TBs effective capacity* 5 – 10TBs raw capacity	Up to 250+TBs effective capacity* 5 – 80TBs raw capacity	Up to 500+TBs effective capacity* 20 – 176TBs raw capacity	Up to 1.5PBs effective capacity* 42 – 512TBs raw capacity
All-Inclusive Performance**	Up to 100,000 32K IOPS† <1ms average latency Up to 3 GB/s bandwidth††	Up to 200,000 32K IOPS† <1ms average latency Up to 6 GB/s bandwidth††	Up to 270,000 32K IOPS† <1ms average latency Up to 9 GB/s bandwidth††	Up to 370,000 32K IOPS† <1ms average latency Up to 11.5 GB/s bandwidth††
Connectivity	16 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI 1 Gb/s Management & Replication ports	16 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI 10 Gb/s Replication ports 1 Gb/s Management ports	16 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI 10 Gb/s Replication ports 1 Gb/s Management ports	16 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI 10 Gb/s Replication ports 1 Gb/s Management ports
Physical	3U 575 – 625 Watts (nominal – peak) 95 lbs (43.1 kg) 5.12" x 18.94" x 29.72" chassis	3U – 5U 600 – 950 Watts (nominal – peak) 95 lbs (43.1 kg) fully loaded 5.12" x 18.94" x 29.72" chassis	3U – 5U 650 – 1280 Watts (nominal – peak) 95 lbs (43.1 kg) fully loaded + 44 lbs per expansion shelf 5.12" x 18.94" x 29.72" chassis	5U – 7U 1230 – 1760 Watts (nominal – peak) 97 lbs (44.0 kg) fully loaded + 44 lbs per expansion shelf 5.12" x 18.94" x 29.72" chassis

* Stated specifications are applicable to //M R2 versions.

** Effective capacity assumes HA, RAID, and metadata overhead, GB-to-GiB conversion, and includes the benefit of data reduction with always-on inline deduplication, compression, and pattern removal. Average data reduction is calculated at 5-to-1.

*** Includes always-on RAID-3D, deduplication, compression, and encryption.

† Why does Pure Storage quote 32K, not 4K IOPS? The industry commonly markets 4K IOPS benchmarks to inflate performance numbers, but multiple real world workloads consolidating on a single array average closer to 32K. FlashArray adapts automatically to 512B – 32KB IO for superior performance, scalability, and data reduction. IOPS are based on 100% 32KiB reads.

†† Data throughput is based on 100% 32KiB reads.

Figure 2. FlashArray//M technical specifications

PURITY OPERATING ENVIRONMENT

Purity implements advanced data reduction, storage management, and flash management features. All features of Purity are included in the base cost of FlashArray//M.

Storage Software Built for Flash – Pure's FlashCare technology virtualizes the entire pool of flash within the FlashArray, and allows Purity to both extend the life and ensure the maximum performance of consumer-grade MLC flash.

Granular and Adaptive – Purity Core is based upon a 512-byte variable block size metadata layer. This fine-grained metadata enables all of Purity’s data and flash management services to operate at the highest efficiency.

Best Data Reduction Available – FlashReduce implements five forms of inline and post-process data reduction to offer the most complete data reduction in the industry. Data reduction operates at a 512-byte aligned variable block size, to enable effective reduction across a wide range of mixed workloads without tuning.

Highly Available and Resilient – FlashProtect implements high availability, dual-parity RAID-3D, non-disruptive upgrades, and encryption, all of which are designed to deliver full performance to FlashArray during any failure or maintenance event.

Backup and Disaster Recovery Built In – FlashRecover combines space-saving snapshots, replication, and protection policies into an end-to-end data protection and recovery solution that protects data against loss locally and globally. All FlashProtect services are fully integrated in FlashArray and leverage native data reduction capabilities.



Pure1® Manage – By combining local web-based management with cloud-based monitoring, Pure1 Manage allows you to manage your FlashArray wherever you are – with just a web browser.

Pure1 Connect – A rich set of APIs, plugins, application connectors, and automation toolkits enable you to connect FlashArray//M to all your data center and cloud monitoring, management, and orchestration tools.

Pure1 Support – FlashArray//M is constantly cloud-connected, enabling Pure Storage to deliver the most proactive support experience possible. Highly trained staff combined with big data analytics help resolve problems before they start.

Pure1 Collaborate – Extend your development and support experience online, leveraging the Pure1 Collaborate community to get peer-based support and to share tips, tricks, and scripts.

EXPERIENCE EVERGREEN STORAGE

Tired of the 3-5-year array replacement merry-go-round? The move to FlashArray//M can be your last data migration. Purchase and deploy storage once and once only – then expand capacity and performance incrementally in conjunction with your business needs, and without downtime. Pure’s vision for Evergreen Storage is delivered by a combination of FlashArray’s stateless, modular architecture and the Evergreen business model, enabling you to extend the lifecycle of storage from 3-5 years to a decade or more.

CISCO UNIFIED COMPUTING SYSTEM

The Cisco® Unified Computing System is a next-generation data center platform that unites compute, network, storage access, and virtualization into a cohesive system designed to reduce total cost of ownership (TCO) and increase business agility. The system integrates a low-latency, lossless 10- and 40-Gigabit Ethernet unified network

fabric with enterprise-class, x86-architecture servers. The system is an integrated, scalable, multi-chassis platform in which all resources participate in a unified management domain. Cisco UCS is a next-generation solution for blade and rack server computing.

Cisco UCS unites the following main components:

1. Computing

Based on an entirely new class of computing system that incorporates rack mount and blade servers based on Intel Xeon Processors E5 and E7, the Cisco UCS Servers offer Cisco Extended Memory Technology to support applications with large datasets and allow more virtual machines per server.

2. Network

The system is integrated onto a low-latency, lossless, 10 and 40-Gbps unified network fabric. This network foundation consolidates LAN, SAN, and high-performance computing networks, which are separate networks today. The unified fabric lowers costs by reducing the number of network adapters, switches, and cables, and by decreasing power and cooling requirements.

3. Virtualization

The system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtualized environments to better support changing business and IT requirements.

4. Storage Access

The system provides consolidated access to both SAN storage and Network Attached Storage (NAS) over the unified fabric. By unifying storage access, the Cisco Unified Computing System can access storage over Ethernet (NFS or iSCSI) and Fibre Channel over Ethernet (FCoE). This provides customers with choice for storage access and investment protection. In addition, server administrators can pre-assign storage access policies for system connectivity to storage resources, simplifying storage connectivity and management for increased productivity.

5. Management

Cisco UCS uniquely integrates all system components to enable the entire solution to be managed as a single entity by the Cisco UCS Manager. The Cisco UCS Manager has an intuitive graphical user interface (GUI), a command-line interface (CLI), and a powerful scripting library module for Microsoft® PowerShell, built on a robust application programming interface (API) to manage all system configuration and operations.

CISCO UCS MANAGER

Cisco UCS Manager resides within the Cisco UCS 6300 Series Fabric Interconnect. It makes the system self-aware and self-integrating, managing all of the system components as a single logical entity. Cisco UCS Manager uses service profiles to define the personality, configuration, and connectivity of all resources within Cisco UCS,

radically simplifying provisioning of resources so the process takes minutes instead of days. This simplification allows IT departments to shift their focus from constant maintenance to strategic business initiatives like Splunk Enterprise.

Some of the key elements managed by Cisco UCS Manager include:

- **Cisco UCS Integrated Management Controller (IMC)** firmware
- **RAID controller firmware** and settings
- **BIOS firmware and settings**, including server universal user ID (UUID) and boot order
- **Converged network adapter (CNA)** firmware and settings, including MAC addresses and worldwide names (WWNs) and SAN boot settings
- **Virtual port groups** used by virtual machines, using Cisco Data Center VM-FEX technology
- **Interconnect configuration**, including uplink and downlink definitions, MAC address and WWN pinning, VLANs, VSANs, quality of service (QoS), bandwidth allocations, Cisco Data Center VM-FEX settings, and Ether Channels to upstream LAN switches

GREATER TIME-ON-TASK EFFICIENCY

Automated configuration can change an IT organization's approach from reactive to proactive. The result is more time for innovation, less time spent on maintenance, and faster response times. These efficiencies allow IT staff more time to address strategic business initiatives. They also enable better quality of life for IT staff, which means higher morale and better staff retention – both critical elements for long-term efficiency.

Cisco UCS Manager is an embedded, model-based management system that allows IT administrators to set a vast range of server configuration policies, from firmware and BIOS settings to network and storage connectivity. Individual servers can be deployed in less time and with fewer steps than in traditional environments. Automation frees staff from tedious, repetitive, time-consuming chores which are often the source of errors that cause downtime, making the entire data center more cost effective.

EASIER SCALING

Automation means rapid deployment, reduced opportunity cost, and better capital resource utilization. With Cisco UCS, rack-mount and blade servers can move from the loading dock and into production in a “plug-and-play” operation. Automatically configure blade servers using predefined policies simply by inserting the devices into an open blade chassis slot. Integrate rack-mount servers by connecting them to top-of-rack Cisco Nexus® fabric extenders. Since policies make configuration automated and repeatable, configuring 100 new servers is as straightforward as configuring one server, delivering agile, cost-effective scaling.

VIRTUAL BLADE CHASSIS

With a separate network and separate management for each chassis, traditional blade systems are functionally an accidental architecture based on an approach that compresses all the components of a rack into each and every chassis. Such traditional blade systems are managed with multiple management tools that are combined to give

the illusion of convergence for what is ultimately a more labor-intensive, error-prone, and costly delivery methodology. Rack-mount servers are not integrated and must be managed separately or through additional tool sets, adding complexity, overhead, and the burden of more time.

Architecturally, Cisco UCS blade and rack-mount servers are joined into a single virtual blade chassis that is centrally managed yet physically distributed across multiple blade chassis, rack-mount servers, and even racks and rows. This capability is delivered through Cisco fabric interconnects that provide redundant connectivity, a common management and networking interface, and enhanced flexibility. The larger virtual chassis, with a single redundant point of management, results in lower infrastructure cost per server, with fewer management touch points, and lower administration, capital, and operational costs.

SERVICE PROFILES

Cisco UCS resources are abstract in the sense that their identity, I/O configuration, MAC addresses and worldwide names (WWNs), firmware versions, BIOS boot order, and networking attributes (including quality of service (QoS) settings, pin groups, and threshold policies) are all programmable using a just-in-time deployment model. The manager stores this identity, connectivity, and configuration information in service profiles that reside on the Cisco UCS 6300 Series Fabric Interconnects. A service profile can be applied to any blade server to provision it with the characteristics required to support a specific software stack. A service profile allows server and network definitions to move within the management domain, enabling flexibility in the use of system resources. Service profile templates allow different classes of resources to be defined and applied to a number of resources, each with its own unique identities assigned from predetermined pools. Furthermore, service profiles can, at any time, be migrated from one physical server to another. This logical abstraction of the server personality separates the dependency of the hardware type or model and is a result of Cisco's unified fabric model.

RED HAT ENTERPRISE LINUX 7

Red Hat® Enterprise Linux 7 not only lays the foundation for the open hybrid cloud and serves enterprise workloads across converged infrastructures, it also pushes the operating system beyond today's position as a commodity platform. Built to meet modern datacenter demands along with next-generation IT requirements, Red Hat Enterprise Linux 7 powers the spectrum of enterprise IT, from application containers to cloud services.



SPLUNK

Splunk Inc. provides the leading platform for Operational Intelligence. Splunk® software searches, monitors, analyzes, and visualizes machine-generated big data from websites, applications, servers, networks, sensors, and mobile devices. More than 13,000 organizations use Splunk software to deepen business and customer understanding, mitigate cybersecurity risk, improve service performance, and reduce costs.



Splunk® Enterprise monitors and analyzes machine data from any source to deliver Operational Intelligence to optimize your IT, security, and business performance. With intuitive analysis features, machine learning, packaged applications, and open APIs, Splunk Enterprise is a flexible platform that scales from focused use cases to an enterprise-wide analytics backbone.

Splunk Enterprise provides an end-to-end, real-time solution for machine data, delivering the following core capabilities.

1. **Universal collection and indexing** of machine data, from virtually any source
2. **Powerful search processing language (SPL)** to search and analyze real-time and historical data
3. **Apps provide solutions** for security, IT Ops, business analysis, and more
4. **Real-time monitoring** for patterns and thresholds; real-time alerts when specific conditions arise
5. **Powerful reporting** and analysis
6. **Custom dashboards** and views for different roles
7. **Resilience** and horizontal scalability
8. **Granular role-based security** and access controls
9. **Support for multi-tenancy** and flexible, distributed deployments on-premises or in the cloud
10. **Robust, flexible platform** for big data apps

SPLUNK AND CISCO UCS – BENEFITS

AVAILABILITY

- UCS failover capabilities protect against common hardware failures, which is a key requirement for Splunk Enterprise especially in a distributed environment

SCALABILITY/FLEXIBILITY

- Capacity on demand and dynamic resource allocation
- Stateless blades enable rapid provisioning of nodes

MANAGEABILITY

- Single management interface to manage the hardware

PERFORMANCE

- Improved interconnect and Cache Fusion performance
- Fast, low latency, and lossless 10 Gb Ethernet enables large interconnect between Splunk indexers for replication

SPLUNK ENTERPRISE

Splunk Enterprise makes it simple to collect, analyze, and act upon the value of the big data generated by your technology infrastructure, security systems, and business applications – giving you the insights to drive operational performance and business results. With Splunk Enterprise, everyone from data or security analysts to business users can gain powerful insights across multiple use cases like security, IT operations, application delivery, industrial data, and IoT.

SPLUNK ARCHITECTURE

Splunk software is designed in such a way that a single installation file can be configured to function as one or all of the various Splunk components like Indexer, Search Head, Deployer, Cluster Master, etc. Splunk can be deployed in Single instance or Distributed mode. In the single-instance deployment, one instance of Splunk Enterprise performs all aspects of data management, from input to indexing to searching. Distributed Splunk deployment, on the other hand, scales your deployment by distributing Splunk Enterprise instances across multiple machines, where each machine can be configured to perform a specific task.

In general, Splunk Enterprise performs three key functions as part of data processing, namely data ingestion, indexing, and search. To scale the environment, these three functionalities can be split across multiple nodes of Splunk Enterprise, which are typical tiers of the distributed deployments. The following diagram illustrates the three tiers of processing.

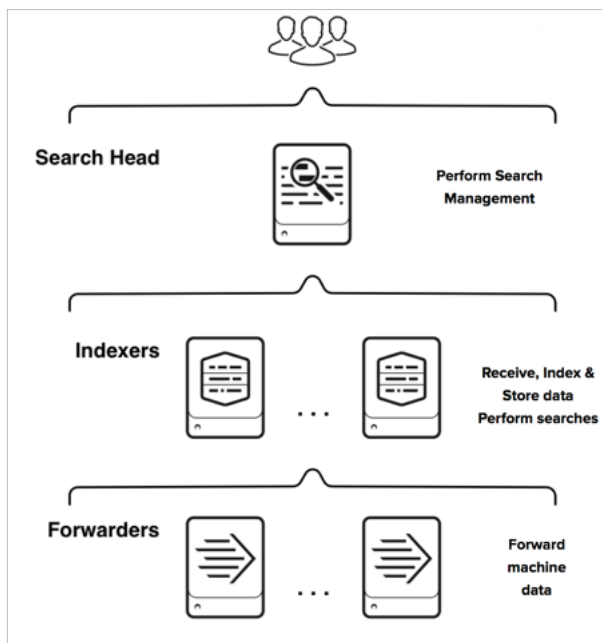


Figure 3. Three tiers of key function

To scale the system, you can add more components to each tier. To meet high availability requirements, enable horizontal scaling of searches, and improve ease of management, components can be clustered.

Figure 4 provides a high-level overview of the Splunk system architecture. This reference architecture is built using both Indexer Cluster and Search Head Cluster to support high availability, horizontal scaling, and ease of management. For more information on how to scale your deployment with Splunk Enterprise components, please see the Splunk documentation online¹.

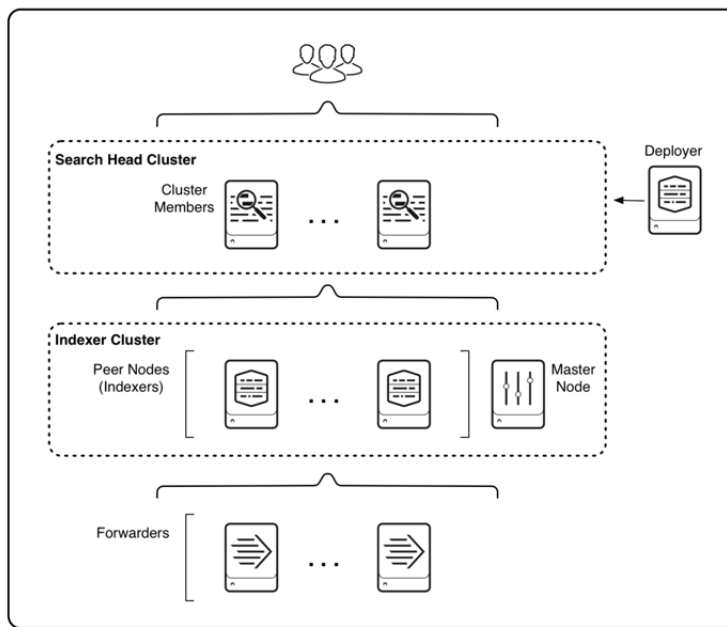


Figure 4. Splunk system with cluster architecture

HOW THE INDEXER STORES DATA

Splunk **Indexer** indexes incoming data by transforming raw data into events based on timestamps and stores them in index files. Indexer also services search requests from the Search Head and passes the search results back. Indexer is the primary location where the Splunk index data is stored in “buckets”, which are nothing but directories on the server.

As the Indexer indexes the machine data, it creates a number of flat files. These files contain two types of data:

- Compressed rawdata, estimated² at 15% of the size of the incoming data
- Index files or tsidx files (indexes that point to the raw data), estimated¹ at 35% of the size of the incoming data

Note: These reduction rates were based on syslog-type input data.

The reduction rate can vary substantially based on various factors, including

¹ <https://docs.splunk.com/Documentation/Splunk/6.5.3/Deploy/Distributedoverview>

² <http://docs.splunk.com/Documentation/Splunk/6.1.2/Indexer/Systemrequirements>

the input data type. Please see Splunk's documentation "Estimating your storage requirements" for more details³.

The following figure illustrates how Splunk stores the data on an indexer for syslog type data. Overall, Splunk consumes as much as 50% of the ingested capacity for its compressed rawdata and index and meta data files.

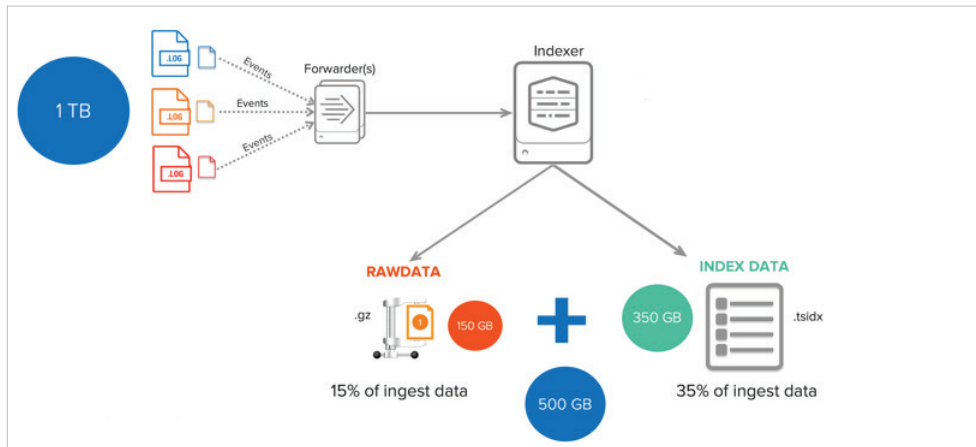


Figure 5. How Splunk stores data

Splunk stores all its data in buckets on the Indexer nodes. **Buckets** are simply index directories on the indexer nodes. A Bucket moves through various stages as it ages, via Hot, Warm, Cold, Frozen, Thawed buckets.

Buckets roll from one stage to the next as they age. When data is indexed, it lands in a hot bucket which is both searchable and actively being written into. Hot is the only bucket that can be written to. When certain conditions are met, like a hot bucket reaches a certain size, the hot bucket rolls into a warm bucket, and a new hot bucket is created in its place. Warm buckets are searchable, but cannot be actively written into. When further conditions are met, like index reaches the maximum number of warm buckets, the indexer rolls the warm buckets to cold, based on their age. The oldest warm buckets are the candidates to be rolled to the cold bucket. After a configured period of time, cold buckets roll to frozen buckets which can either be archived or deleted. Archived frozen data can be thawed as needed and is available for searches.

³ <http://docs.splunk.com/Documentation/Splunk/6.1.2/Installation/Estimateyourstoragerequirements>

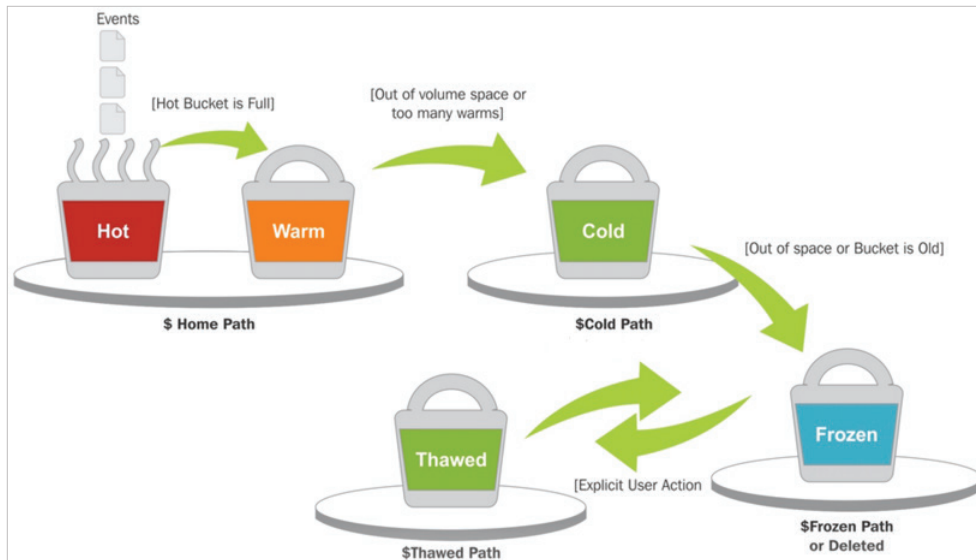


Figure 6. Buckets lifecycle

STORAGE REQUIREMENTS IN A CLUSTERED ENVIRONMENT

Splunk offers Indexer Clusters to prevent data loss while promoting data availability for searching by index replication, whereby Splunk keeps multiple copies of incoming data. While index replication provides the benefits of data availability, it comes with the cost of additional storage to store the replicated data, along with a smaller degree of processing load on indexers, and additional network traffic to replicate the data between indexers.

There are two key elements named *replication factor* and *search factor* that determine the data availability requirement either for fault-tolerance or search:

- **Replication Factor (RF)**

- Determines the indexer cluster's failure tolerance
- Determines the number of copies of data that the indexer cluster maintains
- Cluster can tolerate a failure of (RF -1) peer nodes
- Contains the rawdata in compressed format

- **Search Factor (SF)**

- Determines the number of searchable copies of data the indexer cluster maintains
- Logical copy on peer nodes as the index data is generated locally on the peer node based on the rawdata that was replicated
- Increased search factor doesn't mean improved search performance, rather improved search availability
- Recommended and default value is two (2)

In an indexer cluster environment, the total storage requirement is dependent on the replication factor and the search factor. The following figures illustrate the storage requirement in an indexer cluster environment with varying RF and SF.

REPLICATION FACTOR = 3, SEARCH FACTOR = 2, INGEST DATA = 1 TB

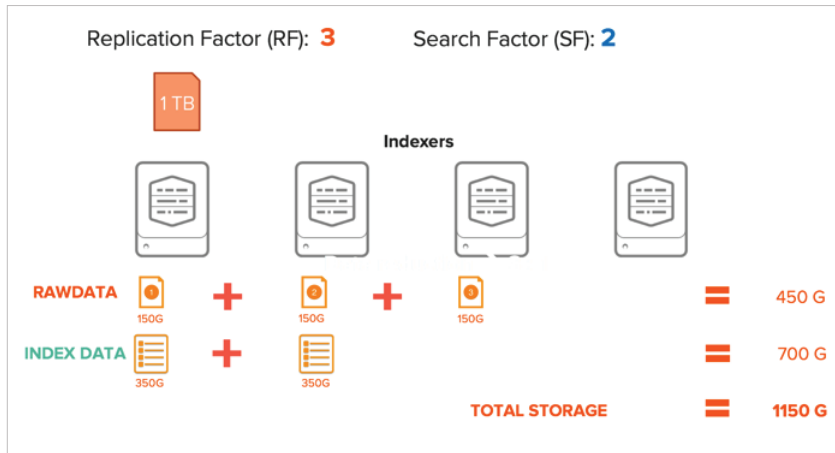


Figure 7. Storage requirement for RF-3, SF-2

REPLICATION FACTOR = 4, SEARCH FACTOR = 2, INGEST DATA = 1 TB

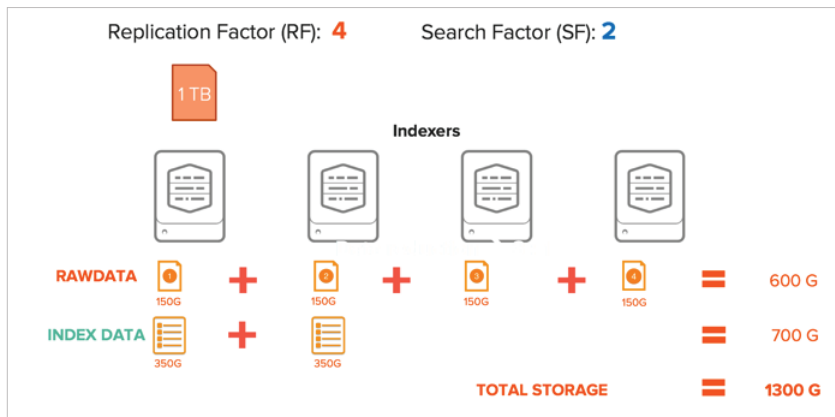


Figure 8. Storage requirement for RF-4, SF-2

The above is an illustration showing the storage requirements for 1 TB of ingest data for a single day. In reality, you would need the above storage multiplied by the retention period to get the overall storage requirements. Also, the actual reduction might vary based on your machine logs.

SOLUTION DESIGN

DESIGN TOPOLOGY

This section describes design considerations for the FlashStack for Splunk Enterprise deployment. In this solution design, we have used two chassis with 13x Intel CPU-based Cisco UCS B-series B200 M4 blade servers for hosting the Splunk Enterprise, which comprises 8 indexers, 3 search heads, two admin servers, and Hot/Warm and Cold buckets for the indexers provisioned out of Pure FlashArray//M70.

The server has Cisco UCS VIC 1340; they were connected to the four ports from each Cisco Fabric extender of the Cisco UCS chassis to the Cisco Fabric Interconnect, which in turn was connected to the Cisco MDS 9148S for upstream connectivity to access the Pure Storage FlashArray//M LUNs. The hardware configuration is described in Table 1.

UCS HARDWARE CONFIGURATION

Component	Description
Indexer	8 Cisco UCS B200-M4 server blades each with: • 2 Intel Xeon processor E5-2670 v3 CPUs (24 cores) • 512 GB of memory
Search Head	3 Cisco UCS B200-M4 server blades each with: • 2 Intel Xeon processor E5-2630 v4 CPUs (20 cores) • 256 GB of memory
Administration Nodes	2 Cisco UCS B200-M4 server blades each with: • 2 Intel Xeon processor E5-2630 v4 CPUs (20 cores) • 256 GB of memory
Networking	2 Cisco UCS 6332 UP 16-Port Fabric Interconnects 2 Cisco MDS 9148S fabric 16G FC Switches 2 Cisco Nexus 9372PX Ethernet Switches
VIC (Virtual Interface Card) 1340	40 Gbps Unified I/O ports on Cisco UCS VIC 1380
Chassis	2 x Cisco UCS Chassis 5108 (6RU each)

Table 1. UCS hardware configuration

FLASHARRAY CONFIGURATION

The FlashStack design is comprised of FlashArray//M70 for increased capacity, scalability, and throughput. Even though this reference architecture is built with 79.10TB of raw space, FlashArray//M70 can support raw storage of up to 512TB. The table below shows the components of the array.

There are no special configurations or performance knobs to tune on the FlashArray. The hosts are redundantly connected to the controllers with 6 connections to each controller from four redundant HBAs on each host over the FC protocol for a total of twenty-four logical paths. Zoning was performed on the Cisco MDS 9148S switches to allow Pure Storage FlashArray//M to see the initiators.

Component	Description
FlashArray	//M70
Capacity	79.10 TB raw (base chassis + additional shelf with 2x44 TB data packs) 49.13 TB usable
Connectivity	12 x 16 Gb/s redundant Fibre Channel ports 1 Gb/s redundant Ethernet (Management port)
Physical	5U 5.12" x 18.94" x 29.72" FlashArray//M chassis

Table 2. FlashArray //M70 configuration

OS AND SOFTWARE

Operating System and Software	Description
Linux	Red Hat Enterprise Linux Server 7.2 (64 bit) 3.10.0-327.el7.x86_64
Splunk	Splunk 6.5.1 (build f74036626f0c)
Cisco UCS Manager	3.1 (2b)
Cisco MDS 9148S System Version	7.3(0)D1(1)
Cisco Nexus 9372PX NXOS Version	7.0(3)I2(3)
Pure Storage Purity	4.8.8

Table 3. OS and software

PHYSICAL TOPOLOGY

Pure Storage FlashStack consists of a combined stack of hardware (storage, network, and compute) and software (Cisco UCS Manager, Splunk Enterprise, Purity, and Red Hat Enterprise Linux).

Considering an Enterprise-grade deployment of Splunk that allows both a large amount of machine data to be ingested quickly as well as a large number of concurrent searches, the reference architecture of FlashStack for Splunk is made up of a distributed configuration with eight (8) Cisco UCS B200-M4 blade servers as indexers, three (3) Cisco UCS B200-M4 blade servers as search heads, and two (2) Cisco UCS B200-M4 blade servers as administrative servers supporting License Manager, Deployer, Master Node, and Deployment server. Figure 6 shows the architecture of the FlashStack for Splunk Enterprise deployment.

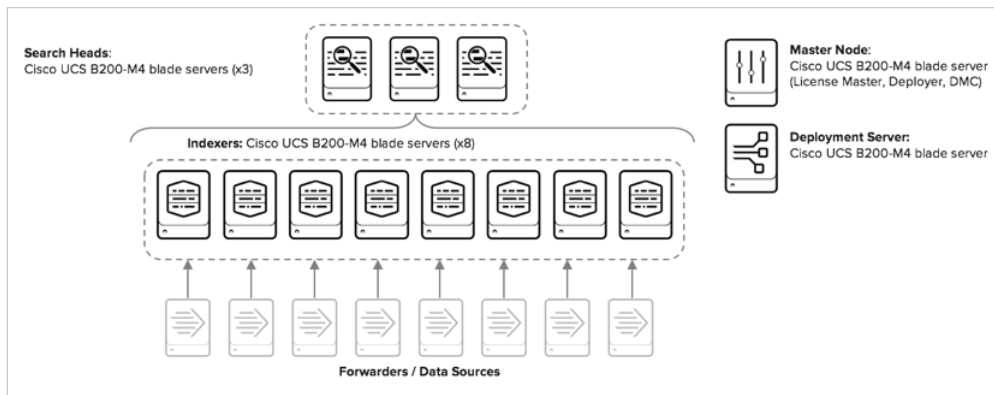


Figure 9. Distributed deployment architecture of Splunk Enterprise

For testing purposes, we used Cisco UCS C240-M4 rack servers to set up the Virtual environment, where Linux VMs were configured as Forwarders. In your environment, you would configure forwarders which require minimal resources and usually reside on the machines where the data originates. Figure 10 shows the physical architecture of FlashStack for Splunk.

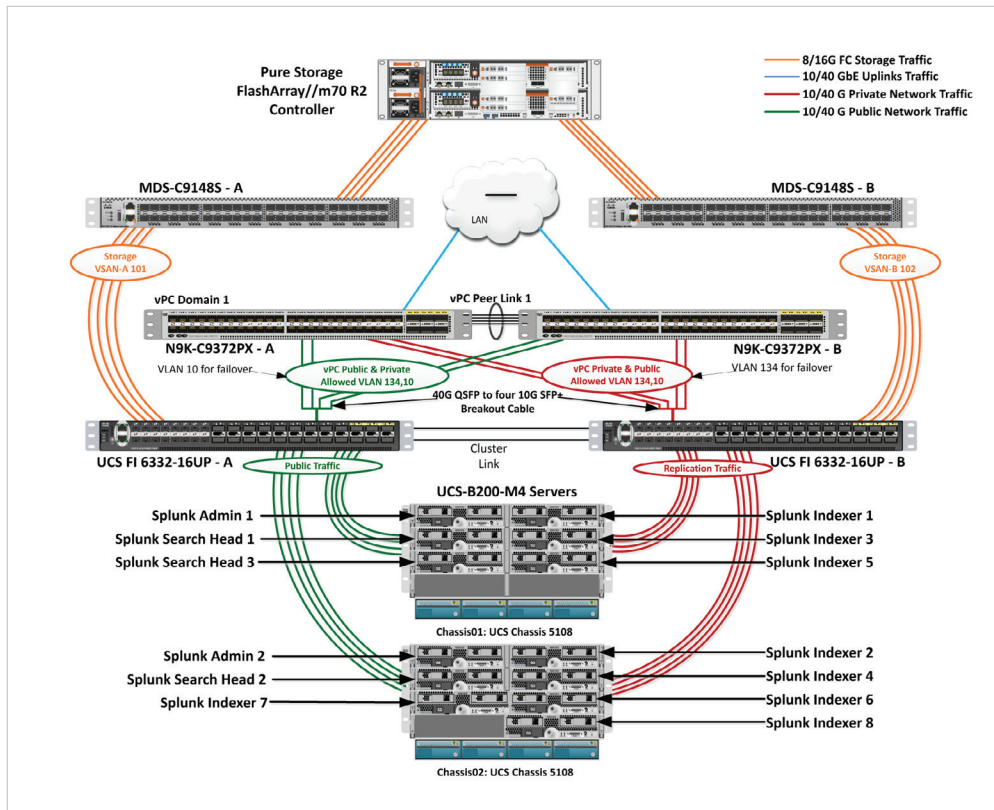


Figure 10. FlashStack Converged Infrastructure for Splunk

Figure 10 is a typical network configuration that can be deployed in a customer's environment. The best practices and setup recommendations are described later in this document.

As shown in Figure 10, a pair of Cisco UCS 6332-16UP fabric interconnects carries both storage and network traffic from the blades with the help of Cisco Nexus 9372PX and Cisco MDS 9148S switches. Both the fabric interconnect and the Cisco Nexus switch are clustered with the peer link between them to provide high availability. Two virtual Port-channels (vPCs) are configured to provide public network and private/replication/search network paths for the blades to northbound switches. Each vPC has VLANs created for application network data and management data paths.

Eight (four per chassis) links go to Fabric Interconnect A. Similarly, eight links go to Fabric Interconnect B. FC Storage access from Fabric Interconnect A & B are shown with an orange line.

For Splunk Enterprise with distributed clustering deployment, we recommend keeping all private traffic (replication and search) local on a single Fabric Interconnect. In this case, the private traffic will stay local to the fabric interconnect and will not be routed via a northbound network switch, meaning all inter-blade communication will be resolved locally at the fabric interconnect, which can significantly reduce latency.

It is beyond the scope of this document to cover detailed information regarding a UCS infrastructure setup, connectivity, and configuration. Documentation guides and examples are available at <http://www.cisco.com/c/en/us/support/servers-unified-computing/ucs-manager/products-installation-and-configuration-guides-list.html>

Following are the high-level steps involved for a Cisco UCS configuration:

- Configure Fabric Interconnects for a Cluster setup
- Configure Fabric Interconnects for Chassis and Blade Discovery
 - Configure Global Policies
 - Configure Server Ports
- Configure LAN and SAN on Cisco UCS Manager
 - Configure Ethernet LAN Uplink Ports
 - Configure FC SAN Uplink Ports
 - Configure VLAN
 - Configure VSAN
- Configure UUID, IP, MAC, WWNN and WWPN Pools
 - UUID Pool creation
 - IP and MAC Pool creation
 - WWNN and WWPN Pool creation
- Configure vNIC and vHBA Template
 - Create Public vNIC template
 - Create Private vNIC template
 - Create Storage vHBA template
- Configure Ethernet Uplink Port-Channels
- Create Server Boot Policy for SAN boot

LOGICAL CONFIGURATION

Figure 11 shows the logical architecture of the Splunk Enterprise setup used in this reference architecture. This is a distributed deployment model with clustering enabled for both Indexers and Search Heads, to enable scalable and highly available indexers and search heads.

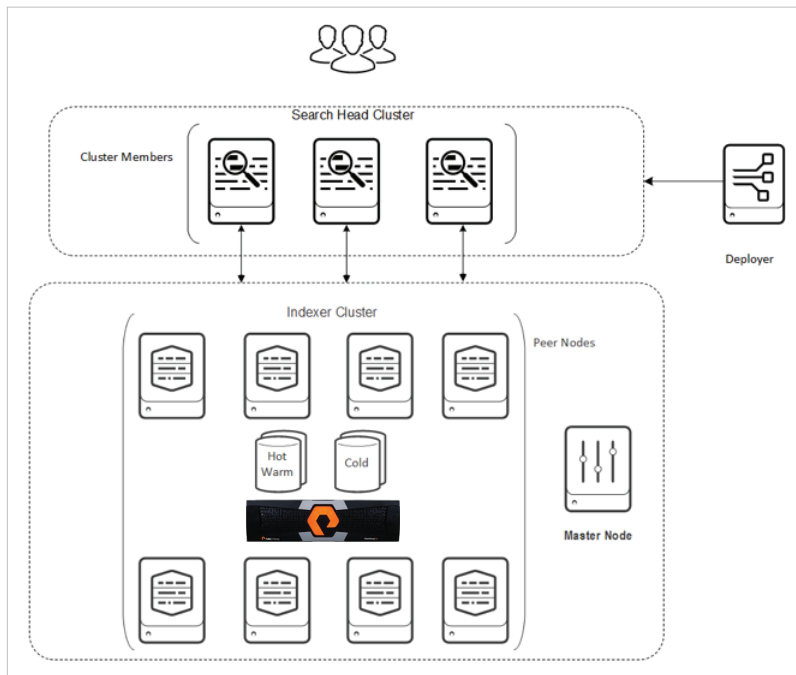


Figure 11. Logical architecture

The Splunk Enterprise software was installed as part of the gold image of the first server. Given the flexibility of Splunk, which is designed to take up any functions based on the deployment model, **splunk-admin1** was configured as the License Manager, Cluster Master node. **splunk-indx01** to **splunk-indx08** were configured as Indexers and added to an Indexer Cluster. Similarly, **splunk-srch01** to **splunk-srch03** were configured as Search Heads and added to a Search Head Cluster. The server **splunk-admin1** was also configured as the Deployer to deploy configuration changes to the search head clusters.

TEST CONFIGURATION

TEST OVERVIEW

Load 1 TB of machine data onto the 8 indexers that have buckets configured on the Pure Storage FlashArray with replication factor 3 and search factor 2.

TEST SETUP

DATA INPUT

We opted to use the syslog data, as Splunk has preconfigured it as one of the source types, which means Splunk knows the fields in the events and can format the data appropriately during indexing. You can use a Splunk App like EventGen or a random syslog generator to generate the syslog file. We used the SplunkIT utility to generate the 1 TB syslog data which was then split into 8 files and placed into the forwarder.

FORWARDERS

We loaded 8 forwarders (**splunk-fwdr01** to **splunk-fwdr08**) each with 128GB of log data that will be forwarded to the Indexers to ingest a total of 1TB of data.

To speed up the data through the Forwarders, we updated the throughput entry **maxKBps** in **limits.conf** to 0 (unlimited). Default throughput of the universal forwarders is 256KBps. For testing purposes, larger throughputs (like 10240 or 0) might be acceptable, but for standard operating procedures, you might want to set this to a value that reflects your environment and not overwhelm your network infrastructure.

```
[splunk@splunk-fwdr01 local]$ more limits.conf
[thruput]
maxKBps=0
[splunk@splunk-fwdr01 local]$
```

Figure 12. *limits.conf* on a forwarder

Forwarders were configured to forward the log data through the private network which is also used for replication traffic between indexers. Figure 13 shows the **outputs.conf** file on a Forwarder.

```
[splunk@splunk-fwdr01 local]$ more outputs.conf
[tcput]
defaultGroup=search_peers

[tcput:search_peers]
autoLB=true
forceTimebasedAutoLB=true
server=splunk-indx01-rep:9997,splunk-indx02-rep:9997,splunk-indx03-rep:9997,splunk-
-indx04-rep:9997,splunk-indx05-rep:9997,splunk-indx06-rep:9997,splunk-indx07-rep:9
997,splunk-indx08-rep:9997
[splunk@splunk-fwdr01 local]$
```

Figure 13. outputs.conf on a forwarder

Note: The `/etc/hosts` file across all Splunk nodes includes hostnames and private IP address details for proper name resolution.

INDEXERS

A set of eight (8) Volumes or LUNs were created on the Pure FlashArray for Hot/Warm, and Cold buckets were attached to the eight indexers.

Figure 14 shows the volume details on Pure FlashArray before ingesting the data.

DASHBOARD

STORAGE

PROTECTION

ANALYSIS

SYSTEM

MESSAGES

Search Hosts and Volumes

Hosts

Volumes

SPLUNK-HOT

SPLUNK-HOT-IDX-01

SPLUNK-HOT-IDX-02

SPLUNK-HOT-IDX-03

SPLUNK-HOT-IDX-04

SPLUNK-HOT-IDX-05

SPLUNK-HOT-IDX-06

SPLUNK-HOT-IDX-07

SPLUNK-HOT-IDX-08

Volumes

Provisioned 90.34 TB

Total Reduction 16.7 to 1

Data Reduction 4.0 to 1

25% full

Used 5.49 TB

Total 22.33 TB

Volumes 24

Destroyed Volumes 0

Destroyed Snapshots 0

SPLUNK-HOT

1-8 of 8

NAME	# HOSTS	# HOST GROUPS	PROVISIONED	SOURCE	VOLUMES	SNAPSHOTS	REDUCTION	SERIAL
 SPLUNK-HOT-IDX-01	1	0	3 TB		88.25 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101B
 SPLUNK-HOT-IDX-02	1	0	3 TB		88.25 KB	0 GB	2.1 to 1	766EE97CCC364E5F00011019
 SPLUNK-HOT-IDX-03	1	0	3 TB		87.75 KB	0 GB	2.2 to 1	766EE97CCC364E5F0001101A
 SPLUNK-HOT-IDX-04	1	0	3 TB		88.75 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101B
 SPLUNK-HOT-IDX-05	1	0	3 TB		87.75 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101C
 SPLUNK-HOT-IDX-06	1	0	3 TB		88.25 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101D
 SPLUNK-HOT-IDX-07	1	0	3 TB		88.25 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101E
 SPLUNK-HOT-IDX-08	1	0	3 TB		97.25 KB	0 GB	2.1 to 1	766EE97CCC364E5F0001101F

Figure 14. Volume details on FlashArray before data ingestion

For ease of management and to enable an easier backup mechanism for buckets, we created separate volumes for Hot/Warm and Cold, even though the performance characteristics will be the same between these volumes.

The volumes were attached to the indexers and discovered. As the **root** user on the Indexer, primary partitions were created on the volumes using **parted** and an XFS filesystem was created on top of the partition and mounted as **/h01** for Hot/Warm, **/c01** for Cold. See Appendix D for details on OS-level operations to accomplish this activity.

We updated the **indexes.conf** file on the Master Cluster node (**splunk-admin1**) with the following entries: The **maxDataSize** was configured with **auto_high_volume** that defaults

to 10GB buckets. Hot and Cold volumes were configured to point to **/h01** and **/c01** on the indexer nodes. New index **idxsyslog** will be created, which will have the **homePath** pointing to the Hot volume and **coldPath** pointing to the Cold volume. The internal indexes were left in the default location (**/opt/splunk/var/lib**) and only the new index **idxsyslog** was placed on the Pure volumes.

```
[splunk@splunk-admin1 local]$ pwd
/opt/splunk/etc/master-apps/_cluster/local
[splunk@splunk-admin1 local]$ more indexes.conf
[default]
homePath.maxDataSizeMB=34406400
maxDataSize=auto_high_volume

[volume:hot]
path=/h01
maxVolumeDataSizeMB=2150400

[volume:cold]
path=/c01
maxVolumeDataSizeMB=2150400

[idxsyslog]
repFactor          = auto
homePath           = volume:hot/idxsyslog/db
coldPath           = volume:cold/idxsyslog/colddb
thawedPath         = $SPLUNK_DB/idxsyslog/thaweddb
maxHotBuckets      = 5
[splunk@splunk-admin1 local]$
```

Figure 15. *indexes.conf* on Master Cluster node

To speed up the indexing process, we opted to enable **Index Parallelization** by setting up two pipelines. By default, the indexer runs on a single pipeline set. Setting it to two requires additional compute and IO resources.

```
[splunk@splunk-admin1 local]$ pwd
/opt/splunk/etc/master-apps/_cluster/local
[splunk@splunk-admin1 local]$ more server.conf
[general]
parallelIngestionPipelines=2
[splunk@splunk-admin1 local]$
```

Figure 16. *server.conf* on Master Cluster node

As the Cisco blades we used for Indexers have enough compute power, and the FlashArray's performance was barely taxed, we enabled index parallelization by adding an entry in the **server.conf** file. See Splunk documentation for more details on Index Parallelization and its effects.

We deployed the index configuration to the peer nodes using the splunk command with apply cluster-bundle option. Alternatively, the index configuration can also be deployed via the GUI (Settings → Distributed Environment → Indexer Clustering → Edit → Distribute Configuration Bundle).

```
[splunk@splunk-admin1 local]$ /opt/splunk/bin/splunk apply cluster-bundle -auth
admin:splunk
Warning: Under some circumstances, this command will initiate a rolling restart
of all peers. This depends on the contents of the configuration bundle. For
details, refer to the documentation. Do you wish to continue? [y/n]: y
Created new bundle with checksum=DD89DB195375135F7DD456F88B423B79
Applying new bundle. The peers may restart depending on the configurations
in applied bundle.
Please run 'splunk show cluster-bundle-status' for checking the status of the
applied bundle.
```

DATA INGESTION

We used the oneshot feature within Splunk to load the syslog data from each forwarder to start the data ingestion process.

```
[splunk@splunk-fwdr01 data01]$ ls -l splunk_data01
-rw-r--r-- 1 splunk splunk 137438953770 Feb  4 11:02 splunk_data01
[splunk@splunk-fwdr01 data01]$ /opt/splunkforwarder/bin/splunk add oneshot -source
./splunk_data01 -sourcetype syslog -index idxsyslog -auth admin:splunk
Oneshot '/x01/data01/splunk_data01' added
[splunk@splunk-fwdr01 data01]$
```

Figure 17. oneshot command to load data

TEST RESULTS

The data ingestion for 1 TB of syslog data from 8 forwarders onto 8 indexers with replication factor (RF) of 3 and search factor (SF) of 2 took **2 hours 40 minutes or 160 minutes**.

Volume Detail: Deployment

Group: All Indexers Volume: **hot** Hide Filters

Select views: All Snapshot Historical

Snapshots

Volume Structure Overview

8 Indexers	1075.04 GB Total Volume Size	134.38 GB Average Volume Size Per Indexer
----------------------	--	---

Instances (8)

Instance	Volume Usage (GB)	Volume Capacity (GB)	Volume Path
splunk-idx02	145.94 / 2100.00	2100.00	/h01
splunk-idx01	101.95 / 2100.00	2100.00	/h01
splunk-idx04	156.87 / 2100.00	2100.00	/h01
splunk-idx05	129.24 / 2100.00	2100.00	/h01
splunk-idx06	117.67 / 2100.00	2100.00	/h01
splunk-idx07	111.05 / 2100.00	2100.00	/h01
splunk-idx08	185.78 / 2100.00	2100.00	/h01
splunk-idx03	126.54 / 2100.00	2100.00	/h01

Figure 18. Hot Volume details as reported by Splunk GUI

PERFORMANCE TESTS

TEST RESULTS

DATA INGESTION

As per Splunk's guidelines⁴, 8 indexers are capable of ingesting 300GB/day/indexer, which equals 2.4TB/day. Based on our configuration of FlashStack for Splunk using a synthetic benchmark on core Splunk Enterprise, we were able to ingest 1TB of data into 8 indexers in 160 minutes, which is equivalent to 9TB/day of data ingestion.

Note: Your mileage may vary based on your specific Splunk workload and use cases. For example, environments and applications that make heavy use of Data Model Acceleration and/or have a high number of scheduled searches like Enterprise Security have lower daily ingest rates (60-100GB/day/indexer).

As always, when in doubt, refer to Splunk's documented best practices for sizing and scaling your environment.

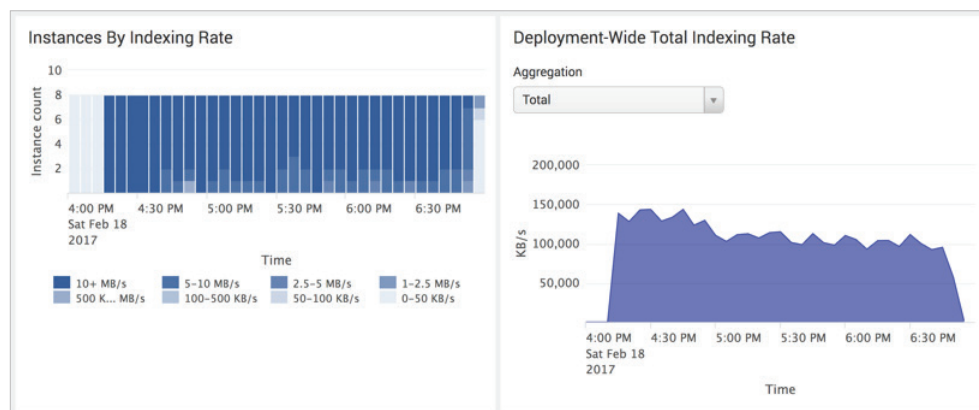


Figure 19. Indexing performance

The indexers were sized not only to support the load during data ingestion, concurrent searches, and reporting overhead that Splunk Enterprise would generate, but also the growth customers would likely experience. During data ingestion, system utilization across 8 indexers was less than 10%, which leaves a lot of computing room for all the other tasks the Indexers would incur.

The current ingestion rate was achieved with 8 forwarders and parallel pipelines at the indexers. Increasing the forwarders can also improve the data ingestion rate as FlashArray has a lot more performance bandwidth available.

⁴ <http://docs.splunk.com/Documentation/Splunk/6.5.3/Capacity/Summaryofperformancerecommendations>

SEARCH PERFORMANCE

We performed a “needle in a haystack” type search with the keyword **needle** against 7 billion events with only one occurrence. Figure 20 shows the total event count for the index **idxsyslog** and Figure 21 shows the search result.

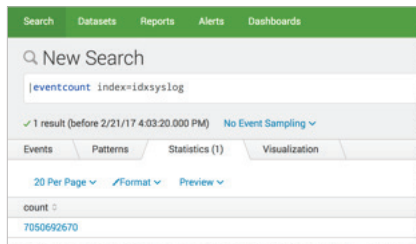


Figure 20. Index Eventcount

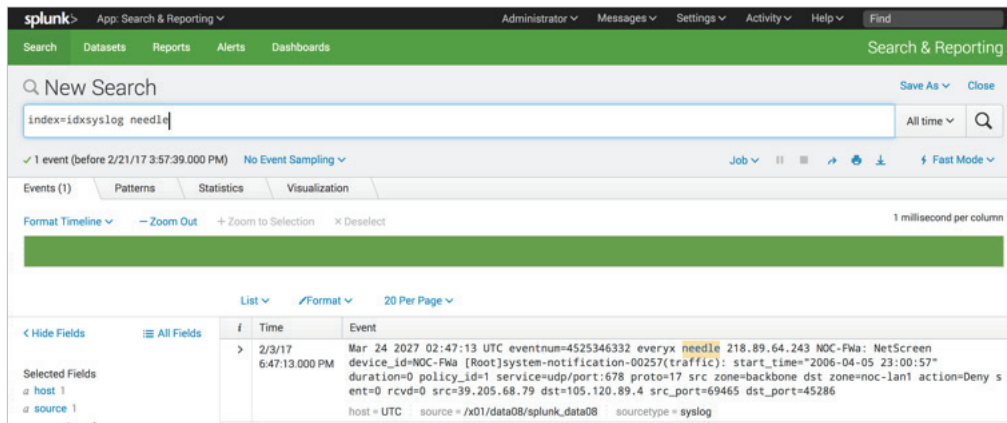


Figure 21. Search results

As you can see from Figure 22, which shows the search result metrics, the overall runtime was 2.1 sec for a single event to be searched across 7 billion events.

Frequently Run Searches					
Report Name/Search String	Count	Median Runtime	Max Runtime	Users	Hosts
search2	2	-	-	admin	splunk-admin1
bundle_rep_log_base	1	-	-	admin	splunk-admin1
search index=idxsyslog needle	1	2.1 sec	2.1 sec	admin	splunk-srch01

Figure 22. Search result metrics

SHARED STORAGE BENEFITS

In a shared storage model, storage resources are treated as a single logical resource that can be used by multiple servers. Servers are allocated storage according to their needs. As needs change, more or less storage can be allocated as necessary. This delivers numerous benefits:

- **Increased Reliability and Higher Availability** – Using shared storage software minimizes downtime, since data is independent from compute resources.
- **Higher Performance Levels** – Shared storage systems are designed to meet even the most rigorous storage demands by offering high I/O rates and low latency.
- **Better scalability** – A good shared storage solution can scale linearly, without downtime or disruption to applications.
- **Central Management and Simplicity** – One pool of storage to manage, which is operationally easier, compared to managing storage server by server.
- **Advanced Data Features** – Features such as thin provisioning, snapshots & clones, deduplication, compression, etc.

The Pure Storage FlashArray delivers all these benefits through software-defined all-flash power and reliability for every need and every budget, from the entry-level FlashArray//M10 to the new FlashArray//X – the first mainstream, 100% NVMe, enterprise-class all-flash array. With our Purity Operating Environment software, every FlashArray model enables organizations to consolidate and accelerate workloads while enjoying simplified operations, proven 99.9999% availability, completely non-disruptive upgrades, and Pure1 support that's above and beyond. Our industry leading data reduction is inline and always-on, which means you will save on storage in addition to power, cooling, and space, and the storage space savings can be kept with data reduction-aware snapshots and replication.

BEST PRACTICES FOR SPLUNK ON PURE FLASHARRAY

STORAGE

Configuring volumes for Splunk indexers could not be any simpler: due to the unique capabilities of flash and the design of Purity, the factors below are neither relevant nor significant on FlashArray.

Factors	Relevancy	Details
Stripe width and depth	Automatic	Purity Operating environment automatically distributes data across all drives in the array
RAID level	Automatic	Pure FlashArray uses RAID-3D, designed to protect against 3 failure modes specific to flash storage: device failure, bit errors, and performance variability
Intelligent data placement	Insignificant	Purity Operating Environment has been designed from the ground up to take advantage of flash's unique capabilities as they are not constrained by the disk paradigm anymore, and, as such, "hot" and "cold" disk platter placements are not relevant

PURE VOLUMES

For ease of bucket management, and to enable backups of Warm or Cold buckets, we recommend using separate Pure volumes for Hot/Warm, Cold, and Frozen buckets (if you decide to use Frozen on FlashArray) per indexer.

Bucket Type	Volume count	Location in indexes.conf
Hot/Warm	1 volume per indexer	[volume:hot] stanza
Cold	1 volume per indexer	[volume:cold] stanza
Frozen ⁵	1 volume per indexer	coldToFrozenDir or coldToFrozenScript under each <index> stanza

As FlashArray volumes are always thin-provisioned, Splunk administrators can provision larger-sized volumes/luns to avoid resizing them.

⁵ <http://docs.splunk.com/Documentation/Splunk/6.5.3/Indexer/Automatearchiving>

LINUX MOUNT OPTIONS

We validated the use of both EXT4 and XFS filesystems for Splunk indexers. As buckets age and when directories are removed, use the **DISCARD** mount option to issue the **TRIM** command to FlashArray to release the space occupied by those directories. Following are the recommended mount options:

discard,noatime

If the discard option is not a preferred option based on your standard operating procedure, make sure to issue the **fstrim** command periodically, once a day or once a week, to release the space at the FlashArray level.

LINUX BEST PRACTICES

The Linux recommended settings for FlashArray, including multipathing queue settings, are documented under the Solutions page at the Pure Storage support site.

https://support.purestorage.com/Solutions/Operating_Systems/Linux/Reference/Linux_Recommended_Settings

CONCLUSION

Analytical applications like Splunk have become mission-critical to organizations for log analytics/security, Operational Intelligence, and decision support. The importance of these use cases will only increase and become more strategic. This reference architecture is designed to provide a validated path to enable enterprises to achieve these goals.

The testing results clearly demonstrate the benefit of deploying Splunk on FlashStack converged infrastructure – a reduction in storage costs, improved availability, simplified management, and a reduction in overall datacenter operational costs. The FlashStack converged infrastructure solution is more than capable of supporting the most demanding of Splunk environments, allowing customers to gain maximum insight from their data, whether it be for log analytics, machine learning, or Internet-of-Things applications. This allows customers to focus on delivering insights and value for their business, without having to worry about the underlying infrastructure.

To learn more about how customers are leveraging the Pure FlashStack solution to accelerate their Splunk deployments go to www.purestorage.com.

REFERENCES

The following documents and links were referred to in preparing this document.

1. Splunk Enterprise Documentation
<https://docs.splunk.com/Documentation/Splunk>
2. Pure Storage Community pages
<https://support.purestorage.com>
3. Cisco UCS Manager GUI Configuration Guide
http://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/sw/gui/config/guide/2-2/b_UCSM_GUI_Configuration_Guide_2_2.pdf
4. RedHat Customer Portal
<https://access.redhat.com/>
5. Cluster Shell documentation
<http://clustershell.readthedocs.io/en/latest/>
6. Linux Host Cloning using Pure Snapshot
<http://community.purestorage.com/t5/Pure-Customer-Knowledge-Base/Cloning-SAN-Boot-volumes-on-Pure-amp-UCS/ta-p/6719>

APPENDIX A:

SPLUNK COMPONENTS

An **Index** is a collection of databases which are subdirectories. Splunk manages all its index data in the flat file format and doesn't use any sophisticated database management systems.

An **Indexer Cluster** is a group of indexers configured to replicate each other's data, so that the system keeps multiple copies of all data to prevent data loss while enabling data availability for searching in case of an indexer node failure. As Indexer cluster features automatic failover, in case of an indexer failure Splunk automatically fails over that indexer to the next indexer which enables the incoming data to be indexed and the indexed data continues to be available for searching.

Search Head is a Splunk Enterprise instance that handles incoming search requests. In a distributed search environment, the search head sends requests to a group of indexers, aka "search peers", which perform the actual searches on their indexes and send the results back. The search head merges the results and presents them to the user. Dedicated search heads don't have any indexes dedicated to perform search management functions like consolidate and display.

A **Search Head Cluster** is a group of interchangeable and highly available search heads, aka **Cluster Members**, that share configurations, job scheduling, and search artifacts. By increasing concurrent user capacity and by eliminating single points of failure, search head clusters enable highly available and scalable search services.

Forwarder is a small-footprint version of a Splunk instance that forwards data to remote indexers for data processing and storage.

Cluster Master or **Master Node** is another Splunk Enterprise instance that regulates the functioning of an indexer cluster.

Deployer is a Splunk Enterprise instance that distributes apps and other configurations to the search head cluster members. One requirement for deployer is not to run on the same instance as a cluster member.

Monitoring Console is the Splunk Enterprise monitoring tool that allows you to view detailed performance information about your Splunk Enterprise deployment through pre-defined dashboards. Some of the available dashboards are indexing performance, index and volume usage, license usage, search performance, search head and indexer clustering, etc.

License Master controls one or more license slaves and is generally used when you have more than one indexer and want to manage their access to purchased license capacity from a central location.

Deployment Server is a Splunk Enterprise instance that acts as a centralized configuration manager. It is the tool for distributing configurations, apps, and content updates to groups of Splunk Enterprise instances. This is generally used to update forwarders, non-clustered indexers, and search heads. This is not a required component to manage forwarders and other instances. Based on your preference and/or standard operating procedures, tools like Chef, Puppet, Salt, etc., can be used.

APPENDIX B: RAPID DEPLOYMENT OF SPLUNK HOSTS

Cisco UCS service profiles represent all the attributes of a logical server that disassociates from the physical hardware and limits the constraints around server provisioning, but they don't address the rapid deployment of the underlying operating system. UCS service profiles complemented by Pure's FlashRecover snapshot functionality helps address this problem. A key operational value proposition of FlashStack for Splunk is that it helps with rapid deployment of Splunk hosts.

This requires the boot LUN for the systems to be configured with a SAN boot on FlashArray, which is the preferred approach in a FlashStack environment. The idea is to install an operating system on a server that is SAN-booted from Pure, to be treated as the gold image, and then snapshot that image to create clones of the host to deploy on all the remaining servers.

The high-level process for rapid deployment of Splunk Hosts is:

1. Create service profiles from the Service Profile Template for all Splunk servers to instantiate the vHBAs and vNICs.
2. Create zoning information on MDS-9148S to allow FlashArray to see the server ports.
3. Create all the Splunk hosts on FlashArray with their relevant WWNs.
4. Create a volume/LUN on FlashArray and attach it to the source server where the operating system will be installed.
5. Boot the source server with the ISO image of the OS attached.
6. Install the operating system on top of the LUN with all relevant packages and configurations, to be treated as the gold image.
7. Run a clone script on the source server to issue a Pure FlashRecover snapshot after mimicking the identity of the target server.
8. Create clones of the source LUN using "Copy feature" from FlashArray to create the boot luns for the remaining Splunk servers.
9. Attach the cloned boot LUNs to the hosts on FlashArray.
10. Boot the remaining servers to start them with a new identity.

The following sections overview the configuration of the source server (**splunk-admin1**) after the installation of Red Hat Linux 7.2, with the details of relevant packages and software to get to the gold image state, followed by the cloning process to deploy 12 servers (8 indexers, 3 search heads, 1 additional admin).

Servers / Policies / root / Boot Policies / Boot Policy Boot-Splunk

General Events

Show Policy Usage Use Global

Description : Boot Policy for Splunk

Owner : Local

Reboot on Boot Order Change : ☐

Enforce vNIC/vHBA/iSCSI Name : ☐

Boot Mode : ☒ Legacy ☐ Uefi

Warning

The type (primary/secondary) does not indicate a boot order presence.
The effective order of boot devices within the same device class (LAN/Storage/iSCSI) is determined by PCIe bus scan order.
If **Enforce vNIC/vHBA/iSCSI Name** is selected and the vNIC/vHBA/iSCSI does not exist, a config error will be reported.
If it is not selected, the vNICs/vHBAs are selected if they exist, otherwise the vNIC/vHBA with the lowest PCIe bus scan order is used.

+ Local Devices

+ CIMC Mounted vMedia

+ vNICs

+ vHBAs

+ iSCSI vNICs

Boot Order

+ - Advanced Filter Export Print

Name	vNIC/vHBA/iSCSI...	Type	WWN	LUN Na...
Local CD/DVD				
▼ San				
▼ SAN Primary				
	fc0	Primary		
	SAN Target Primary	Primary	52:4A:93:77:33:06:3B:00	1
	SAN Target Secondary	Secondary	52:4A:93:77:33:06:3B:11	1

Figure 23. Boot policy

A Service Profile Template with a boot from SAN policy was created within Cisco UCS Manager to deploy the 13 servers quickly with a standard configuration. The boot from SAN policy is configured to boot from Pure Storage boot LUNs. The service profile template for Splunk included 4 vHBAs and 2 vNICs (one for public and one for private traffic to handle replication and search traffic). The SAN target primary and secondary were populated with the Pure FlashArray/M's WWN and the LUN number set to 1.

A single LUN (**SPLUNK-ADMIN-NODE-01-BootVol**) was provisioned for the very first node (**SPLUNK-ADMIN-NODE-01**) and the first stateless server was booted off SAN. We installed RHEL 7.2 on this LUN with a software selection of **Infrastructure Server**, including the following add-ons.

Performance Tools

Development Tools

Security Tools

Network File System Client (if you want to mount any network shares, which might be relevant if you decide to use NFS shares for Frozen buckets within Splunk)

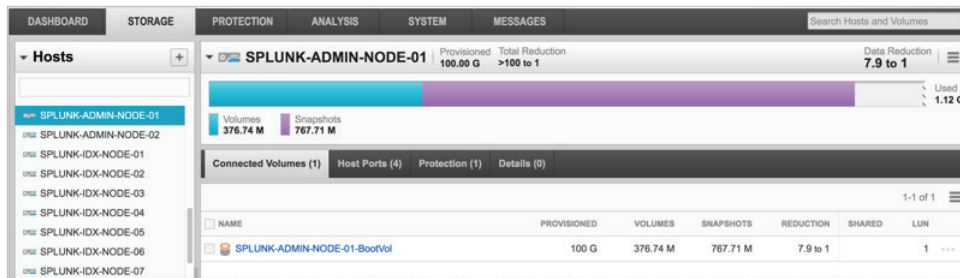


Figure 24. Boot Lun

POST OS INSTALL CONFIGURATION

Once RHEL 7.2 is installed and rebooted, update the system with the following configuration.

1. Enable NTP and restart of NTP daemon across reboots

Install the ntp package if it is not available. Once the ntp package is installed update the `/etc/ntp.conf` with the ntp server information.

```
[root@splunk-admin1 ~]# yum -y install ntp
[root@splunk-admin1 ~]# more /etc/ntp.conf
server time1.purestorage.com
server time2.purestorage.com
```

Start the ntp daemon using service or the `systemctl` command.

```
[root@splunk-admin1 ~]# service ntpd start
Redirecting to /bin/systemctl start ntpd.service
```

Enable the start of NTP daemons across reboots.

```
[root@splunk-admin1 ~]# chkconfig ntpd on
```

Note: Forwarding request to 'systemctl enable ntpd.service'.

Created symlink from `/etc/systemd/system/multi-user.target.wants/ntpd.service` to `/usr/lib/systemd/system/ntpd.service`.

2. Enable syslog

```
[root@splunk-admin1 ~]# rsyslogd -v
rsyslogd 7.4.7, compiled with:
```

FEATURE_REGEX:	Yes
FEATURE_LARGEFILE:	No
GSSAPI Kerberos 5 support:	Yes
FEATURE_DEBUG (debug build, slow code):	No
32bit Atomic operations supported:	Yes
64bit Atomic operations supported:	Yes
Runtime Instrumentation (slow code):	No
uuid support:	Yes

See <http://www.rsyslog.com> for more information.

```
[root@splunk-admin1 ~]# service rsyslog status
Redirecting to /bin/systemctl status rsyslog.service
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Thu 2016-12-08 23:17:42 PST; 16h ago
 Main PID: 1461 (rsyslogd)
    CGroup: /system.slice/rsyslog.service
            - 1461 /usr/sbin/rsyslogd -n

Dec 08 23:17:42 splunk-admin1.puretec.purestorage.com systemd[1]: Starting System
Logging Service...
Dec 08 23:17:42 splunk-admin1.puretec.purestorage.com systemd[1]: Started System
Logging Service.
```

3. Set resource limits on OS as required by Splunk

Update the `nofile` property in `/etc/security/limits.conf` file to enable the inodes that can be opened simultaneously. Default is 1024.

```
root    soft    nofile   64000
root    hard    nofile   64000
splunk  soft    nofile   64000
splunk  hard    nofile   64000
```

```
[root@splunk-admin1 ~]# su -
Last login: Fri Dec  9 13:58:28 PST 2016 from 192.168.3.4 on pts/1
[root@splunk-admin1 ~]# ulimit -n
64000
```

Set TCP retries to 5 in `/etc/sysctl.conf` file.

```
[root@splunk-admin1 ~]# vi /etc/sysctl.conf
[root@splunk-admin1 ~]# sysctl -p
net.ipv4.tcp_retries2 = 5
```

Unless IPV6 is your standard network setup, it can be disabled.

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

4. Disable Transparent Huge Pages

Append `/etc/rc.local` with the following two commands

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

The current value of transparent hugepage can be verified as followed and can be changed.

```
[root@splunk-admin1 ~]# cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
[root@splunk-admin1 ~]# cat /sys/kernel/mm/transparent_hugepage/defrag
[always] madvise never
[root@splunk-admin1 ~]# echo never > /sys/kernel/mm/transparent_hugepage/enabled
[root@splunk-admin1 ~]# echo never > /sys/kernel/mm/transparent_hugepage/defrag
[root@splunk-admin1 ~]# cat /sys/kernel/mm/transparent_hugepage/defrag
always madvise [never]
[root@splunk-admin1 ~]# cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
```

5. Update /etc/hosts with all Splunk hosts details

Update **/etc/hosts** with all the Splunk hosts along with their IP details. Also include the replication network details.

```
10.21.122.151 splunk-admin1 splunk-admin1.puretec.purestorage.com
10.21.122.152 splunk-admin2 splunk-admin2.puretec.purestorage.com
10.21.122.153 splunk-srch01 splunk-srch01.puretec.purestorage.com
10.21.122.154 splunk-srch02 splunk-srch02.puretec.purestorage.com
10.21.122.155 splunk-srch03 splunk-srch03.puretec.purestorage.com

10.21.122.156 splunk-indx01 splunk-indx01.puretec.purestorage.com
10.21.122.157 splunk-indx02 splunk-indx02.puretec.purestorage.com
10.21.122.158 splunk-indx03 splunk-indx03.puretec.purestorage.com
10.21.122.159 splunk-indx04 splunk-indx04.puretec.purestorage.com
10.21.122.160 splunk-indx05 splunk-indx05.puretec.purestorage.com
10.21.122.161 splunk-indx06 splunk-indx06.puretec.purestorage.com
10.21.122.162 splunk-indx07 splunk-indx07.puretec.purestorage.com
10.21.122.163 splunk-indx08 splunk-indx08.puretec.purestorage.com

192.168.1.156 splunk-indx01-rep
192.168.1.157 splunk-indx02-rep
192.168.1.158 splunk-indx03-rep
192.168.1.159 splunk-indx04-rep
192.168.1.160 splunk-indx05-rep
192.168.1.161 splunk-indx06-rep
192.168.1.162 splunk-indx07-rep
192.168.1.163 splunk-indx08-rep
```

6. Apply Linux best practices for Pure Storage

The Linux best practices for Pure Storage recommends updating the HBA timeout and Queue settings. The Linux best practices is available at https://support.purestorage.com/Solutions/Operating_Systems/Linux/Reference/Linux_Recommended_Settings


```
cat /etc/udev/rules.d/99-pure-storage.rules

# Recommended settings for Pure Storage FlashArray.

# Use noop scheduler for high-performance solid-state storage
ACTION=="add|change", KERNEL=="sd*[!0-9]", SUBSYSTEM=="block", ENV{ID_
VENDOR}=="PURE", ATTR{queue/scheduler}="noop"

# Reduce CPU overhead due to entropy collection
ACTION=="add|change", KERNEL=="sd*[!0-9]", SUBSYSTEM=="block", ENV{ID_
VENDOR}=="PURE", ATTR{queue/add_random}="0"

# Spread CPU load by redirecting completions to originating CPU
ACTION=="add|change", KERNEL=="sd*[!0-9]", SUBSYSTEM=="block", ENV{ID_
VENDOR}=="PURE", ATTR{queue/rq_affinity}="2"

# Set the HBA timeout to 60 seconds
ACTION=="add", SUBSYSTEMS=="scsi", ATTRS{model}=="FlashArray      ", RUN+="/bin/sh
-c 'echo 60 > /sys/$DEVPATH/device/timeout'"
```

7. Update multipath configuration as per Pure Storage's recommendation

The best practice for managing multipathed devices is to use the alias instead of `user_friendly_names`. This gives control in managing the device name that persists across reboots.

One exception in using the alias is on the boot lun if that system will be snapshotted to clone further systems. If alias is used for the boot lun on a system and then cloned to create a second system, it involves additional post cloning steps to identify the new WWID, update the **multipath.conf** file, restart multipath services, and probably reboot the system. Instead, not including an alias for the boot lun enables cloning of the system seamlessly – and provisioning time of new system is improved significantly.

Update the **/etc/multipath.conf** as recommended in the Linux Best practices article.

```
defaults {
    find_multipaths yes
    polling_interval      10
}

devices {
    device {
        vendor            "PURE"
        path_selector      "queue-length 0"
        path_grouping_policy multibus
        path_checker        tur
        fast_io_fail_tmo    10
        dev_loss_tmo        60
        no_path_retry       0
    }
}
```

8. Update lvm.conf file to avoid boot LVM assuming single path

LVM is used for the root volume in Linux. When the system is started, LVM scans all available block devices for LVM signatures to check for physical volumes. In a

multipathed environment, LVM will find signatures on the device file representing the singular multipathed device and the device files representing each of the paths to the volume in question (all /dev/sdb, /dev/sdc, /dev/sdd, et cetera...). In this case, all the device files represent the same volume and will have the same PV signature. Hence LVM may choose to use any one of them to access the volume. This will work fine but in a failure event, as LVM is using a single path device, if that path is not available it will render the system unusable, even though other paths might have been available through multipathing. To avoid this, we ought to inform LVM to disregard the non-multipathed device files using the filter clause. As we are using alias for the multipath, it is preferable to identify the devices with the /dev/mapper prefix.

Update the following three entries on **/etc/lvm/lvm.conf** with the configuration provided.

```
scan = [ "/dev/mapper" ]
preferred_names = [ "^/dev/mapper", "^/dev/mpath/", "^/dev/mapper/mpath", "^/dev/[hs]d"
]
filter = [ "r|/dev/sd.*|", "a|/dev/mapper/.*/|", "a|.*/|" ]
```

10. Update initramfs after all changes are completed

Update the **initramfs** after all changes are performed to persist the change across reboots using the following command. This is critical before cloning the source system. If needed, take a backup of the current image before creating the new image.

```
[root@splunk-admin1 ~]# cp -p initramfs-$(uname -r).img initramfs-$(uname -r).img.bak
[root@splunk-admin1 ~]# dracut -f -v
```

11. Install Splunk software and change the password of the splunk user.

The core Splunk Enterprise software is available as a single software package and designed in such a way that it can be configured to function with a specific role. This means the installation of Splunk across all nodes is the same, with no specific parameters to configure the function of the node at the time of installation.

Download Splunk Enterprise 6.5.1 rpm to the admin server.

Install the rpm (which automatically creates the “splunk” user and “splunk” group).

```
[root@splunk-admin1 sw]# rpm -Uvh splunk-6.5.1-f74036626f0c-linux-2.6-x86_64.rpm
warning: splunk-6.5.1-f74036626f0c-linux-2.6-x86_64.rpm: Header V4 DSA/SHA1
Signature, key ID 653fb112: NOKEY
Preparing...                               ##### [100%]
Updating / installing...
 1:splunk-6.5.1-f74036626f0c   ##### [100%]
complete

[root@splunk-admin1 sw]# grep -i splunk /etc/passwd
splunk:x:1000:1000:Splunk Server:/opt/splunk:/bin/bash

[root@splunk-admin1 sw]# passwd splunk
Changing password for user splunk.
```

```
New password: *****
Retype new password: *****
passwd: all authentication tokens updated successfully.
```

12. Run the cloning script to provision the remaining Splunk servers.

Once all the above configurations are complete, run the following **clonesys.sh** script to take a snapshot of the source server with pre-configured network interfaces and a hostname that reflects the clone server. The script reads host entry records from the **hosts.txt** file under the same directory. Enter details like hostname, public IP address, private IP address, hostname as it appears in FlashArray, public MAC address, and private MAC address for all the hosts that are to be cloned from the source system.

```
#
# Usage: clonesys.sh
# Script reads one record at a time from hosts.txt file which has the following
format:
# <hostname> <Public IP> <Private IP> <Hostname as in Pure FA> <Public MAC addr>
<Private MAC addr>
#
# The script updates the network config & hostname on the source server with the
details from
# hosts.txt for every server and takes Pure FlashRecover snapshot, instantiates
the snapshot
# to a volume and attaches the volume to the target server.
#
# Sample hosts.txt
#splunk-srch01 10.21.122.153 192.168.1.153 SPLUNK-SRCH-NODE-01 00:25:B5:84:48:0D
00:25:B5:92:93:5D
#splunk-srch02 10.21.122.154 192.168.1.154 SPLUNK-SRCH-NODE-02 00:25:B5:84:48:1D
00:25:B5:92:93:1D
#splunk-srch03 10.21.122.155 192.168.1.155 SPLUNK-SRCH-NODE-03 00:25:B5:84:48:4C
00:25:B5:92:92:BD
#splunk-indx01 10.21.122.156 192.168.1.156 SPLUNK-INDX-NODE-01 00:25:B5:84:48:2D
00:25:B5:92:92:2D
#splunk-indx02 10.21.122.157 192.168.1.157 SPLUNK-INDX-NODE-02 00:25:B5:84:48:2E
00:25:B5:92:92:3E
#splunk-indx03 10.21.122.158 192.168.1.158 SPLUNK-INDX-NODE-03 00:25:B5:84:48:3E
00:25:B5:92:93:6E
#splunk-indx04 10.21.122.159 192.168.1.159 SPLUNK-INDX-NODE-04 00:25:B5:84:48:0E
00:25:B5:92:93:2E
#
# The functions pubip_config, privip_config and snap_copy should be updated to meet
# your environment requirements
#
function pubip_config {
```

```

cat << EOF > /etc/sysconfig/network-scripts/ifcfg-enp6s0
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
NAME=enp6s0      # Modify to meet your environment requirement
DEVICE=enp6s0   # Modify to meet your environment requirement
ONBOOT=yes
PEERDNS=no
IPADDR=$1
HWADDR=$2
PREFIX=24        # Modify to meet your environment requirement
GATEWAY=10.21.122.1 # Modify to meet your environment requirement
EOF
}

function prvip_config {
    cat << EOF2 > /etc/sysconfig/network-scripts/ifcfg-enp7s0
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=no
NAME=enp7s0      # Modify to meet your environment requirement
DEVICE=enp7s0   # Modify to meet your environment requirement
ONBOOT=yes
IPADDR=$1
HWADDR=$2
PREFIX=24        # Modify to meet your environment requirement
EOF2
}

function upd_host {
    hostname $1
    echo $1 > /etc/hostname
}

function snap_copy {
    #
    # Modify purehostname with your Pure FlashArray connection info (IP or hostname)
    # Modify the volume name SOURCE-BOOTVOL with your source boot volume from which
    clones to be made
    #
    ssh -n pureuser@purehostname purevol snap --suffix rhel72-gold-$1 SOURCE-BOOTVOL
    ssh -n pureuser@purehostname purevol copy --overwrite SOURCE-BOOTVOL.rhel72-
gold-$1 $1-BootVol

```

```

echo "Attaching volume to host $2"

ssh -n pureuser@purehostname purehost connect --lun 1 --vol $1-BootVol $2
}

# Preserve the hostname and network interface config of the source system
ohost=$(hostname)
cp /etc/sysconfig/network-scripts/ifcfg-enp7s0 /etc/sysconfig/network-scripts/ifcfg-enp7s0.orig
cp /etc/sysconfig/network-scripts/ifcfg-enp6s0 /etc/sysconfig/network-scripts/ifcfg-enp6s0.orig

while read host pubip prvip purehname pubmac prvmac
do
    echo "Working on $host"
    prvip_config $prvip $prvmac
    pubip_config $pubip $pubmac
    upd_host $host
    sync
    sleep 2
    sync
    snap_copy $host $purehname
    sleep 2
done < hosts.txt

# Revert back the hostname and network interface config back to the source system
upd_host $ohost
mv /etc/sysconfig/network-scripts/ifcfg-enp6s0.orig /etc/sysconfig/network-scripts/ifcfg-enp6s0
mv /etc/sysconfig/network-scripts/ifcfg-enp7s0.orig /etc/sysconfig/network-scripts/ifcfg-enp7s0

```

*Invoking Pure CLI through SSH requires the password to be entered every time. This can be avoided by using REST-based APIs where authentication can be done once for the whole session. Alternatively, you can generate a public key from your workstation and save it inside the Pure FA using the **pureadmin setattr --publickey** command.*

In addition to the service profiles, the use of Pure Storage's FlashRecover snapshots with SAN boot policy brings the following benefits

- **Scalability** – Rapid deployment of new servers to the environment
- **Manageability** – Seamless hardware maintenance and upgrades without any restrictions.
- **Flexibility** – Easily repurpose physical servers for different applications and services as needed

- **Availability** – Hardware failures are not impactful and critical: in the rare case of a server failure, it is easier to associate the logical service profile to another healthy physical server to reduce any impact

APPENDIX C: CLUSTER SHELL UTILITY INSTALLATION

The Cluster Shell (**clush**) utility enables system and Splunk administrators to be more productive in managing Linux clusters by executing commands in parallel on a cluster. It can execute commands within shell scripts or interactively. **clush** requires ssh.

1. To use clush, first install the EPEL (Extra Packages for Enterprise Linux) repository on the central server where you would manage other servers, followed by installing cluster shell utility. In our setup, we had setup EPEL on the **splunk-admin1** server, through which we would be managing all other Splunk nodes.

```
[root@splunk-admin1 sw]# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
--2017-01-03 14:02:34-- https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
Resolving dl.fedoraproject.org (dl.fedoraproject.org)... 209.132.181.23, 209.132.181.26, 209.132.181.25, ...Connecting to dl.fedoraproject.org (dl.fedoraproject.org)|209.132.181.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14612 (14K) [application/x-rpm]
Saving to: 'epel-release-latest-7.noarch.rpm'

100%[=====>] 14,612
--.-K/s in 0.02s

2017-01-03 14:02:34 (596 KB/s) - 'epel-release-latest-7.noarch.rpm' saved [14612/14612]

[root@splunk-admin1 sw]# yum install epel-release-latest-7.noarch.rpm
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Examining epel-release-latest-7.noarch.rpm: epel-release-7-8.noarch
Marking epel-release-latest-7.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-8 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version Repository                               Size
=====
```

```

Installing:
  epel-release           noarch  7-8      /epel-release-latest-7.noarch  24 k

Transaction Summary
=====
Install 1 Package

Total size: 24 k
Installed size: 24 k
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : epel-release-7-8.noarch
               1/1
  Verifying  : epel-release-7-8.noarch
               1/1

Installed:
  epel-release.noarch 0:7-8

Complete!

[root@splunk-admin1 sw]# yum -y install clustershell
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package clustershell.noarch 0:1.7.2-1.el7 will be installed
--> Processing Dependency: PyYAML for package: clustershell-1.7.2-1.el7.noarch
--> Running transaction check
---> Package PyYAML.x86_64 0:3.10-11.el7 will be installed
--> Processing Dependency: libyaml-0.so.2()(64bit) for package: PyYAML-3.10-11.el7.x86_64
--> Running transaction check
---> Package libyaml.x86_64 0:0.1.4-11.el7_0 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version      Repository      Size
=====
Installing:
  clustershell     noarch    1.7.2-1.el7  epel             363 k

```

```

Installing for dependencies:
PyYAML          x86_64      3.10-11.el7      rhel-7-server-eus-rpms  153 k
libyaml         x86_64      0.1.4-11.el7_0   rhel-7-server-eus-rpms   55 k

Transaction Summary
=====
Install 1 Package (+2 Dependent packages)
Total download size: 571 k
Installed size: 2.4 M

Downloading packages:
(1/3): libyaml-0.1.4-11.el7_0.x86_64.rpm | 55 kB 00:00:00
(2/3): PyYAML-3.10-11.el7.x86_64.rpm | 153 kB 00:00:00
warning: /var/cache/yum/x86_64/7Server/epel/packages/clustershell-1.7.2-1.el7.noarch.
rpm: Header V3 RSA/SHA256 Signature, key ID 352c64e5: NOKEY
Public key for clustershell-1.7.2-1.el7.noarch.rpm is not installed
(3/3): clustershell-1.7.2-1.el7.noarch.rpm | 363 kB 00:00:00
-----
Total                               1.5 MB/s | 571 kB 00:00:00
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Importing GPG key 0x352C64E5:
  Userid      : "Fedora EPEL (7) <epel@fedoraproject.org>"
  Fingerprint: 91e9 7d7c 4a5e 96f1 7f3e 888f 6a2f aea2 352c 64e5
  Package     : epel-release-7-8.noarch (@/epel-release-latest-7.noarch)
  From        : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libyaml-0.1.4-11.el7_0.x86_64
                1/3
  Installing : PyYAML-3.10-11.el7.x86_64
                2/3
  Installing : clustershell-1.7.2-1.el7.noarch
                3/3
  Verifying  : libyaml-0.1.4-11.el7_0.x86_64
                1/3
  Verifying  : clustershell-1.7.2-1.el7.noarch
                2/3
  Verifying  : PyYAML-3.10-11.el7.x86_64
                3/3
Installed:

```



```
clustershell.noarch 0:1.7.2-1.el7

Dependency Installed:
  PyYAML.x86_64 0:3.10-11.el7               libyaml.x86_64 0:0.1.4-11.el7_0

Complete!
```

2. Setup passwordless SSH access to all nodes that are to be managed by the central server through clush. If you want to manage all nodes as splunk user, perform the following as “splunk” user.

2.1 Configure an SSH publickey for passwordless access to all nodes using the ssh-keygen command.

2.2 Run ssh-copy-id command to copy the id_rsa.pub file across all the nodes.

```
[splunk@splunk-admin1 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/splunk/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/splunk/.ssh/id_rsa.
Your public key has been saved in /home/splunk/.ssh/id_rsa.pub.
The key fingerprint is:
66:5b:97:cc:5b:3e:f2:8a:d4:4a:91:11:88:e6:89:34 splunk@splunk-admin1
The key's randomart image is:
+--[ RSA 2048]-----+
|      .  .          |
|    E o . .         |
|   . = . .          |
|  . o   = .         |
|      S + = .       |
|    o o + +         |
|   . o + o          |
|    o o o .         |
|    o ...           |
+-----+

[splunk@splunk-admin1 ~]$
[splunk@splunk-admin1 ~]$ for host in admin1 admin2 indx01 indx02 indx03 indx04
indx05 indx06 indx07 indx08 srch01 srch02 srch03;
do
    echo -n "$host -> ";
    ssh-copy-id -i ~/.ssh/id_rsa.pub splunk-$host;
done
```

3. Configure the group details for the cluster shell by editing the following file with the node details.

```
vi /etc/clustershell/groups
all: splunk-admin[1-2],splunk-indx[01-08],splunk-srch[01-03]
indexers: splunk-indx[01-08]
searchers: splunk-srch[01-03]
```

4. Verify the cluster shell utility works by running a command. If you want to run the command only on indexers nodes, you can use the “-g indexers” argument, which limits the command to be run on the group specified.

```
[splunk@splunk-admin1 ~]$ clush -a hostname
splunk-admin2: splunk-admin2
splunk-admin1: splunk-admin1
splunk-indx01: splunk-indx01
splunk-indx03: splunk-indx03
splunk-indx02: splunk-indx02
splunk-indx04: splunk-indx04
splunk-indx06: splunk-indx06
splunk-srch02: splunk-srch02
splunk-indx05: splunk-indx05
splunk-indx08: splunk-indx08
splunk-srch03: splunk-srch03
splunk-srch01: splunk-srch01
splunk-indx07: splunk-indx07

[root@splunk-admin1 ~]# clush -g indexers hostname
splunk-indx04: splunk-indx04
splunk-indx01: splunk-indx01
splunk-indx03: splunk-indx03
splunk-indx02: splunk-indx02
splunk-indx05: splunk-indx05
splunk-indx06: splunk-indx06
splunk-indx07: splunk-indx07
splunk-indx08: splunk-indx08
```

For more information on ClusterShell, visit <http://clustershell.readthedocs.io/en/latest/>

APPENDIX D: SPLUNK VOLUMES SETUP

1. Create the Hot and Cold volumes on Pure FlashArray using the Pure GUI.

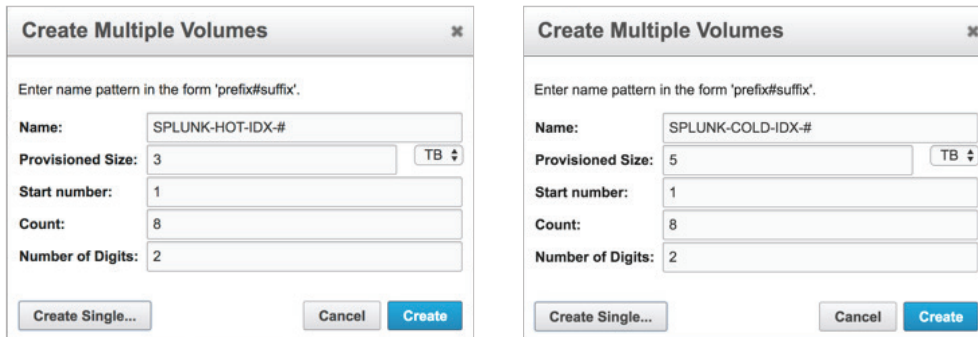


Figure 25. Volume creation

Alternatively, the volumes can be created using the CLI as well. Replace the array hostname according to your environment.

```
for i in {1..8}
do
    ssh pureuser@10.21.122.12 purevol create -size 3T SPLUNK-HOT-IDX-0$i
    ssh pureuser@10.21.122.12 purevol create -size 5T SPLUNK-COLD-IDX-0$i
done
```

2. Attach the 'Hot' and 'Cold' volumes to their corresponding indexer server. This can be easily accomplished through shell scripting with CLI.

```
for i in {1..8}
do
    ssh pureuser@10.21.122.12 purevol connect --host SPLUNK-IDX-NODE-0$i SPLUNK-
HOT-IDX-0$i
    ssh pureuser@10.21.122.12 purevol connect --host SPLUNK-IDX-NODE-0$i SPLUNK-
COLD-IDX-0$i
done
```

3. As our preference is to use aliases for the multipath devices, as root update the `/etc/multipath.conf` with the alias information to reflect the respective volumes in the indexer servers. Excerpt of the `/etc/multipath.conf` file.

```
multipath {
    wwid      3624a9370766ee97ccc364e5f00011018
    alias     splunk_hot_data01
}
```

Note: The WWID can be obtained from the Pure GUI under the volume's Details tab. The first 9 characters will always be 3624a9370, as it corresponds to the designator for the Pure device regardless of the host/array combination.

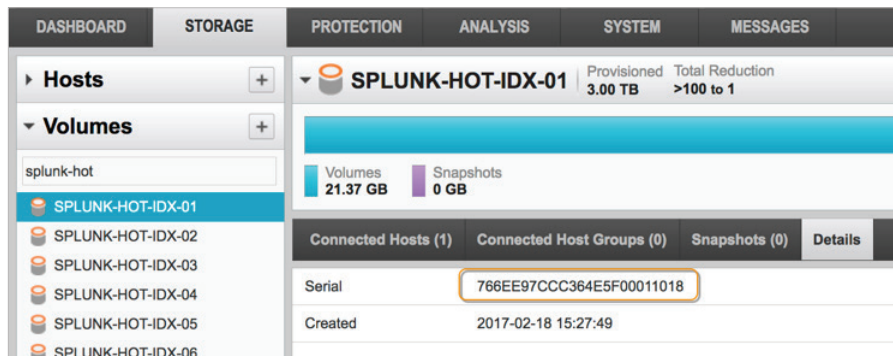


Figure 26. Volume serial number

- Restart the multipathd services to reflect the alias information.

Note: Activities like this where the same command has to be executed across all indexer nodes can be easily accomplished through multi-terminal utilities like csshx for Mac, or MobaXterm for Windows, or Cluster Shell for Linux. The following shows the Cluster Shell usage.

```
[root@splunk-admin1 pure]# clush -g indexers systemctl restart multipathd
```

- Rescan the SCSI devices across all the indexer nodes for the attached Hot and Cold volumes.

```
[root@splunk-admin1 pure]# clush -g indexers rescan-scsi-bus.sh
```

- Create the primary partition on each of the volume (Hot and Cold) on every indexer node. Since the file size is beyond 2TB, use **parted** to create the partition instead of **fdisk**.

```
[root@splunk-idx01 ~]# parted /dev/mapper/splunk_hot_data01
GNU Parted 3.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: /dev/mapper/splunk_hot_data01: unrecognised disk label
Model: Linux device-mapper (multipath) (dm)
Disk /dev/mapper/splunk_idx_data01: 3299GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
(parted) mklabel gpt
(parted) mkpart primary 0GB 3299GB
(parted) print
```

```
Model: Linux device-mapper (multipath) (dm)
```

```
Disk /dev/mapper/splunk_hot_data01: 3299GB
```

```
Sector size (logical/physical): 512B/512B
```

```
Partition Table: gpt
```

```
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
1	4194kB	3299GB	3299GB		primary	

```
(parted) quit
```

```
Information: You may need to update /etc/fstab.
```

```
[root@splunk-indx01 ~]# ls -ltr /dev/mapper/splunk*
```

```
lrwxrwxrwx 1 root root 7 Jan 17 18:58 /dev/mapper/splunk_hot_data01p1 -> ../dm-7
```

```
lrwxrwxrwx 1 root root 7 Jan 17 18:59 /dev/mapper/splunk_hot_data01 -> ../dm-6
```

7. Make the filesystem on the primary partition type XFS for both Hot and Cold volumes.

```
[root@splunk-indx01 ~]# mkfs -t xfs /dev/mapper/splunk_hot_data01p1
```

```
[root@splunk-indx01 ~]# mkfs -t xfs /dev/mapper/splunk_cold_data01p1
```

8. Update **/etc/fstab** with the filesystem details and mount options.

```
/dev/mapper/splunk_hot_data01p1 /h01 xfs noatime,discard,nobarrier 1 2
```

```
/dev/mapper/splunk_cold_data01p1 /c01 xfs noatime,discard,nobarrier 1 2
```

9. Create the same directory structure across all indexer nodes to mount the hot and cold volumes.

```
[root@splunk-admin1 ~]# clush -g indexers mkdir -p /h01
```

```
[root@splunk-admin1 ~]# clush -g indexers mkdir -p /c01
```

10. Mount the filesystems to their respective directories.

```
[root@splunk-admin1 ~]# clush -g indexers mount /h01
```

```
[root@splunk-admin1 ~]# clush -g indexers mount /c01
```

11. Change the ownership of these directories to allow Splunk user to read and write. If Splunk doesn't have the correct permissions, the **indexes.conf** cannot be deployed successfully.

```
[root@splunk-admin1 ~]# clush -g indexers chown splunk:splunk /h01
```

```
[root@splunk-admin1 ~]# clush -g indexers chown splunk:splunk /c01
```

12. As splunk user, Update **indexes.conf** in the Master node to reflect the Hot and Cold volumes.

```
[splunk@splunk-admin1 local]$ more /opt/splunk/etc/master-apps/_cluster/local/indexes.conf
```

```
[default]
```

```
homePath.maxDataSizeMB=34406400
```

```
maxDataSize=auto_high_volume
```

```
[volume:hot]
path=/h01
maxVolumeDataSizeMB=2150400

[volume:cold]
path=/c01
maxVolumeDataSizeMB=2150400
```

13. Once you have configured **indexes.conf** with the index information, they can be deployed into all indexers using the following splunk command.

```
[splunk@splunk-admin1 ~]$ /opt/splunk/bin/splunk apply cluster-bundle -auth
admin:splunk
```

APPENDIX E: UCS CLI TO START AND SHUTDOWN SERVERS

You can use the following sample CLI commands to start and shutdown Cisco UCS servers.

1. Login to the Fabric Interconnect through SSH using admin user.

Startup commands

```
scope org /
scope service-profile <SPLUNK-IDX-NODE-01>
power up
commit-buffer
exit
scope service-profile <SPLUNK-IDX-NODE-02>
power up
commit-buffer
exit
```

Shutdown commands

```
scope org /
scope service-profile <SPLUNK-IDX-NODE-01>
power down
commit-buffer
exit
scope service-profile <SPLUNK-IDX-NODE-02>
power down
commit-buffer
exit
```

ABOUT THE AUTHOR

Somu Rajarathinam is the Pure Storage Solutions Architect responsible for defining application solutions based on the company's products, performing benchmarks, and developing reference architectures for applications on Pure.

Somu has over 20 years of database experience, including as a member of Oracle® Corporation's Systems Performance and Oracle Applications Performance Groups. His career has also included assignments with Logitech®, Inspirage®, and Autodesk®, ranging from providing database and performance solutions to managing infrastructure, to delivering database and application support, including for Splunk, both in-house and in the cloud.



© 2017 Pure Storage, Inc. All rights reserved. Pure Storage, FlashStack, Evergreen, and the "P" Logo are trademarks or registered trademarks of Pure Storage, Inc. in the U.S. and other countries. Cisco, Cisco UCS, and Nexus are registered trademarks of Cisco in the U.S. and other countries. Splunk and Splunk Enterprise are registered trademarks of Splunk in the U.S. and other countries. Red Hat Enterprise Linux is the registered trademark of Red Hat, Inc. in the U.S. and other countries. The Pure Storage product described in this documentation is distributed under a license agreement and may be used only in accordance with the terms of the agreement. The license agreement restricts its use, copying, distribution, decompilation, and reverse engineering. No part of this documentation may be reproduced in any form by any means without prior written authorization from Pure Storage, Inc. and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

ps_ra_flashstack-for-splunk_01



Pure Storage, Inc.

Twitter: [@purestorage](https://twitter.com/purestorage)

www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088

F: 650-625-9667

Sales: sales@purestorage.com