ES-024 - Elastic - ElasticSearch System Administrator Guide

January 12, 2024
OA DCGS

AFLCMC/HBGEB
AF DCGS Engineering
235 Byron Street, Suite 19A Robins AFB, GA 31098

Controlled by: AFLCMC/HBGEB
Controlled by: AFRL/RIEB
CUI Category: Defense, CTI
Distribution/Dissemination Controls: Dist. E
POC: AFRL.RIE.OADCGSCM@us.af.mil

CHANGE LOG

This record shall be maintained throughout the life of the document. Each published update shall be recorded. Revisions are a complete re-issue of the entire document. A revision shall be made annually or when applicable. Refer questions concerning this document to: steven.truxal.ctr@us.af.mil

CHANGE / REVISION RECORD

Revision	Date	Description of Change	Made By
Initial	11/02/2021	Initial Release CR-2021-OADCGS-078	S. Truxal/AFRL/ES
		Tech Edit	J. Smith/AFRL/ES
A	11/16/2021	PKI Information Added	S. Truxal/AFRL/ES
B	8/16/2022	Service Account Password Updates	S. Truxal/AFRL/ES
C	8/16/2022	Adding OS Patches	S. Truxal/AFRL/ES
D	1/12/2024	Updates for 8.11 release	S. Truxal

TABLE OF CONTENTS

1 Introduction 1
1.1 Overview of Software/System 1
1.2 System Environment 5
1.2.1 Minimum Hardware Requirements 5
1.2.1.1 Elastic Search Cluster 5
1.2.1.1.1 15 Node Cluster – Large (Production High & Low) 6
1.2.1.1.2 10 Node Cluster – Medium (CTE & MTE) 7
1.2.1.1.3 37 Node Cluster – Small (REL) 7
1.2.1.2 Logstash Nodes 8
1.2.1.3 REL 8
1.2.2 Elastic Repo 8
1.2.3 Puppet 9
1.2.3.1 Elastic profiles 9
1.2.3.1.1 Elastic Servers – dsil_elastic_servers Module 10
1.2.3.1.2 Elastic Clients – dsil_elastic_clients Module 10
1.2.3.1.2.1 dsil_elastic_clients::setup_beat Function 11
1.2.3.2 Puppetfile 12
1.2.3.3 Elastic Node Groups (Classifications) 13
1.2.4 SCCM 16
1.3 Minimum Software Requirements 17
1.4 Site Requirements 17
2 System Administration 18
2.1 Basic Concepts 18
2.1.1 Near Real Time (NRT) 18
2.1.2 Cluster 18
2.1.3 Node 18
2.1.4 Index 18
2.2 Elastic Terminology and Setup on OA 20
2.2.1 Data Ingest 20
2.2.1.1 Logstash Pipelines 21
2.2.1.2 Ingest Pipelines 21
2.2.2 Collecting Monitoring Data with Metricbeat 22
2.2.3 Monitoring of Domain Controllers 22

2.2.4 Health and Status Monitoring 22
2.2.4.1 Infrastructure Monitoring 24
2.2.3.1.1 Infrastructure Switch Symptoms 26
2.2.4.2 Host/Application Monitoring 27
2.2.4.3 Event Collection 37
2.2.5 Metricbeat vSphere Data Collection 38
2.2.6 Heartbeat 39
2.2.6.1 Heartbeat Monitors 39
2.3 Daily Checks 40
2.3.1 Cluster Health 40
2.3.2 Node Status 43
2.3.3 Logstash Nodes Running Properly and Data Ingested Properly 43
2.3.3.1 Verify All Logstash Instances Talking to Cluster 43
2.3.3.2 Verify Expected Data Types Being Ingested 44
2.3.4 Disk Space Issues 45
2.3.5 elasticDataCollector 46
2.3.5.1 Verify elasticDataCollector Service Running on All Logstash Instances 46
2.3.5.2 Verify elasticDataCollector Is Not Generating Critical Errors or Exceptions 47
3 PKI Certificates 50
3.1 Obtain new PKI Certificates 50
3.1.1 Elastic Certificates (includes Kibana) 50
3.1.2 Logstash Certificates 51
3.2 Certificate Authority (CA) changed 52
3.2.1 Distribute new CA 52
3.2.2 Restarting Beats on Linux Hosts 53
3.3 Updating certificates on Elastic hosts 58
3.4 Updating certificates on Logstash hosts 59
3.5 Check for errors 59
4 Elastic Service account 60
4.1 Updating the service account password for a DCGS site 60
4.1.1 SAKM keytab updates on all hosts 60
4.1.2 Service account password updates on Logstash VMs 61
4.1.2.1 Elastic Data Collector 61
4.1.2.2 Metricbeat Keystore 67

4.1.2.3 Logstash Keystore 67
5 Troubleshooting 69
5.1 Kibana Troubleshooting 69
5.1.1 Web Page Does Not Appear or is Not Working Properly 69
5.1.2 Grok Debugger Gives Error “Something Went Wrong” 69
5.2 Elasticsearch (Cluster) Troubleshooting 70
5.2.1 Verify Elasticsearch is Running on a Node 70
5.2.2 Missing Elastic Node 71
5.2.3 Check Elastic Cluster Health from Linux Command Line 72
5.2.3.1 Display Cluster Health 73
5.2.3.2 List Nodes of Cluster 73
5.2.3.3 List Indexes in Cluster 73
5.2.4 Unassigned Shards 73
5.2.5 ILM Issues 75
5.2.5.1 Attribute box_type causing ILM to fail 76
5.3 Logstash Troubleshooting 76
5.4 Elastic Data Collector Extended Details 76
5.5 Metricbeat Script Processor 78
6 Node Maintenance 79
6.1.1 Rebooting an Elastic Node 79
6.1.2 Shut Down Entire Cluster 80
6.1.3 Adding OS Patches on an Elastic Node 81
6.1.4 Decommission an Elastic Node 81
6.1.5 6.1.5 Stop ILM for Old Indexes 81
6.1.5.1 Determine if Any Indexes are Candidates 81
6.1.5.2 6. Delete Old Write Index 83
7 Useful Commands from Kibana Console 86
7.1 DOD PKI Certificates 87
7.2 License Expiration 87
7.3 Accounts and Passwords 87
7.3.1 Elastic Certificates (includes Kibana) 88
7.3.3 Logstash Certificates 89
7.4 Exporting Beat Templates for Installation 91
7.5 Configure NSX Load Balancer 91

7.6 Service Account Kerberos Management (SAKM) 92
7.7 Setup Repo with Core RPMs 93
7.7.1 Verify Service Account (From Each VM) 94
7.7.2 Elasticsearch Install – Adding a Node 94
7.7.2.1 Verify VM can see Elastic Repo 95
7.7.2.2 Verify SSL Settings for ElasticSearch 95
7.7.2.3 Add License for Elasticsearch 96
7.7.3 Kibana 96
7.7.3.1 Test Kibana (For Each Kibana Node, If Applicable) 97
7.7.3.2 Disable Telemetry (On One Kibana Node) 99
7.7.3.3 Audit Settings 100
7.7.3.3.1 Verify Audit Settings 100
7.7.3.4 Configure Index Lifecycle Management (ILM) 102
7.7.3.4.1 Index Template for ILM 102
7.7.3.4.2 Verify Indexes are Moving Through Lifecycle 104
7.7.3.5 Adjust Concurrent Rebalancing (optional) 105
7.7.3.6 Memory Lock Check 105
7.7.4 Logstash 106
7.7.4.1 Verify Logstash 106
7.8 Secure Elastic with Break-Glass Password 108
7.8.1 Verify Role Mappings 109
7.8.2 Verify Roles 109
7.8.3 Remove Unneeded Accounts 110
7.8.4 Verify Beat Templates are Loaded 112
7.8.5 Database Queries 112
7.8.5.1 SCCM Database 114
7.8.5.2 IDM Database 115
7.8.5.3 SQL Server Statistics 116
8 System Dashboards 118
8.1 Status Counts 118
8.2 Syslog & Windows Events 119
8.3 App Status & Problem Hosts 119
8.4 Problem Apps & Devices 120
8.5 Metricbeat Host CPU & Memory 121

8.6 Groups 121
8.7 Puppet Agent 121
9 References 122

List of Figures

Figure 1 Elastic Cluster Configuration at ECH/WCH Hub 2
Figure 2 Data collected from sites is sent to the hub 3
Figure 3 Collectors at each site send their data to Logstash at the site 4
Figure 4 Elastic repo layout 9
Figure 5 Elastic Classifications 13
Figure 6 Node Matching Rules 13
Figure 7 profile::elastic_clients 14
Figure 8 Allow Elastic Servers Puppet module to run on all Elastic and Logstash Servers 15
Figure 9 profile::elastic_servers 16
Figure 10 ElasticDataCollector Overview 23
Figure 11 Infrastructure Monitoring Objects 24
Figure 12 Device Class Architecture 25
Figure 13 Example of running infrastructure threads 26
Figure 14 Host/Application Health Objects 28
Figure 15 Default metricbeat.yml file for Windows 29
Figure 16 metricbeat.yml application monitoring definition 30
Figure 17 Advance Process Monitoring 31
Figure 18 Example of appmonitor data appended to "process_summary" document 32
Figure 19 Overview of Health Document Creation 33
Figure 20 Flow A, Creation of App-Host Documents 34
Figure 21 Flow B, Watcher Thread 34

Figure 22 Flow C, Querier Thread 35
Figure 23 Flow D, elasticDataCollector Main Thread 35
Figure 24 Existing Heartbeat Host Monitoring 36
Figure 25 appsconfig.ini header 36
Figure 26 group.ini file header 37
Figure 27 vCenter Audit Event Collection 38
Figure 28 Verify Heartbeat data is received 40
Figure 29 Elastic Stack Monitoring *Cluster Listing* 41
Figure 30 Elastic 7.16 Stack Monitoring tab showing a Healthy Status 42
Figure 31 Elastic 7.16 Node Status 43
Figure 32 Example visual to monitor Metricbeat events 44
Figure 33 Example filter for Elastic OR Logstash hosts 45
Figure 34 turning off KQL syntax for queries 45
Figure 35 Example visual to monitor elasticDataCollector service on Logstash VMs 47
Figure 36 Example saved search to monitor Python applications on Logstash VMs 48
Figure 37 - Access Elastic Clients Module 54
Figure 38- Select classes for Elastic Clients 54
Figure 39- Select restart_beats parameter 55
Figure 40- Set restart_beats to "true" and select "Add node to group" 55
Figure 41- Select "Commit 1 change" 56
Figure 42- restart_beats parameter is set to "true" 56
Figure 43- Search for bad_certificate using Kibana 57
Figure 44- Remove restart_beats parameter 58
Figure 45 - Commit removal of restart_beats parameter 58
Figure 46 X11 Forwarding Setup Successfully 62

Figure 47 xauth list example with logstash host 63
Figure 48 xauth add <cookie> 64
Figure 49 Device Configuration GUI 64
Figure 50 Update password 65
Figure 51 Publish Device Configuration 65
Figure 52 Successful Publish 66
Figure 53 Unsuccessful Publish 66
Figure 54 Figure 10 Set httpd_sys_content_t 93
Figure 55 Figure 11 df command 95
Figure 56 Figure 13 Login Screen 98
Figure 57 Figure 14 Explore on my own 98
Figure 58 Figure 15 Stack Management 99
Figure 59 Figure 16 Disable Telemetry 100
Figure 60 Figure 19 Dev Tools 101
Figure 61 Figure 23 Index Management 104
Figure 62 Figure 24 Lifecycle phase 105
Figure 63 Figure 27 Logstash 107
Figure 64 Figure 28 Nodes tab 107
Figure 65 Figure 29 Pipelines tab 108
Figure 66 Figure 30 Edit User 111
Figure 67 Figure 31 Disabled accounts 112
Figure 68 Figure 56 Errors 113
Figure 69 Figure 57 Edit SCCM Pipeline Example 114
Figure 70 Figure 58 Verify SCCM data is received 115
Figure 71 Figure 59 Edit IDM Pipeline Example 115

Figure 72 Figure 60 idm-* indexes 116
Figure 73 Figure 61 Verify SQL server data is received 117
Figure 74 *SYSTEM Dashboard* 118
Figure 75 *SYSTEM Dashboard - Status Counts* 118
Figure 76 *SYSTEM Dashboard - Data Collector Detailed Status* 119
Figure 77 *SYSTEM Dashboard - Syslog & Windows Events* 119
Figure 78 *SYSTEM Dashboard - App Status & Problem Hosts* 120
Figure 79 *SYSTEM Dashboard - Problem Apps & Devices* 120
Figure 80 *SYSTEM Dashboard* - Metricbeat Host CPU & Memory 121
Figure 81 *SYSTEM Dashboard* - Groups 121
Figure 82 *SYSTEM Dashboard – Puppet Agent* 121

List of Tables

Table 1 Ports used for data ingest by Elastic 4
Table 2 Cluster Hardware Requirements at Hubs (15 Nodes) 6
Table 3 Cluster Hardware Requirements at Hubs (10 Nodes) 7
Table 4 Cluster Hardware Requirements at Hubs (37 Nodes) 8
Table 5 Logstash Hardware Requirements at Sites in Production and Test Environments 8
Table 6 Logstash Hardware Requirements at Sites for REL 8
Table 7 Logstash Pipelines 21
Table 8 Ingest Pipeline Changes 22
Table 9 Elastic Nodes where Kibana is installed 69
Table 10 Browser Error 70
Table 11 Kibana Console Commands 86

1 Introduction

The purpose of this document is to aid in the sustainment of Elastic for the DCGS OA system. This document can be used as a guide for daily maintenance and troubleshooting of Elastic on the OA system.

Any lines in this document that begin with a # symbol are command line examples. Commands where **{}** are used indicate a variable or context sensitive option such a {username} or {password}; do not include the {} in the command unless noted to do so.

1.1 Overview of Software/System

Elastic Search is used on the OA system to collect log data, audit information, metrics, and other forms of data. Data is collected from every site and sent to the Elastic Cluster that resides at the hub. Once data is ingested, it is available for visualization. The data can be viewed in its raw form or in charts, graphs, etc.

Elastic is comprised of the following components:

- **Elasticsearch cluster:** One or more nodes (servers) collectively become a cluster to hold all collected data, provide indexing and search capabilities. https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
- **Kibana:** Web client for users that provides visualization of collected data. https://www.elastic.co/guide/en/kibana/current/index.html
- **Logstash:** A server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and sends it into the Elastic cluster. https://www.elastic.co/guide/en/logstash/current/index.html
- **Beats Collectors:** Lightweight, purpose-built agents that acquire data and feed it into Elasticsearch.
    - **Metricbeat:** Used to collect metric data from hosts. https://www.elastic.co/guide/en/beats/metricbeat/current/index.html
    - **Filebeat:** Used to collect log data from hosts. https://www.elastic.co/guide/en/beats/filebeat/current/index.html
    - **Heartbeat:** Used for uptime monitoring of hosts, websites, and other endpoints. https://www.elastic.co/guide/en/beats/heartbeat/current/index.html
    - **Winlogbeat:** Used to collect windows event logs. https://www.elastic.co/guide/en/beats/winlogbeat/current/index.html

Elasticsearch clusters reside at the East Coast Hub (ECH) and the West Coast Hub (WCH). ECH is the primary cluster, WCH provides failover.

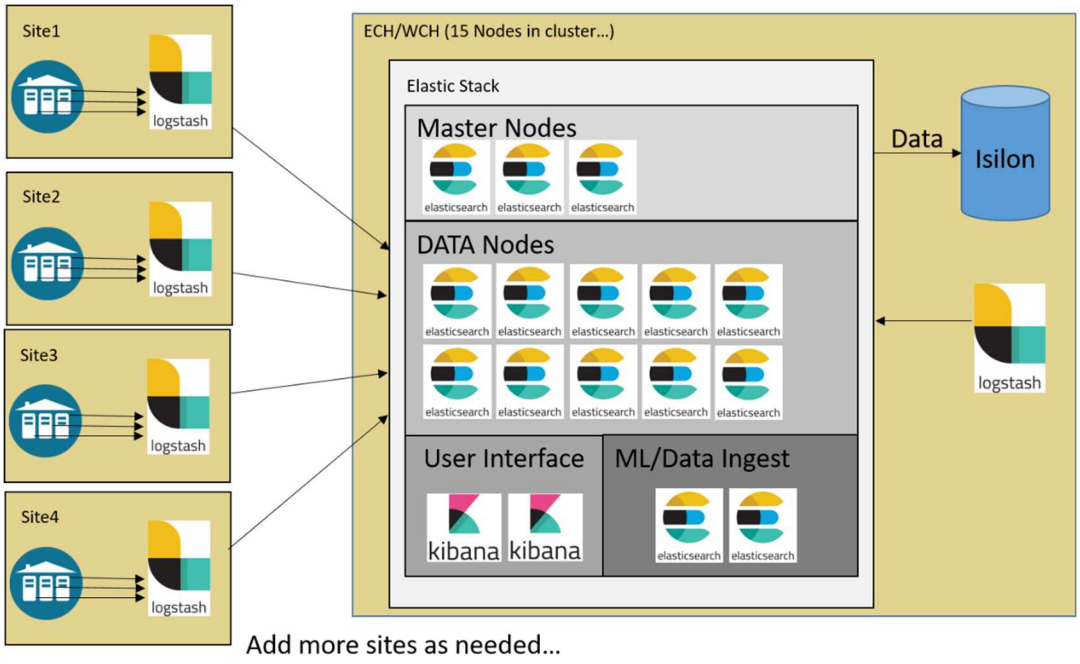


Figure 1 Elastic Cluster Configuration at ECH/WCH Hub

Each cluster may be made up of the following types of specialized nodes:

- **Master Eligible Node:** The master node is used to perform administrative tasks and control the cluster. The master node tracks the availability/failure of the data nodes and is responsible for creating and deleting indices. All the master-eligible nodes participate in an election process to elect the current master node. If at any time the master node goes offline, a new master is elected from the remaining master-eligible nodes.
- **Data Node:** The data nodes hold all the ingested data and perform data-related operations such as Create, Read, Update, and Delete (CRUD), search, and aggregations. In our current implementation all data nodes are also coordinating nodes.
- **Coordinating Node:** Requests such as search requests or bulk-indexing requests may involve data held on different nodes. A search request, for example, is executed in two phases, which are coordinated by the node that receives the client request. The coordinating node forwards the request to the data nodes which hold the data. Each data node executes the request locally and returns its results to the coordinating node. The coordinating node reduces each data node's results into a single global result set. In our current implementation, all data nodes also serve as coordinating nodes.
- **Ingest Node:** The ingest nodes can apply an ingest pipeline to a document in order to transform and enrich the document before indexing.
- **Machine Learning (ML) Node:** The machine learning nodes are used to perform all machine learning jobs for the cluster.

- **Transform Node**: Transforms enable you to convert existing Elasticsearch indices into summarized indices, which provide opportunities for new insights and analytics. For example, you can use transforms to pivot your data into entity-centric indices that summarize the behavior of users or sessions or other entities in your data. Or you can use transforms to find the latest document among all the documents that have a certain unique key.

More information on nodes can be found here:
https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

Data collected from each DCGS site is fed to a Logstash instance at that site. The Logstash instance then performs ingest processing on the data, tags it with the site name, and forwards it into the cluster for analysis and visualization, as demonstrated in the following figure.

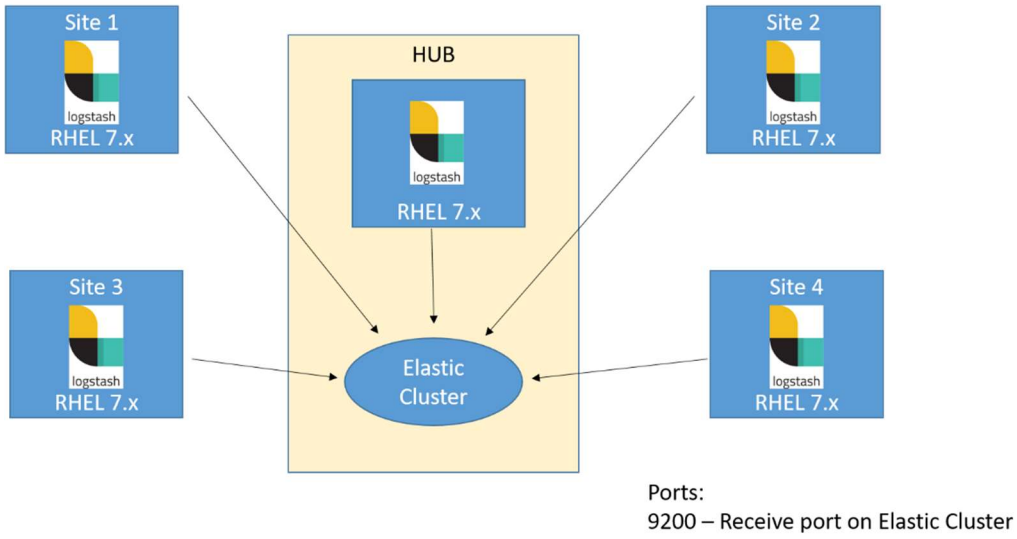


Figure 2 Data collected from sites is sent to the hub

Beats and other collection tools are used to gather all the site's data and send it to a Logstash instance that resides at the site.

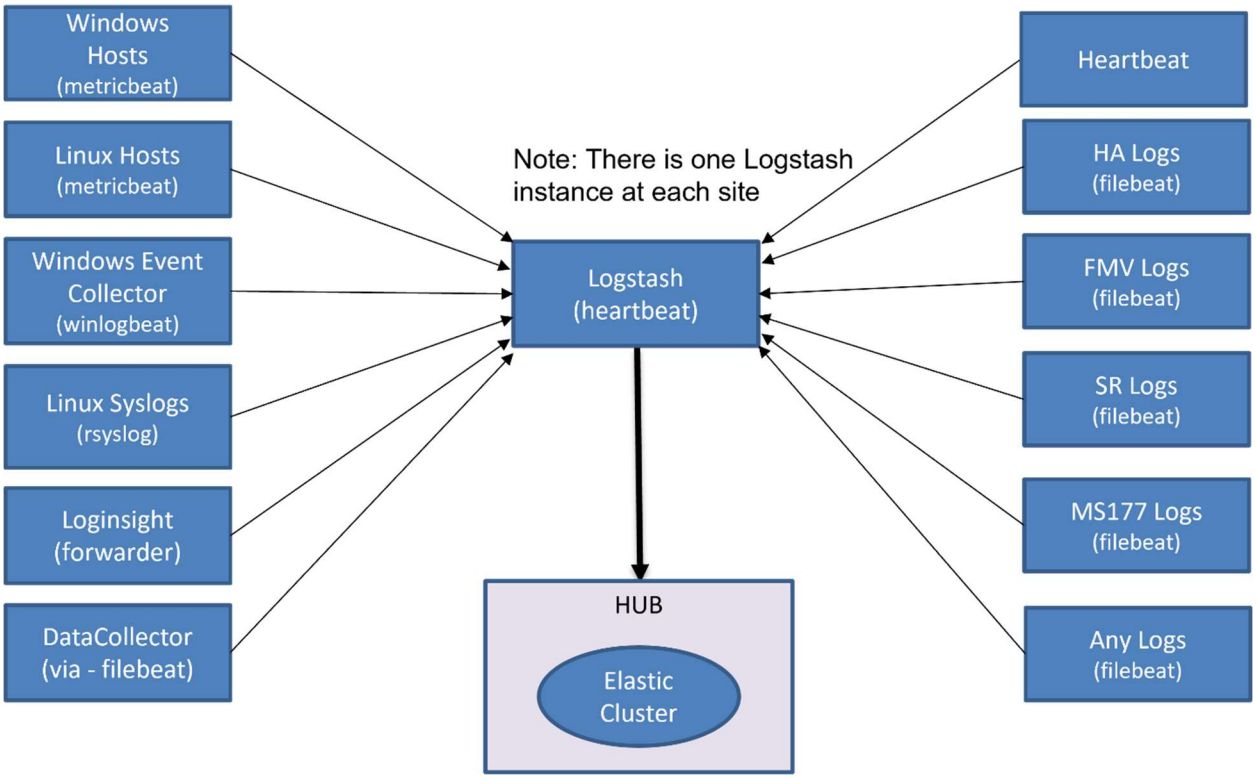


Figure 3 Collectors at each site send their data to Logstash at the site

Ports used for data ingest by Elastic are shown in the following table. A Logstash at each site listens for data on these ports.

Table 1 Ports used for data ingest by Elastic

PORT	Data Type
5040	Future Use
5041	Linux Syslog (rsyslog)
5042	Filebeat-Logstash Only
5043	Filebeat-Singleworker
5044	Filebeat-Generic
5045	Winlogbeat
5046	HBSS EPO Metrics
5047	Heartbeat
5048	Metricbeat
5049	HBSS EPO data
5050	Log Insight

5051	ASHES
5052-5060	

1.2 System Environment

The environment required to run Elastic will be described in this section.

1.2.1 Minimum Hardware Requirements

1.2.1.1 Elastic Search Cluster

Optional Cluster Sizes – This document is intended to be used for different Elastic Cluster sizes. For the VM requirements, review the tables in this section specific to the cluster size.

As Elastic clusters grow, nodes become specialized. The following list describes the roles nodes may have. These abbreviations are used to define node roles in the Cluster configurations that follow. See https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html for more information.

- **Master-eligible node (m):** The master node is responsible for lightweight cluster-wide actions such as creating or deleting an index, tracking which nodes are part of the cluster, and deciding which shards to allocate to which nodes. It is important for cluster health to have a stable master node.
- **Content Data node (s):** Content data nodes are part of the content tier. Data stored in the content tier is generally a collection of items such as a product catalog or article archive. Unlike time series data, the value of the content remains relatively constant over time, so it doesn't make sense to move it to a tier with different performance characteristics as it ages. Content data typically has long data retention requirements, and you want to be able to retrieve items quickly regardless of how old they are.
- **Hot Data node (h):** Hot data nodes are part of the hot tier. The hot tier is the Elasticsearch entry point for time series data and holds your most-recent, most-frequently-searched time series data. Nodes in the hot tier need to be fast for both reads and writes, which requires more hardware resources and faster storage (SSDs). For resiliency, indices in the hot tier should be configured to use one or more replicas.
- **Warm Data node (w):** Warm data nodes are part of the warm tier. Time series data can move to the warm tier once it is being queried less frequently than the recently-indexed data in the hot tier. The warm tier typically holds data from recent weeks. Updates are still allowed, but likely infrequent. Nodes in the warm tier generally don't need to be as fast as those in the hot tier. For resiliency, indices in the warm tier should be configured to use one or more replicas.
- **Cold Data node (c):** Cold data nodes are part of the cold tier. When data is no longer searched regularly, it is moved from the warm tier to the cold tier. While still searchable, this tier is typically optimized for lower storage costs rather than search speed. For better storage savings, data is kept in fully mounted indices of searchable snapshots on the cold tier. Unlike regular indices, these fully mounted indices don't require replicas for reliability. In the event of a failure, they can recover data from the underlying snapshot instead. This halves the local storage needed

for the data. Snapshot repositories on the Isilon are used for the fully mounted indices in the cold tier. Fully mounted indices are read-only.

- **Frozen Data node (f):** Frozen data nodes are part of the frozen tier. Once data is no longer being queried, or being queried rarely, it is moved from the cold tier to the frozen tier where it stays for the rest of its life. The frozen tier requires a snapshot repository. The frozen tier uses partially mounted indices to store and load data from a snapshot repository. This reduces local storage and operating costs while still letting you search frozen data. Because Elasticsearch must sometimes fetch frozen data from the snapshot repository, searches on the frozen tier are typically slower than on the cold tier. Frozen data nodes use a local cache to store the most-used portions of all frozen data; data in the cache provides faster searching than fetching from the snapshot repository.
- **Ingest node (i):** Ingest nodes can execute pre-processing pipelines, composed of one or more ingest processors. Depending on the type of operations performed by the ingest processors and the required resources, it may make sense to have dedicated ingest nodes, that will only perform this specific task.
- **Machine learning node (l):** Machine learning nodes run jobs and handle machine learning API requests. For more information, see Machine learning settings.
- **Transform node (t):** A node that has the `transform` role. If you want to use transforms, there must be at least one transform node in your cluster.
- **Remote Cluster Client (r):** A remote-eligible node acts as a cross-cluster client and connects to remote clusters. Once connected, you can search remote clusters using cross-cluster search. You can also sync data between clusters using cross-cluster replication. On Production, cross-cluster search/replication is set up between ECH and WCH.

1.2.1.1.1 15 Node Cluster – Large (Production High & Low)

The clusters for Elastic reside at the hubs. Each cluster will initially be made up of 15 nodes with the minimum configuration described in the following table. Kibana will be installed on elastic-nodes 10 and 15 in this configuration.

NOTE: The Elastic license will change over time; there are currently 15 nodes in the Elastic cluster but there can be more or fewer. The size of the cluster may change based on the amount of data ingested and stored.

Table 2 Cluster Hardware Requirements at Hubs (15 Nodes)

VM Description	OS	VCPUs	RAM (GB)	Disk 0	Disk 1	Data Dir	Role
elastic-node-1	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-2	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-3	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-4	Linux	8	64	*OS 90 GB*	Data - 100GB(SSD)	/ELK-local	lrst
elastic-node-5	Linux	8	64	*OS 90 GB*	Data - 100GB(SSD)	/ELK-local	lrst
elastic-node-6	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs
elastic-node-7	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs
elastic-node-8	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs
elastic-node-9	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs

elastic-node-10	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-11	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-12	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-13	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-14	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-15	Linux	8	64	*OS 90 GB*	Data - 100GB(SSD)	/ELK-nfs	f

1.2.1.1.2 10 Node Cluster – Medium (CTE & MTE)

The clusters for Elastic reside at the hubs. Each cluster will initially be made up of 10 nodes with the minimum configuration described in the following table. Kibana will be installed on elastic-nodes 7 and 10 in this configuration.

NOTE: The Elastic license will change over time; there are currently 10 nodes in the Elastic cluster but there can be more or fewer. The size of the cluster may change based on the amount of data ingested and stored.

Table 3 Cluster Hardware Requirements at Hubs (10 Nodes)

VM Description	OS	VCPUs	RAM (GB)	Disk 0	Disk 1	Data Dir	Role
elastic-node-1	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-2	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-3	Linux	4	8	*OS 90 GB*	N/A	/ELK-local	mr
elastic-node-4	Linux	8	64	*OS 90 GB*	Data - 100GB(SSD)	/ELK-local	Lrst
elastic-node-5	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs
elastic-node-6	Linux	8	64	*OS 90 GB*	Data - 2TB(SSD)	/ELK-local	hirs
elastic-node-7	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-8	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-9	Linux	8	64	*OS 90 GB*	None(nfs)	/ELK-nfs	cisw
elastic-node-10	Linux	8	64	*OS 90 GB*	Data - 100GB(SSD)	/ELK-nfs	f

1.2.1.1.3 7 Node Cluster – Small (REL)

The clusters for Elastic reside at the hubs. Each cluster will initially be made up of 7 nodes with the minimum configuration described in the following table. Kibana will be installed on elastic-nodes 5 and 6 in this configuration.

NOTE: The Elastic license will change over time; there are currently 7 nodes in the Elastic cluster but there can be more or fewer. The size of the cluster may change based on the amount of data ingested and stored.

Table 4 Cluster Hardware Requirements at Hubs (7 Nodes)

elastic-node-1	Linux	8	64	*OS 90 GB*	Data - 1TB(SSD)	/ELK-local	mlhis
elastic-node-2	Linux	8	64	*OS 90 GB*	Data - 1TB(SSD)	/ELK-local	mhis
elastic-node-3	Linux	8	64	*OS 90 GB*	Data - 1TB(SSD)	/ELK-local	mhis
elastic-node-4	Linux	4	64	*OS 90 GB*	None(nfs)	/ELK-nfs	ciw
elastic-node-5	Linux	4	64	*OS 90 GB*	None(nfs)	/ELK-nfs	ciw

elastic-node-6	Linux	4	64	*OS 90 GB*	None(nfs)	/ELK-nfs	ciw
elastic-node-7	Linux	4	64	*OS 90 GB*	Data – 100GB(SSD)	/ELK-local	f

1.2.1.2 Logstash Nodes

Each site will have one Logstash instance with the minimum configuration described in the following table.

1.2.1.2.1 Production (High & Low), CTE, and MTE

Table 5 Logstash Hardware Requirements at Sites in Production and Test Environments

VM Description	OS	VCPUs	RAM(GB)	Disk 0	Disk 1	Data Dir
Logstash	Linux	8	32	*OS 90 GB*	500GB	ELK-local

1.2.1.3 REL

Table 6 Logstash Hardware Requirements at Sites for REL

VM Description	OS	VCPUs	RAM(GB)	Disk 0	Disk 1	Data Dir
Logstash	Linux	4	32	*OS 90 GB*	500GB	ELK-local

1.2.2 Elastic Repo

There is an Elastic repo on the OA DCGS repo server where the RPMs are loaded for the installation. The RPMs are placed on the repo server in stages to prevent components from being installed/upgraded out of order. **Starting with this release Satellite will be used to sync repositories between sites.**

The following RPMs should be available on the fileserver after a completed installation. These RPMs are added to the repo server as needed during the installation process.

1. Elastic_Core_Components for the current version X.Y.Z (Example: 8.11.3)
    - elasticsearch-X.Y.Z-x86_64.rpm
    - kibana- X.Y.Z-x86_64.rpm
    - logstash- X.Y.Z-x86_64.rpm
    - elasticsearch-curator-5.8.4-1.x86_64.rpm
2. Elastic_Linux_Beats for the current version X.Y.Z
    - metricbeat- X.Y.Z-x86_64.rpm
    - filebeat- X.Y.Z-x86_64.rpm
    - heartbeat- X.Y.Z-x86_64.rpm

NOTE: Older versions of rpms may be present in the elastic directory on the repo server, the newest version will always be installed. Occasional housekeeping should be performed to remove older rpm files.

The following diagram shows the directory layout of the Elastic repository on the repo server. These directories will be populated when extracting the **oadcgs-es-elastic-reposerver- x.x.x.x.tar.gz** archive.

.

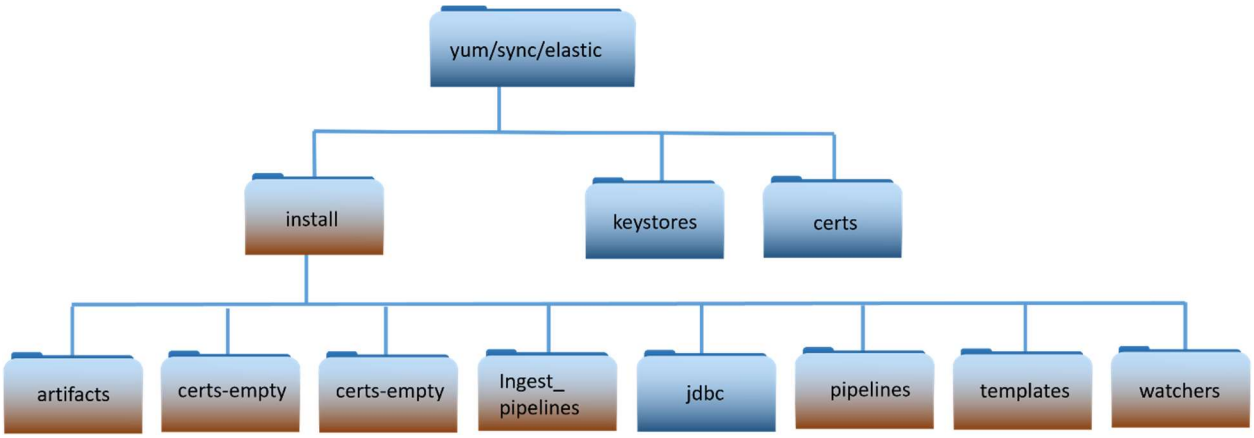


Figure 4 Elastic repo layout

See the README.txt file in the install directory for details on each directories contents.

1.2.3 Puppet

Puppet modules are used to automate some of the configuration on Linux hosts and the installation of some Elastic components. There are currently two modules added for Elastic, but minor modifications to the base OSIF configuration is also necessary during installation. A Puppet SME should have been involved in adding these modules and ensuring Puppet is configured properly for Elastic.

1.2.3.1 Elastic profiles

Execution of the Elastic modules is controlled through the Elastic profiles that should be present in each puppet branch. The following two files and contents should be present in one of the following directories (Dependent on version of OSIF)

- <branch>/site/profile/manifests
- <branch>site-modules/profile/manifests

elastic_clients.pp

class profile::elastic_clients (Boolean $install_beats=true, Boolean $restart_beats=false) {

```
  class { '::dsil_elastic_clients':
   install => $install_beats,
   restart_beats => $restart_beats,
  }
 }
```

elastic_server.pp

```
 class profile::elastic_servers {

   # call the module
   include dsil_elastic_servers

 }
```

See *ES-024 - Puppet - System Administrator Guide* for more information on puppet configurations.

1.2.3.1.1 Elastic Servers – dsil_elastic_servers Module

The **dsil_elastic_servers** module is used to configure Elastic, Kibana, and Logstash servers. The module opens the necessary ports, creates mounts, and performs other configuration tasks necessary to allow each component to run properly. Elasticsearch, Kibana, and Logstash are all installed manually, but they will not be able to run successfully unless the hosts are configured with this module.

The Heartbeat component of Elastic is automatically installed with an initial configuration on each Logstash instance by this Puppet module. This module then ensures that Heartbeat is running on each Logstash host.

1.2.3.1.2 Elastic Clients – dsil_elastic_clients Module

The **dsil_elastic_clients** module is used to automatically install Metricbeat and Filebeat on Linux hosts. Metricbeat is installed on all Linux hosts in the OA DCGS system. Filebeat is installed on all Elasticsearch and Logstash hosts to collect log files from the Elastic applications. This module also supplies a generic utility function **setup_beat** that can be used by any other Puppet modules to request that Filebeat be installed on hosts controlled by that module. OA DCGS ARTs and other Puppet clients that wish to have data gathered using Filebeat will use this function to install Filebeat on their hosts. The basic use of this module is described here; knowledge of Puppet is expected to understand the implementation. Please contact an Enterprise Services Elastic SME for questions or issues using the function.

1.2.3.1.2.1 dsil_elastic_clients::setup_beat Function

This is a generic function used by the **dsil_elastic_clients** module to install Metricbeat and Filebeat on Linux hosts. The function is used by the **dsil_elastic_clients** module to install Metricbeat on all Linux hosts and Filebeat on Elasticsearch and Logstash hosts. The function is also available and should be used by others to install ART-specific Filebeat modules. Each ART that integrates with Enterprise Elastic will use this function in their Puppet modules to request that Filebeat be installed on their hosts. The ART will either supply their own template file or coordinate with an Elastic SME to have one created for them. To

use the **setup_beat** function, a Filebeat template file holding the configuration for each host (or a group of hosts) must be supplied in the templates directory of the module calling the function.

To use this function from another Puppet module this module must be added to the dependencies of the calling module. Add the following to the **dependencies** section of the metadata.json file:

```
{"name":"modules/dsil_elastic_clients"}
```

The **setup_beat** function has the following signature:

```
dsil_elastic_clients::setup_beat($modname, String $beatname, String
$template_file, Array[String] $passed_modlist=[])
```

Parameter definitions:

1. **modname** – The built-in variable $module_name should be passed as the value for this parameter. This allows the function to identify the calling module.
2. **beatname** – The name of the Elasticsearch beat being installed. Currently the only supported value for clients is **filebeat**.

   NOTE: The **metricbeat** parameter is used by and is only valid for the **dsil_elastic_clients** module)

3. **template_file** – The template for the filebeat YML file. This can be specific for a particular host or it can be generic. The template file should be placed in the **templates/<beat>** folder and the format is **XXXX.<beat>.yml.epp** where XXXX is all characters following the first 7 characters of the hostname up to the first dot of the hostname where **filebeat** is to be installed. **<beat>** is the name of the beat being installed.

   Example: ha01.filebeat.yml.epp

   NOTE: This is generally 4 characters but there are some instances where it may be more. For example, if the hostname is u00su01bigname.ech.dcgs.mil then the template would be name **bigname.filebeat.yml.epp**.

   The same template may be used for multiple hosts, in which case a generic name should be given. For example, if using the same template for all “ha” hosts, one would name the template **ha.filebeat.yml.epp**.

   The important thing to understand here is that the **setup_beat** function will look for the file name as **<template_file>.yml.epp**. If using a generic template named **ha.filebeat.yml.epp**, the parameter would be **ha.filebeat**.

   NOTE: Be sure to include the Logstash output in the template:

```
---------------------------- Logstash output --------------------------------

output.logstash:
    hosts: ["logstash-<%= $facts[hostname][0,3] %>:5044"]
```

```
ssl.certificate_authorities: ["/etc/pki/tls/certs/ca-bundle.crt"]
```

4. **beat_modules** – Array of modules to install for the beat. Module names are in the format **[desc].module.[module name]**.

   Example: ls01.module.netflow

   The **setup_beat** function will install modules passed into the function but will also look for any modules that match the name of the host where Filebeat is being installed. For an automated module, detection module configurations can be specified by placing module files in the **templates/<beat>** folder. The module filename format follows the same format referenced previously, **[XXXX].module.[module name].yml.epp** where XXXX is all the characters following the first 7 characters of the hostname up to the first dot of the hostname where the module is to be installed.

   Examples: ha01.module.activemq.yml.epp, bigname.module.iis.yml.epp

1.2.3.2 Puppetfile

The Puppetfile must be updated with the elastic modules for them to be included in the branch when it is pushed.

On each puppet branch, edit the **<branch>/Puppetfile** and lines for each of the elastic modules.

```
mod 'dsil_elastic_clients',
    :git => 'git@{xxx}su01pup1.ech.dcgs.mil:dsil_elastic_clients.git',
    :tag => 'v1.1.XX'
mod 'dsil_elastic_servers',
    :git => 'git@{xxx}su01pup1.ech.dcgs.mil:dsil_elastic_servers.git',
    :tag => 'v1.1.XX'
```

The value for **:git =>** is installation dependent; copy the value from another module in the file.

The **:tag** value is also dependent on the system and can be obtained by running the following command in the modules branch:

```
# git tag -l
```

NOTE: Ensure Elastic repo is using correct RPM signing key, the Elastic RPMs are signed with the elastic.key (e.g. common.yaml should contain .../yum /elastic/elastic.key, not .../yum /dcgs.key).

To see what keys have been imported by rpm on any machine execute the following command as root:

```
# rpm -qa -qf '%{VERSION}-%{RELEASE} %{SUMMARY}\n' gpg-pubkey*
```

1.2.3.3 Elastic Node Groups (Classifications)

Node Groups (Classifications in older puppet versions) for Elastic are configured in the Puppet web console. The rules in each node group should be set up properly to ensure that the correct Elastic Puppet module is run on the correct VMs.

There is an Elastic Servers Node Group for Logstash and Elastic Nodes. There is an Elastic Clients Node Group for all other hosts.
Following is an example of what the classifications for Elastic should look like:

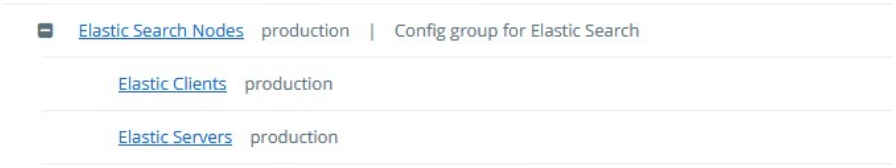


Figure 5 Elastic Classifications

The node matching rules for Elastic clients should include all Linux hosts. The following rule allows the Elastic Clients Puppet module to be run on all RedHat boxes.

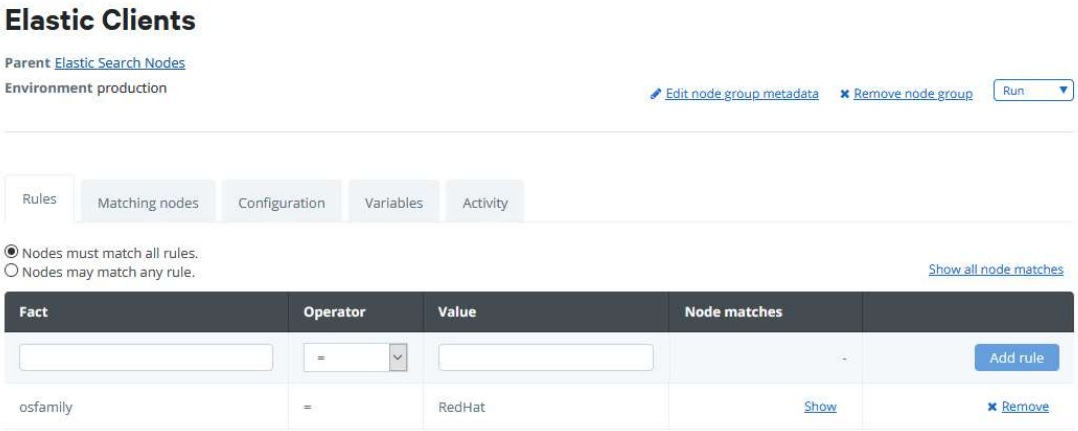


Figure 6 Node Matching Rules

The Elastic clients configuration should include the **profile::elastic_clients** class, as shown:

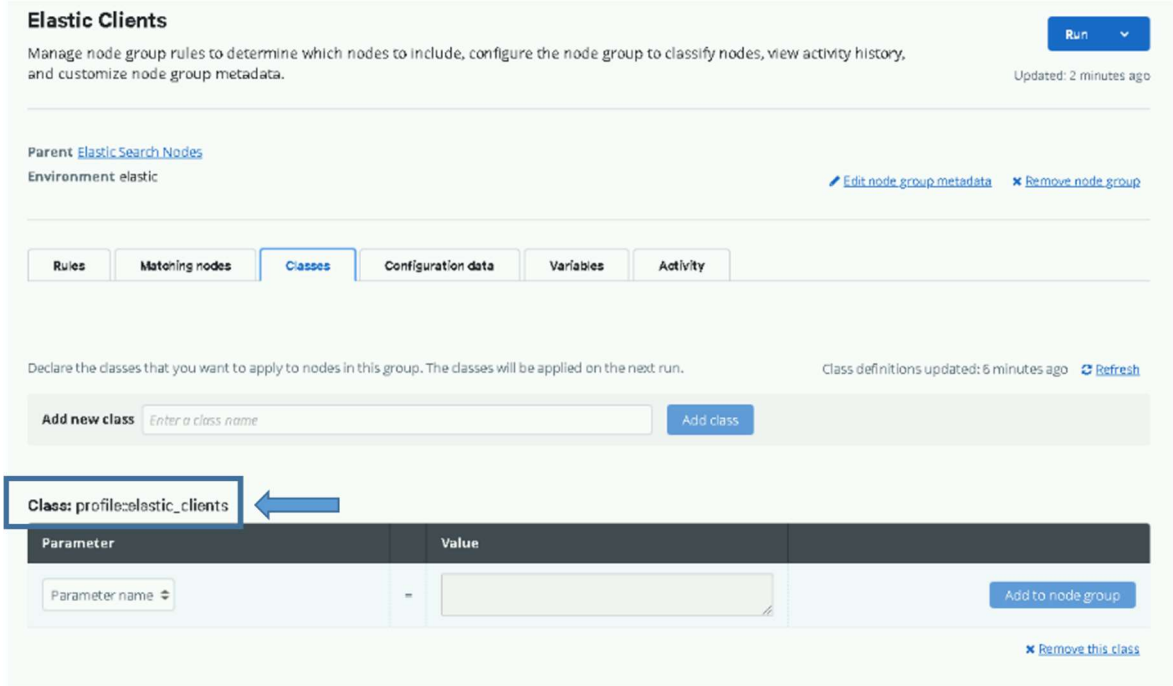


Figure 7 profile::elastic_clients

The node matching rules for Elastic servers should include all Logstash and Elastic VMs. The following rules allow the Elastic Servers Puppet module to be run on all Elastic and Logstash Servers.

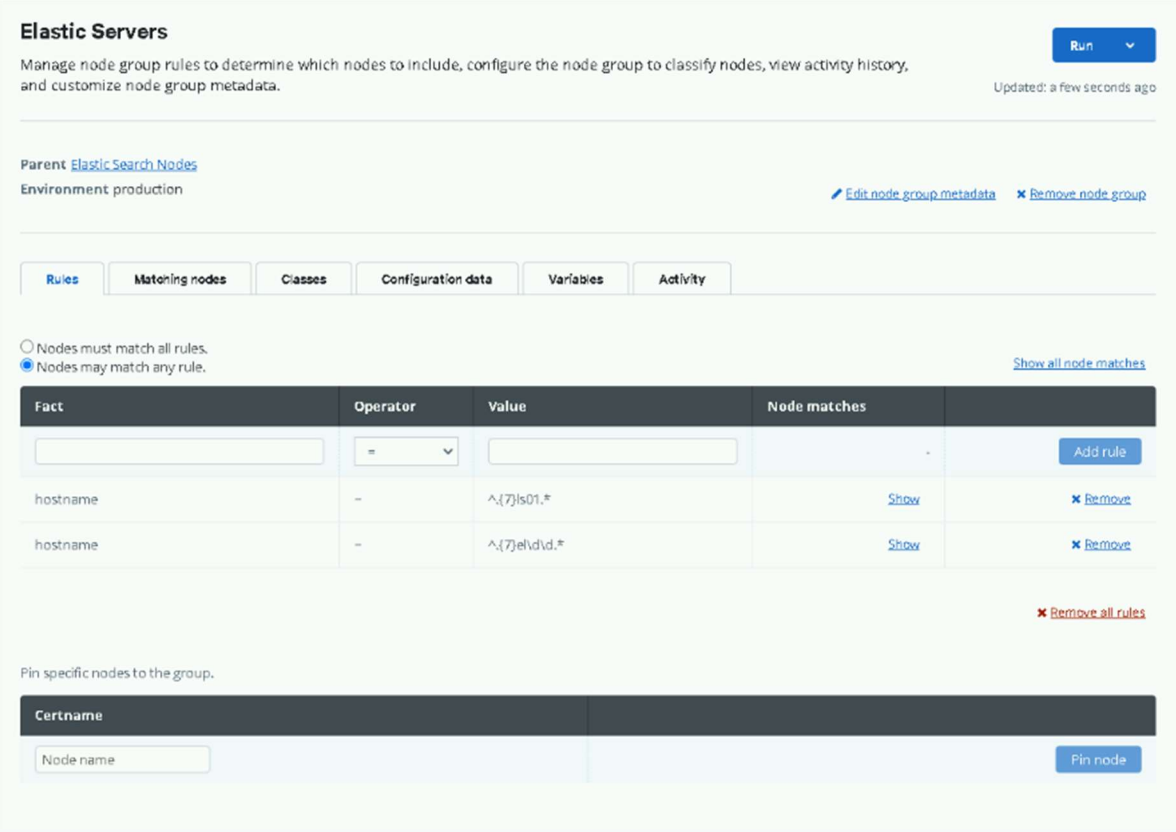


Figure 8 Allow Elastic Servers Puppet module to run on all Elastic and Logstash Servers

The configuration for Elastic servers should include the **profile::elastic_servers** class, as shown:

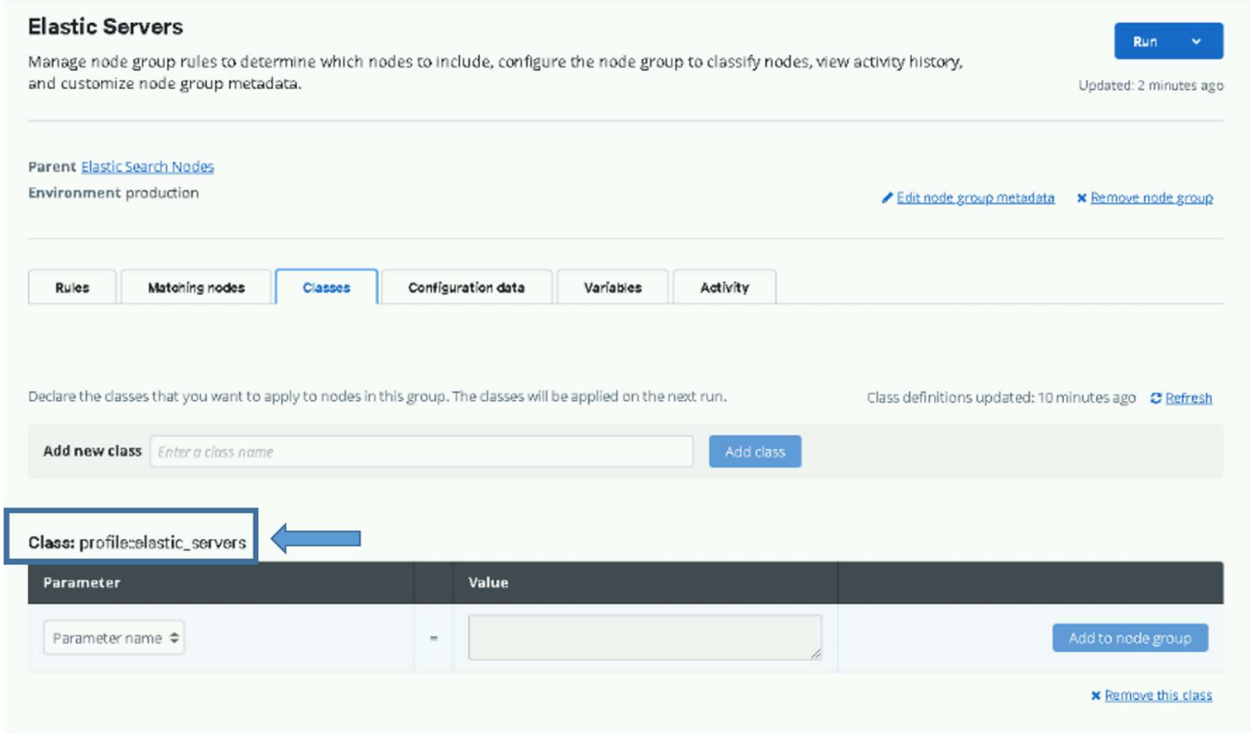


Figure 9 profile::elastic_servers

1.2.4 SCCM

Microsoft System Center Configuration Manager (SCCM) is used for all beats deployments on all windows entities and Puppet is used for all Linux entities.

To configure SCCM for Elasticsearch, refer to ES-018 – SCCM – Instructions for Building an SCCM Package to Install Beats for Windows.

When ARTs or other clients request that Filebeat be installed on their VM to collect logs, the configuration file specific to their application should be placed in the configs/filebeat directory of the SCCM package for Elastic. The config file should be named using the hostname minus the first 7 characters of the host where the Filebeat is to be installed. Filebeat modules can also be specified with the same naming convention.

For example, if a client has a filebeat.yml configuration file and an activemq module file for the host u00sm01cl01, the following files would be placed in the configs/filebeat directory:

- cl01.filebeat.yml (This is the filebeat.yml for the host.)
- cl01.module.activemq.yml (This is the activemq.yml file for the host.)

IMPORTANT NOTE: The output section of the configuration **output.logstash** in the filebeat.yml is automatically generated by the installation script and should not be included in the configuration file placed in the SCCM share.

After placing these files in the correct location in the package and redeploying, SCCM will automatically install and configure Filebeat on the **cl01** host.

1.3 Minimum Software Requirements

Each Elastic and Logstash VM should be installed with RedHat Linux version 7.3 or greater. After the installation of the Linux operating system, each node should be joined to Puppet to become a Puppet client, at which time Puppet will manage the configuration of each system.

Service Account Kerberos Management (SAKM) must then be installed on each VM and configured to sustain the Kerberos ticket for the Elastic service account for the site where the VM resides. Example (00_elastic.svc).

1.4 Site Requirements

Prior to installation, a Logstash VM must be provisioned at each site data is to be ingested from; see Section 1.2.1.1 for hardware requirements. The installation of all the Logstash software and all collection components at the site can be done remotely so no actual site presence is necessary.

2 System Administration

This document assumes the user is a Linux administrator, is a member of either the **ent Elastic Admins** or **ent Kibana Admins** Active Directory (AD) group and that the OA System is up and available. For assistance on the installation or upgrading of Elastic, refer to the Elasticsearch System Installation or Upgrade Instructions.

2.1 Basic Concepts

There are a few concepts that are core to Elasticsearch. Understanding these concepts from the outset will tremendously help ease the learning process.

2.1.1 Near Real Time (NRT)

Elasticsearch is a near real time search platform. This means there is a slight latency (normally one second) from the time you index a document until the time it becomes searchable.

2.1.2 Cluster

A cluster is a collection of one or more nodes (servers) that holds all your data and provides federated indexing and search capabilities across all nodes. A cluster is identified by a unique name, which, on DCGS, is set to the site the cluster resides on concatenated with **_Cluster**. For example, the cluster that resides at ECH is given the cluster name **ECH_Cluster**. This name is important because a node can only be part of a cluster if the node is set up to join the cluster by its name.

2.1.3 Node

A node is a single server that is part of your cluster, stores your data, and participates in the cluster's indexing and search capabilities. Just like a cluster, a node is identified by a name, which, by default, is a random Universally Unique IDentifier (UUID) that is assigned to the node at startup. On DCGS the nodes are given the name of the host they run on. For example, u00su01el01 is node 1 and u00su01el09 is node 9. This name is important for administration purposes where you want to identify which servers in your network correspond to which nodes in your Elasticsearch cluster.

In a single cluster, you can have as many nodes as you want. This allows the scaling of the size of the cluster on DCGS as the amount of data to ingest grows. Note that nodes can be added or removed from a cluster.

2.1.4 Index

An index is a collection of documents that have somewhat similar characteristics. For example, you can have an index for customer data, another index for a product catalog, and yet another index for order data. An index is identified by a name (that must be all lowercase) and this name is used to refer to the index when performing indexing, search, update, and delete operations against the documents in it.

Example indexes on OA DCGS are:

- metricbeat-<version>-<date> - Contains all metric data from hosts
- winlogbeat-<version>-<date> - Contains all windows events from hosts

2.1.5 Document

A document is a basic unit of information that can be indexed. For example, you can have a document for a single customer, another document for a single product, and yet another for a single order. This document is expressed in JavaScript Object Notation (JSON), which is a ubiquitous internet data interchange format. Each document correlates a set of *keys* (names of fields or properties) with their corresponding values (strings, numbers, Booleans, dates, arrays of values, geolocations, or other types of data).

2.1.6 Shards & Replicas

An index can potentially store a large amount of data that can exceed the hardware limits of a single node. For example, a single index of a billion documents taking up 1TB of disk space may not fit on the disk of a single node or may be too slow to serve search requests from a single node alone.

To solve this problem, Elasticsearch provides the ability to subdivide your index into multiple pieces called shards. When you create an index, you can simply define the number of shards that you want. Each shard is a fully functional and independent "index" that can be hosted on any node in the cluster.

Sharding is important for two primary reasons:

1. It allows you to horizontally split/scale your content volume.
2. It allows you to distribute and parallelize operations across shards (potentially on multiple nodes), thus increasing performance/throughput.

The mechanics of how a shard is distributed and how its documents are aggregated back into search requests are completely managed by Elasticsearch and is transparent to you as the user.

In a network/cloud environment where failures can be expected anytime, it is very useful and highly recommended to have a failover mechanism in case a shard/node somehow goes offline or disappears. To this end, Elasticsearch allows you to make one or more copies of your index's shards into what are called replica shards, or replicas for short.

Replication is important for two primary reasons:

1. It provides high availability in case a shard/node fails. It is important to note that a replica shard is never allocated on the same node as the original/primary shard that it was copied from.
2. It allows you to scale out your search volume/throughput since searches can be executed on all replicas in parallel.

To summarize, each index can be split into multiple shards. An index can also be replicated zero (meaning no replicas) or more times. Once replicated, each index will have primary shards (the original shards that were replicated from) and replica shards (the copies of the primary shards).

The number of shards and replicas can be defined per index at the time the index is created. After the index is created, you may also change the number of replicas dynamically anytime. You can change the number of shards for an existing index using the shrink and split APIs; however this is not a trivial task and pre-planning for the correct number of shards is the optimal approach.

By default, each index in Elasticsearch is allocated 1 primary shard and 1 replica. Note that the number of replicas applies to all primary shards so if you have an index that has 3 primary shards, and the number of replicas is set to 1 you will have 3 primary shards each having a replica for a total of 6 shards for the index.

2.2 Elastic Terminology and Setup on OA

- Elastic components are referenced by DNS Aliases, which are set as follows:
    - logstash (or logstash.{site})
    - elastic-node-1
    - elastic-node-2
    - elastic-node-3
    - elastic-node-x
    - kibana (Points to load balancer for Kibana nodes at ECH)
    - kibana-wch (Points to load balancer for failover Kibana nodes at WCH)

  NOTE: Take note of these aliases as you will need them throughout this document.

- Elastic nodes that make up the cluster at the ECH are running on Linux VMs and have the naming convention s00su01elXX, where XX is the node number. WCH nodes are named s0asu01elXX. (High side nodes use t00/t0a.)
- There is a Logstash instance running on a VM at each site that is responsible for sending data for that site to the Elastic Cluster. The naming convention for these Logstash instances is sXXsu01lsYY (or tXXsu01lsYY), where XX is the site number and YY is the Logstash instance number. Note that, to date, there is only one Logstash instance per site, so YY is always 01.
- All Windows events are forwarded from the centralized windows event collector (WEC) at each site to the Logstash instance at the site.
- Syslog events are forwarded via rsyslog from each Linux host to the Logstash instance at the site. Events are sent to the standard syslog port (514) on Logstash. Data received on that port is then forwarded to port 5041 to be processed by the Logstash esp_linux_syslog pipeline before being sent to Elasticsearch.
- There is a Python application called the elasticDataCollector, which runs as a service on every Logstash instance.

2.2.1 Data Ingest

On the OA System, data is ingested into Elastic through the Logstash instances. This is to simplify the communication structure and have all traffic from a site to the hub be through a single path. Beats run on each system reporting data, sending data to the Logstash at the site which processes it in a Logstash Pipeline before sending it to Elastic, which might provide further processing in an Ingest Pipeline.

2.2.1.1 Logstash Pipelines

The Logstash Pipelines run by each instance are centrally-managed within Elastic. The install of Logstash defines a list of Logstash Pipelines for that instance to run. The Logstash process on that VM pulls the pipeline definitions from Elastic; when a change is made to the pipeline content, it is

automatically re-pulled by all instances. A description of the Logstash Pipelines is provided in Table *7* Logstash Pipelines.

Table 7 Logstash Pipelines

Pipeline	Source	Description
esp_ashes	**ASHES**	**Pipeline for the SIGINT ASH (AIMS) application.**
esp_eracent_database	**Erecent**	**Pipeline for database queries to Erecent.**
esp_filebeat	**Filebeat**	**Filebeat pipeline for Elastic, Kibana, and applications that can use multple worker threads (messages are independent of each other).**
esp_filebeat-logstash	**Filebeat**	**Filebeat pipeline for Logstash hosts.**
esp_filebeat-singleworker	**Filebeat**	**Filebeat pipeline for applications that require messages to be process be one worker thread (i.e. relation between messages is important).**
esp_hbss_dlp	**HBSS**	**HBSS Data Loss Prevention(DLP) data.**
esp_hbss_dlp-via-connector	**HBSS**	**Pipeline to receive data from EPO ArcSight connector (DLP Workaround).**
esp_hbss_epo	**HBSS**	**HBSS syslog pipeline.**
esp_hbss_metrics	**HBSS**	**Pipeline to ingest data from a custom HBSS powershell script.**
esp_heartbeat	**Heartbeat**	**Heartbeat pipeline.**
esp_idm_database	**IDM**	**Pipeline for connection to the IDM server (OneIdentity).**
esp_linux_syslog	**rsyslog**	**Linux syslog pipeline.**
esp_loginsightLog	**Insight**	**Loginsight pipeline.**
esp_metricbeat	**Metricbeat**	**Metricbeat pipeline.**
esp_postgres	**Postgres**	**Pipeline for connection to the postgresql database.**
esp_puppet_database	**Puppet**	**Pipeline for connection to the puppet postgresql database.**
esp_sccm_database	**SCCM**	**sccm pipeline.**
esp_serena_database	**Serena**	**Serena database pipeline.**
esp_sqlServer_stats	**SQL**	**SQL Server pipeline.**
esp_syslog_tcp	**SYSLOG**	**Linux syslog_tcp pipeline.**
esp_syslog_udp	**SYSLOG**	**Receive all udp syslog data.**
esp_unicorn_database	**UNICORN**	**UNICORN database pipeline.**
esp_winlogbeat	**Winlogbeat**	**Winlogbeat pipeline.**

2.2.1.2 Ingest Pipelines

Elastic will also process some data in Ingest Pipelines. These are defined by Elastic and the processing is based on data provided by the collecting beat. A few Ingest Pipelines are modified for the OA System, the changes are listed in Table *8* Ingest Pipeline Changes.

Table 8 Ingest Pipeline Changes

Ingest Pipeline	Changes Description
filebeat-x.x.x-auditd-log-pipeline	Expand field splitting to multiple spaces, remove geoip database

filebeat-x.x.x-iptables-log-pipeline	Rename message field to event.original, remove geoip database
winlogbeat-x.x.x-powershell_operational	Add a fingerprint (MD5) for each powershell file

2.2.2 Collecting Monitoring Data with Metricbeat

Elastic recommends the use of Metricbeat for the collection and shipping of monitoring data. The best practice is to set up a separate cluster specifically for monitoring, but in the current version on DCGS, Metricbeat will ship all monitoring data to the Enterprise cluster. To ensure Metricbeat has access to Elastic, the existence of the "metricbeat-user" account will be verified before upgrading the elastic nodes. If the "metricbeat-user" account must be created, the password for this user is randomly generated and stored in a metricbeat.keystore file on the host that the setup script is executed on. If the metricbeat.keystore is generated, it **MUST** be copied to the elastic/install directory on the repo server to be distributed to each elastic node during the upgrade process. The installer will receive instructions to do this during the installation process. See *ES-018 - Active Directory - DCMedia DFS for Elastic Metricbeat Installation Instructions.docx* for details.

2.2.3 Monitoring of Domain Controllers

The Enterprise Elasticsearch installation method for Metricbeat (or any beat) on Windows boxes is via Microsoft System Center Configuration Manager (SCCM). This new deployment method does not work for domain controllers in DCGS because they are not under SCCM control. As a result, a new deployment method for domain controllers was developed that modifies the existing Group Policy Object (GPO) for domain controllers. This modification will allow the installation of Metricbeat on the domain controllers using the install script located on a new DFS directory called **DCMedia\Elastic**.

2.2.4 Health and Status Monitoring

The elasticDataCollector exists on each Logstash instance. This Python application performs custom data collection and analysis for infrastructure devices and data analysis of enhanced Metricbeat data to create custom health and information documents. These documents are ingested into Elasticsearch for analysis/visualizations, which allows easier identification of issues on the DCGS system. An elasticDataCollector is installed on each site Logstash instance to create health and information documents for that site. All collected data is written to json files on the Logstash instance and then collected by the Filebeat that is running on the Logstash VM. The Filebeat forwards that data to Logstash, which then sends it to the Elastic cluster to be available for analysis/visualization.

The elasticDataCollector is made up of specialized objects that run in individual threads, each designed to perform specific functions. The objects can be broken down into Infrastructure device data collection and Host/Application monitoring. The following diagram shows the threads (colored in green) that make up the elasticDataCollector.

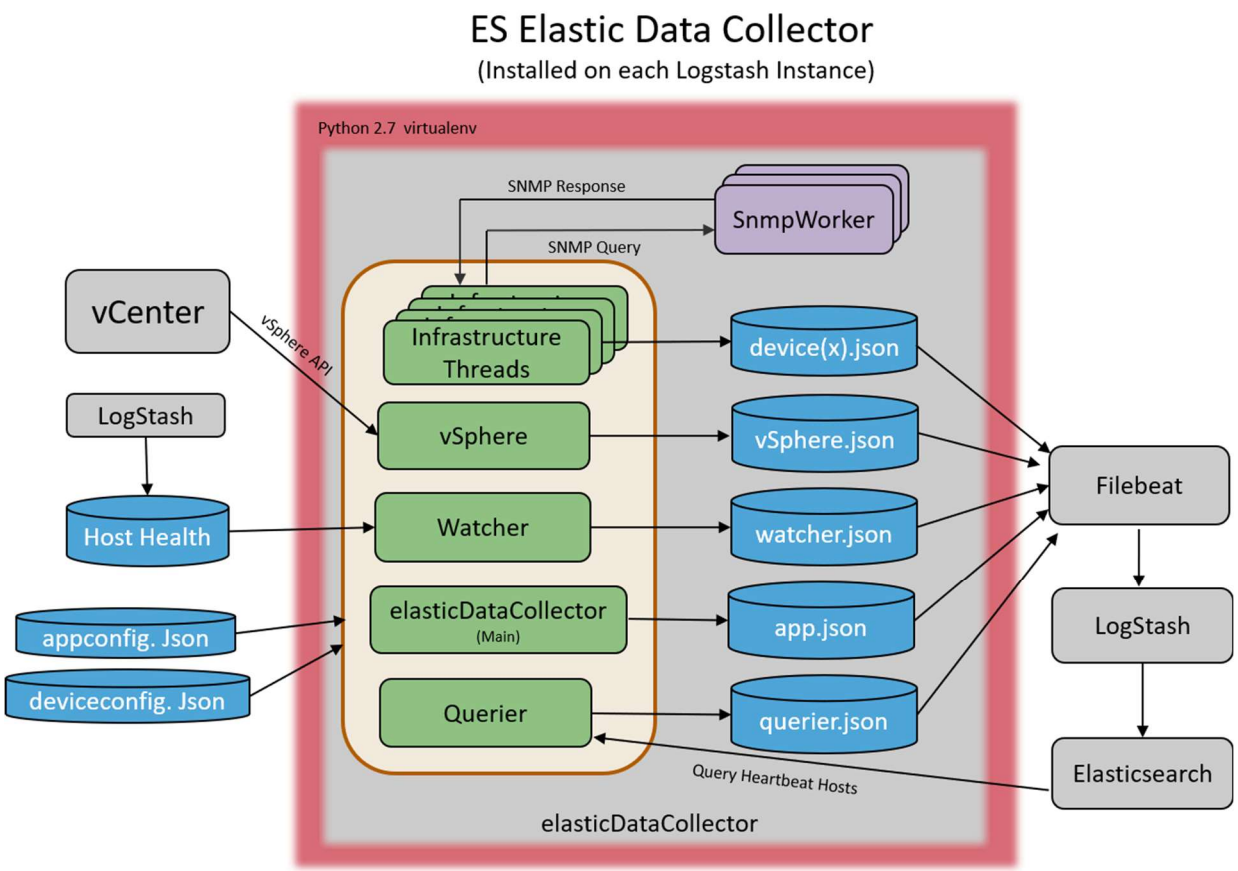


Figure 10 ElasticDataCollector Overview

A worker function "SNMPWorker" is called for every SNMP request. The function executes in its own process space and returns the queried results back to the data collector before exiting. The worker process is named **SNMPWorker-<device>** where **device** is the endpoint where the SNMP requests are being made. The following are a few examples of process names that might be seen while the data collector is executing:

- SNMPWORKER-ech-fx2
- SNMPWorker-ech-nexus5k
- SNMPWorker-u00sv01ex04

These processes will come and go as SNMP requests are being made.

To ensure the correct version of Python is present for the application, a virtual environment containing Python 2.7 is set up on the Logstash VM. The elasticDataCollector is executed from this environment. To help understand the mechanics of the elasticDataCollector, we will break it down into the three functional pieces: Infrastructure monitoring, Host/Application monitoring, and Event Collection.

2.2.4.1 Infrastructure Monitoring

The following diagram highlights in orange the objects of the elasticDataCollector that are used for infrastructure monitoring. A lighter shade of orange signifies that the object is used for more than just infrastructure monitoring.

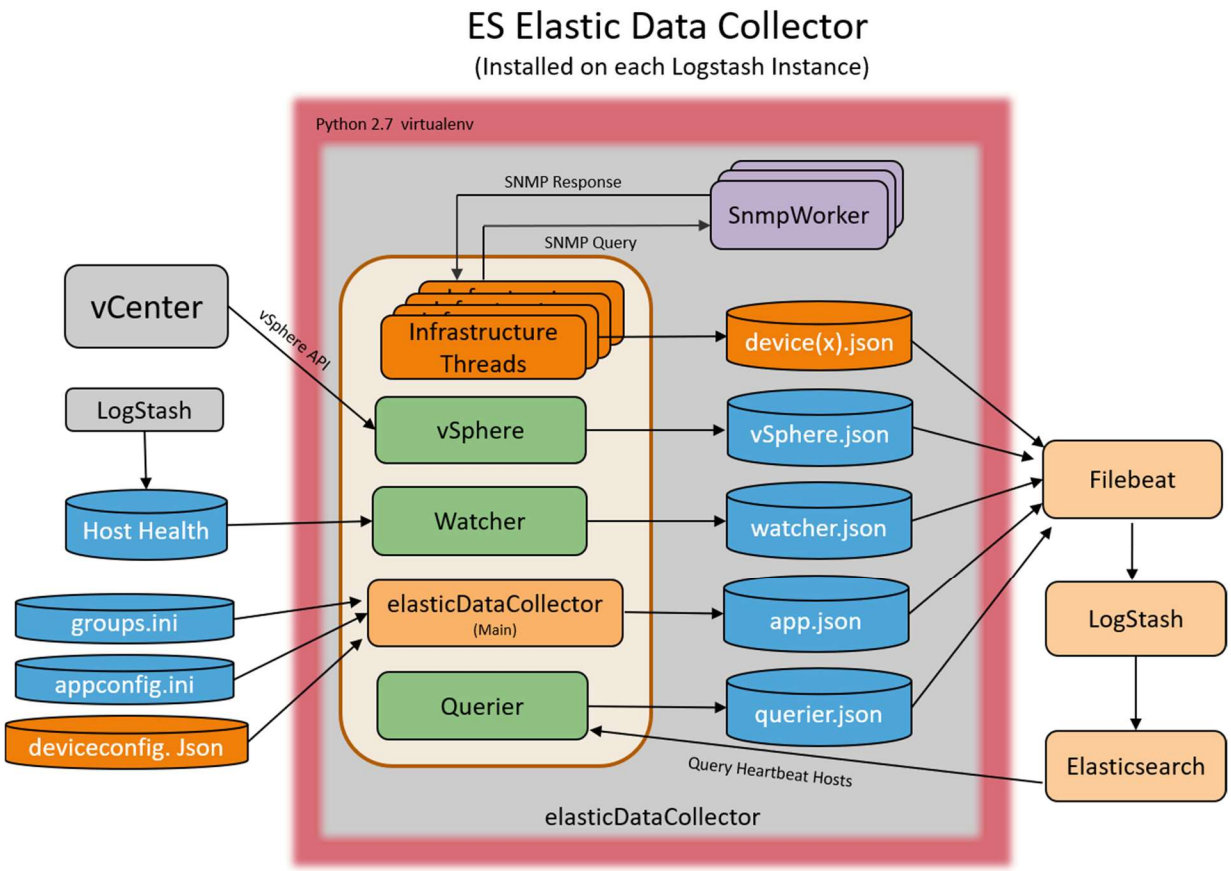


Figure 11 Infrastructure Monitoring Objects

The Infrastructure threads are made up of specific device objects that are used to collect data from each defined device. During the installation process the installer will be asked to use a GUI to create or update a configuration file, which defines the devices to collect data from at the specific Logstash site. An infrastructure thread will be instantiated to collect data from each device. In this version of the elasticDataCollector, the following devices are supported:

- Cisco Nexus5k switch
- Cisco Nexus7k switch
- Cisco 3850 switch
- XtremIO storage device
- Isilon storage device
- Data Domain storage device
- FX2 Chassis

- FC630/640 Blade Servers
- R630/640 Servers

Object oriented design has been used in the elasticDataCollector to allow the easy addition of more devices going forward. The following diagram shows an overview of the architecture for infrastructure devices. Each class derives from the common Device class, which provides common methods. Documents that are created from each class are written to the <device>.json file for collection by the Filebeat running on the Logstash VM.

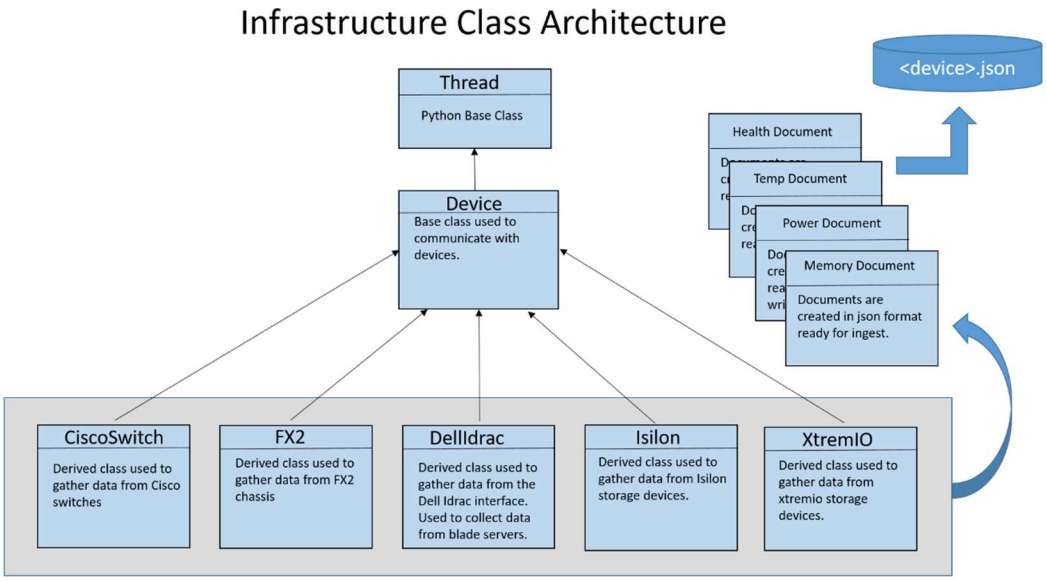


Figure 12 Device Class Architecture

The following diagram shows an example of Infrastructure threads that may be created based on a defined configuration.

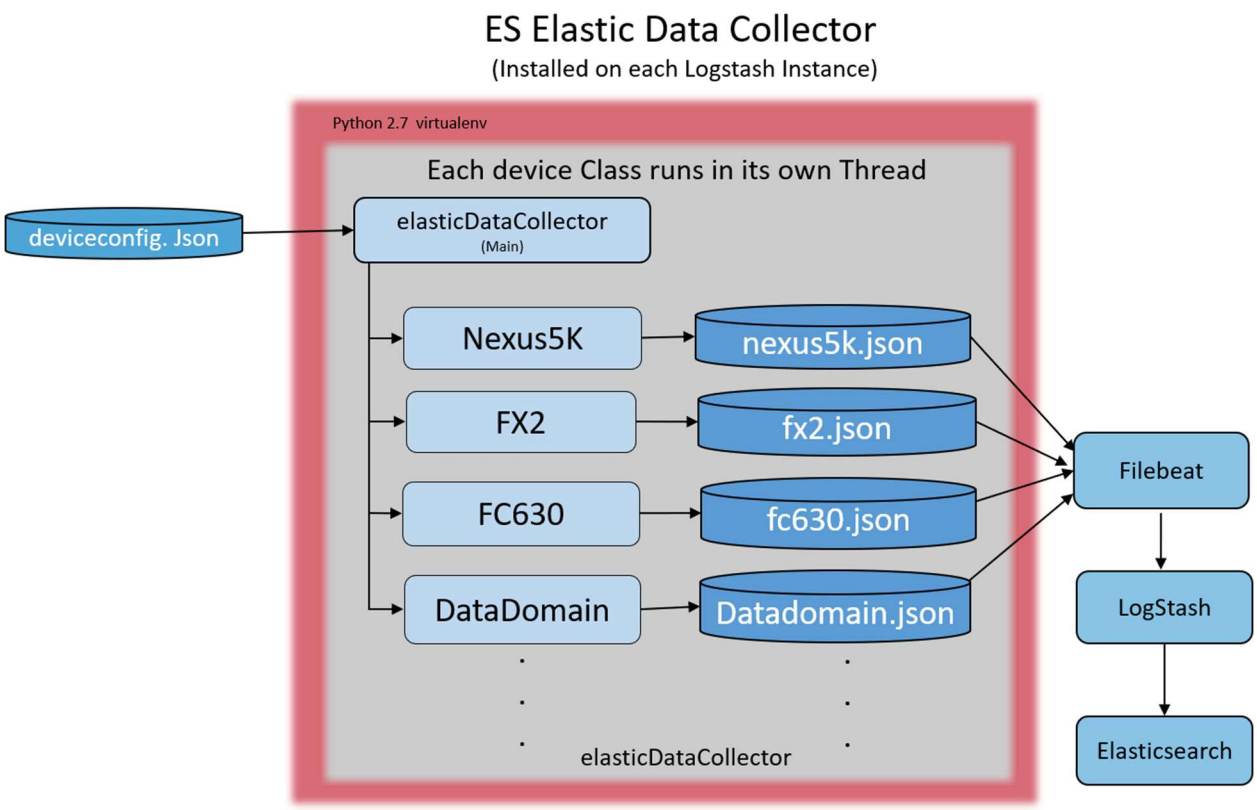


Figure 13 Example of running infrastructure threads

2.2.3.1.1 Infrastructure Switch Symptoms

Table of switch symptoms

Symptom	Symptom Setting	Set Symptom
Fan	Fan "status"	Down
CPU	5 minute CPU average.	> 80%
Memory	CPU memory usage	> 80%
Inlet Major Temp	Second level of incoming air temp threshold warning.	5k > 110 C 7k > 60 C Catalyst > 125 C
Inlet Minor Temp	First level of incoming air temp threshold warning.	5k > 100 C 7k > 42 C

		Catalyst > 105 C
CPU Major Temp	Second level of CPU temp threshold.	5k > 125 C 7k > 85 C Catalyst > 56 C
CPU Minor Temp	First level of CPU temp threshold	5k > 110 C 7k > 75 C Catalyst > 46 C
Interface Input Utilization	Calculates interface input utilization using this queries values against last queries values	> 95%
Interface Output Utilization	Calculates interface output utilization using this queries values against last queries values	> 95%
Uplink Opstate	Check port status on uplink ports to ensure if they are administratively up then they are also operational	Down

2.2.4.2 Host/Application Monitoring

The following diagram highlights in orange the objects of the elasticDataCollector that are used for Host/Application Monitoring. A lighter shade of orange signifies that the object is used for more than just Host/Application monitoring.

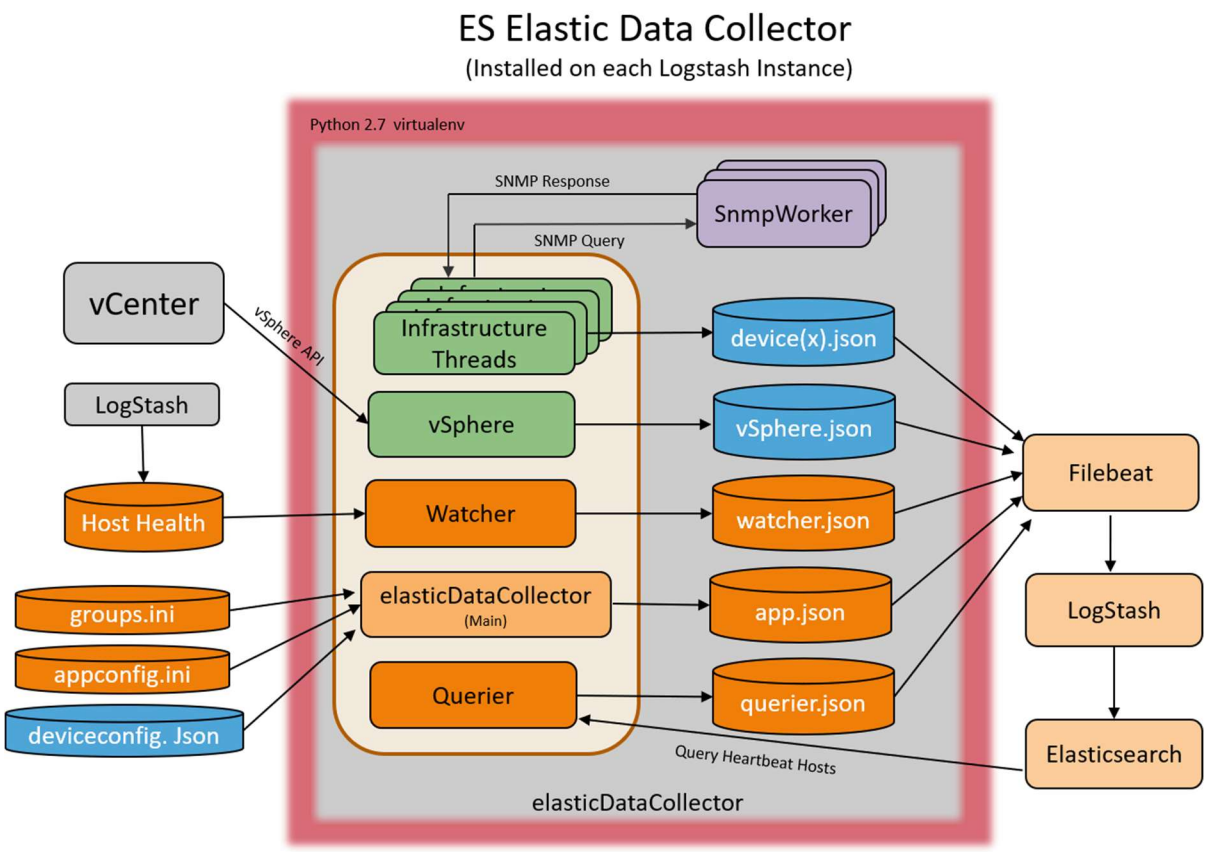


Figure 14 Host/Application Health Objects

Host/Application monitoring can be broken down even more, but we must first examine the source of the data being analyzed. Data flows into the elasticDataCollector from two sources:

- **Host Health:** A data file created and constantly updated by Logstash containing host information for all hosts and application information for any applications that have been configured for monitoring.
- **Elasticsearch:** Heartbeat ICMP information is queried from Elasticsearch to help evaluate the status of hosts.

The originating source of the Host Health data written by Logstash is Metricbeat running on each host. A custom JavaScript processor called appmonitor has been created and is used to examine data that is captured by Metricbeat. Because of minor differences between Windows and Linux metricsets, there is an appmonitor_linux.js and an appmonitor_win.js JavaScript processor. The appmonitor processor is used to combine max CPU, max memory, max file system and networking metrics into a single document. The CPU, memory, and file system values are compared against thresholds to determine if the host is **Ok** or **Degraded**. In addition to combining metrics, the appmonitor processor also has the capability to monitor applications that are running on the host. Applications can be monitored by defining the processes and/or services that the application is made of. The declaration of the appmonitor JavaScript processor along with the thresholds and applications to monitor are part of the metricbeat.yml file. Following is an example of the default metricbeat.yml file for windows:

```
################### Metricbeat Configuration For Windows ######################
# DCGS metricbeat configuration for generic windows boxes
#
# You can find the full configuration reference here:
# https://www.elastic.co/guide/en/beats/metricbeat/index.html

#========================  Modules configuration ============================

metricbeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml

  # Set to true to enable config reloading
  reload.enabled: false

  # Period on which files under path should be checked for changes
  #reload.period: 10s

logging.level: error

# ================================= Processors ================================
# Configure processors to enhance or manipulate events generated by the beat.
#
# The appmonitor processor will pull key metrics from different metricsets and
# evalualte the health of applications by looking at the services/processes that
# are needed for the application to be healty

processors:
  - script:
      lang: javascript
      id: App_Monitor
      file: ./appmonitor_win.js
      params:
        not_justStarted: false
        metrics:
          {
            'cpu': { 'curval': 0.0, 'threshold': .90 },
            'memory': { 'curval': 0.0, 'threshold': .95 },
            'filesystem':
              { 'curval': 0.0, 'threshold': .90, 'mount_point': 'none' },
          }
        network: { 'curBytesin': 0, 'curBytesout': 0 }
        apps:
          {
            'McAfee_Agent':
              {
                'hosteffect': 'Degraded',
                'services':
                  {
                    'macmnsvc': { 'running': false, 'effect': 'Down' },
                    'masvc': { 'running': false, 'effect': 'Down' },
                  },
              },
            'McAfee_EndPoint':
              {
                'hosteffect': 'Degraded',
                'services':
                  {
                    'mfemms': { 'running': false, 'effect': 'Down' },
                    'mfevtp': { 'running': false, 'effect': 'Down' },
                  },
              },
          }
#------------------------------ Logstash output -------------------------------
~
```

Figure 15 Default metricbeat.yml file for Windows

Monitoring of McAfee_Agent and McAfee_Endpoint are included in the default monitoring configuration for all Windows hosts. The format to define an application for monitoring is as follows:

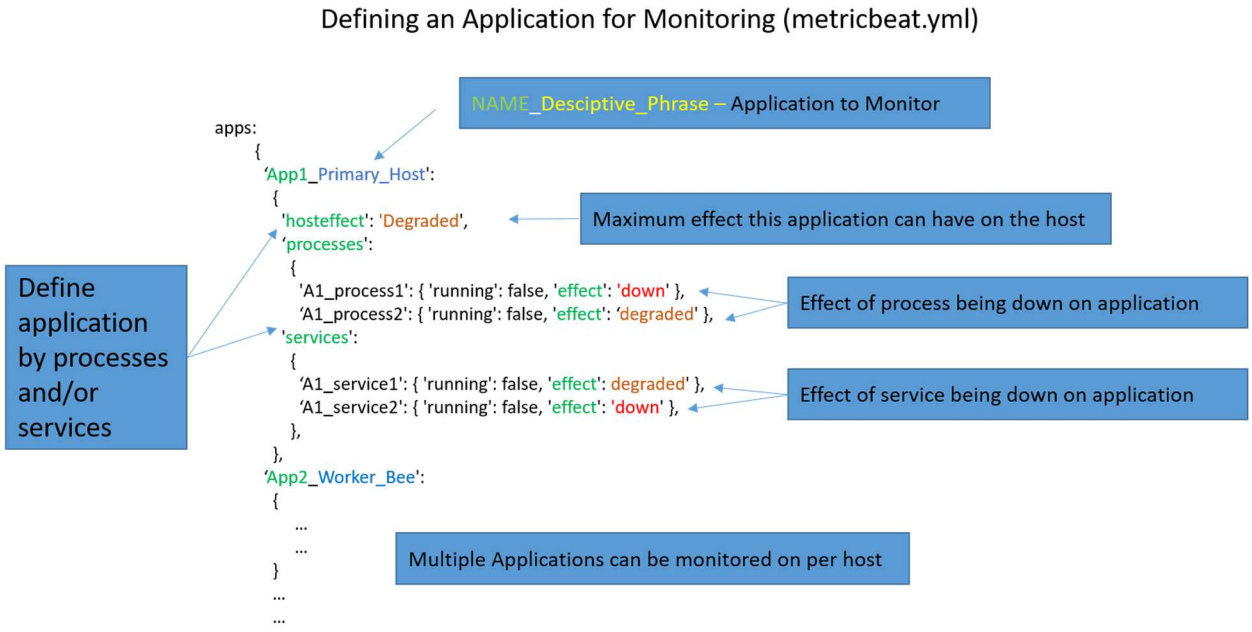


Figure 16 metricbeat.yml application monitoring definition

The definition of an application for monitoring consists of the following:

- **NAME_Descriptive_Phrase:** The name of the application to monitor is all characters up to the first underscore '_'. All characters after the first underscore are used as a descriptive phrase for the application. In the previous example, the application name would be **App1** and the descriptive phrase **Primary_Host**. Looking at Figure 15, the default Metricbeat configuration for Windows shows two McAfee applications, one with descriptive phrase **Agent** and the other **EndPoint**. The purpose of using the same name for applications on a single host or multiple hosts is to look at multiple processes that make up an overall application together.
- **Hosteffect:** This is the definition of the maximum effect this application can have on the overall host health. For example, if this is set to **Degraded**, even if the application is **Down**, the worst the host can be is **Degraded**.
- **Processes/Services:** This is the definition of the processes and/or services that make up an application. One or both sections can be provided. (See the default metricbeat.yml for Windows in Figure 15, which only defines services for the McAfee applications.)
- **Effect:** This setting in both the process/service section defines the effect of the process/service being down on the application itself.

Figure 16 illustrates the ability to monitor any process by name. However, in many cases, there are multiple processes with the same name and the only way to identify what they are is by looking at the arguments that were passed when they were started. A good example of this is java applications; the process name is always java, but the arguments passed determine which java application is running. For

this reason, advanced application monitoring configurations are also available (currently Linux only). This allows the specification of multiple instances of an application (process) to be monitored. The format for advanced process monitoring is shown in the following figure.

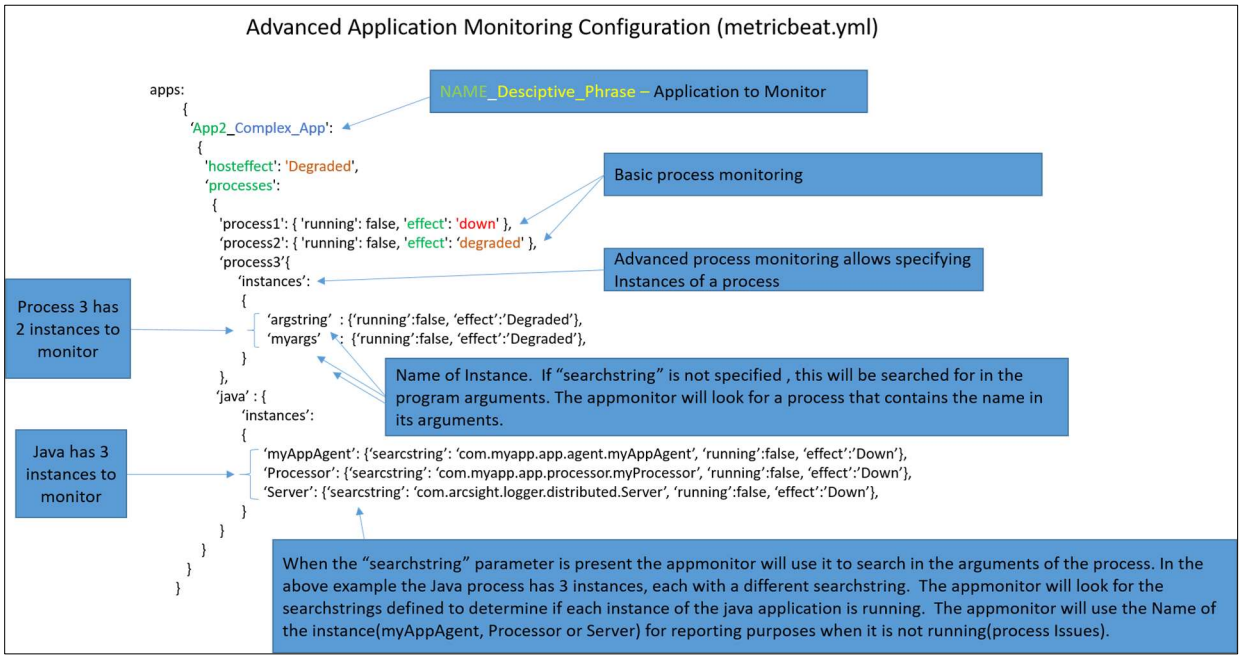


Figure 17 Advance Process Monitoring

The definition of instances of a process consists of the following:

- **The "instances" keyword:** This signals that there will be multiple occurrences of this process to look for.
- **Name of Instance:** This is used to identify the instance of the process. If it is not running, this name will be appended to the process name for identification (reported as **Process issues**). This string will also be used to search the command line arguments to identify the process if a "searchstring" is not present.
- **"searchstring" (optional):** If present, this string will be used in place of the instance name to search for in the command line arguments. This allows naming the instance something different than the string to search for to allow clearer identification of processes in the **Process issues** when reported.

Note that this information is for understanding purposes only, as these configuration files should not be manually modified. The Metricbeat application, JavaScript processor, and metricbeat.yml files are automatically installed on all Windows and Linux hosts on the DCGS System. Specific configurations for certain hosts can be defined but must be added and verified by an Elastic SME. Any new configurations must be added to the baseline for proper automated distribution. If manual changes are made to any metricbeat.yml file they will be automatically overwritten by Puppet or SCCM with the baseline configuration for the host.

Now that you have been told about the data that the JavaScript processor combines, and shown how applications can be defined for monitoring, let's review how the data is packaged and passed to Logstash. In our custom implementation, the **process_summary** metricset is used as a timer for the appmonitor JavaScript processor. Every time a process_summary metric is taken by metricbeat, the appmonitor processor appends our custom host health data to the document created. Figure 18 shows an example of the metrics appended and information about three applications being monitored on a particular host.

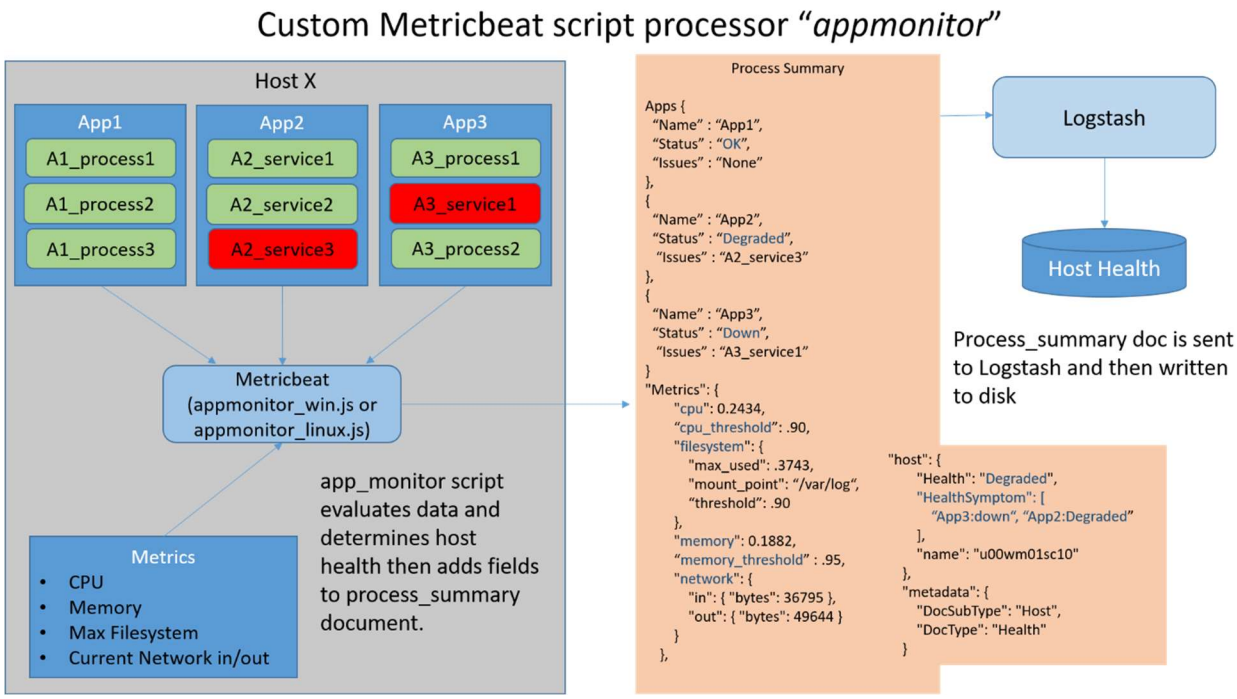


Figure 18 Example of appmonitor data appended to "process_summary" document

The information that is appended to the **process_summary** document is sent by Metricbeat to Logstash where it is then stripped back out of the document. The information is written to the **hosthealth-yyyy-mm-dd.json** (shown as "Host Health" in diagrams) file for the elasticDataCollector to process. Some of the information is also used by Logstash to create documents of subtype app-host, which are sent directly to ElasticSearch.

Now that the source of the data written to the Host Health file has been explained, we can take a closer look at the inner workings of the elasticDataCollector for Host/Application monitoring. The following

figure visualizes how all health documents are created. As you can see, two of the flows (A and B) deal with data received from the Metricbeat appmonitor.

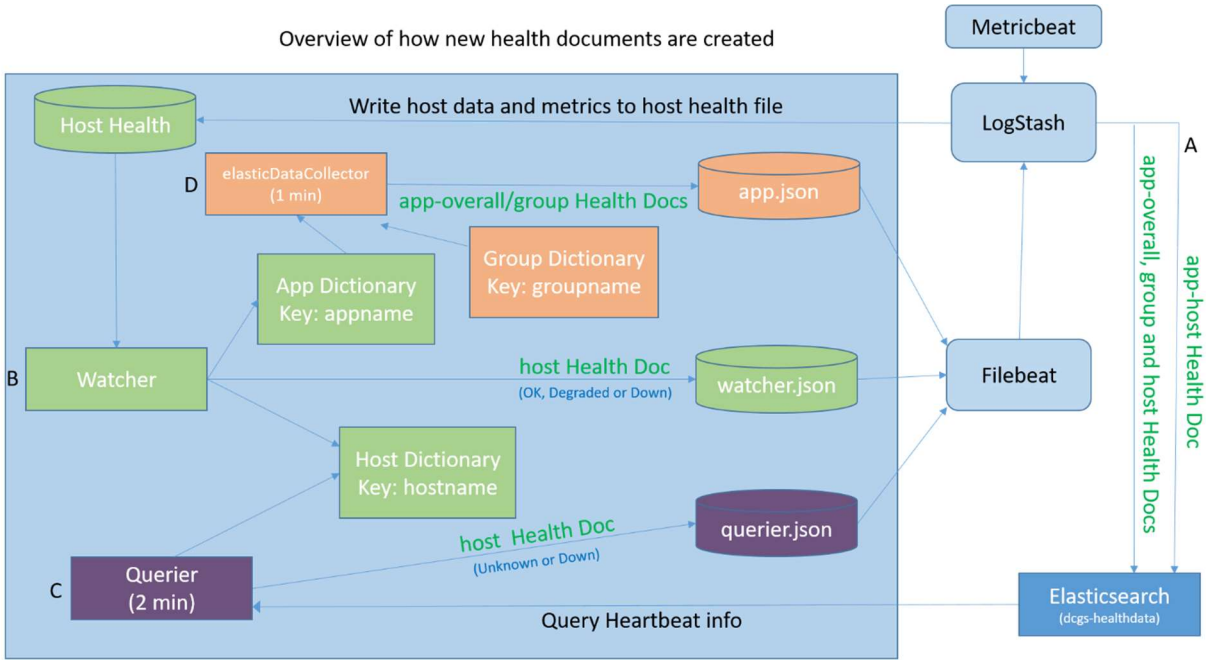


Figure 19 Overview of Health Document Creation

To make things clearer, the previous diagram will now be broken into parts to explain each data flow.

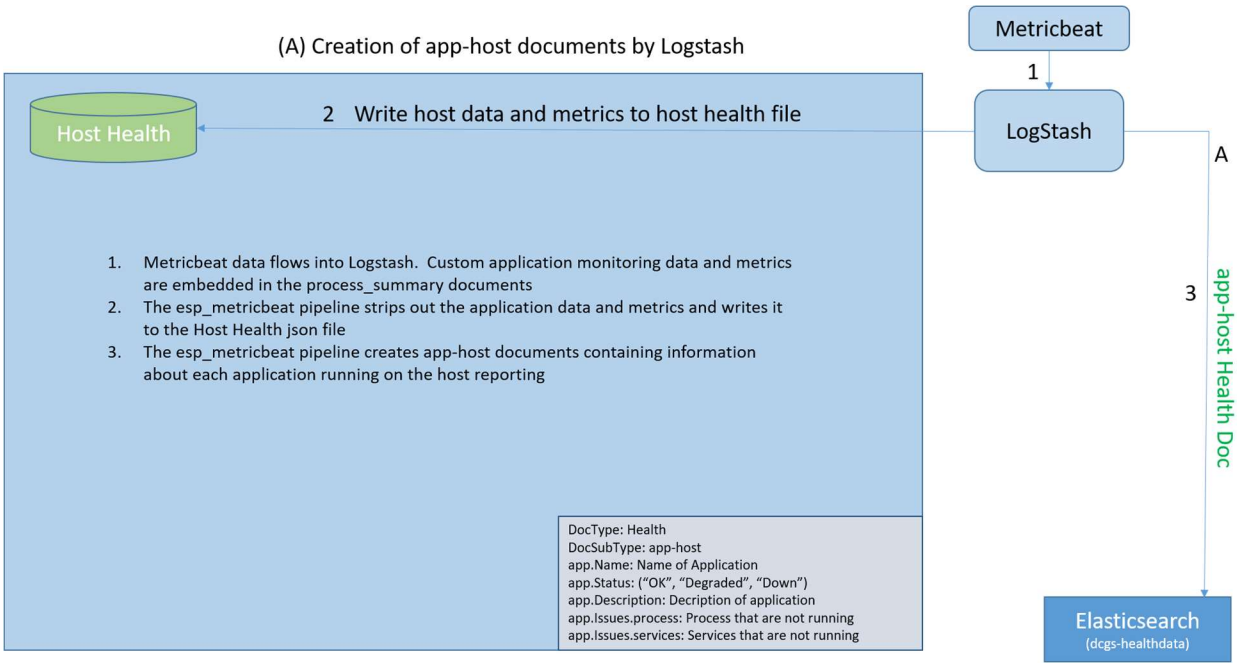


Figure 20 Flow A, Creation of App-Host Documents

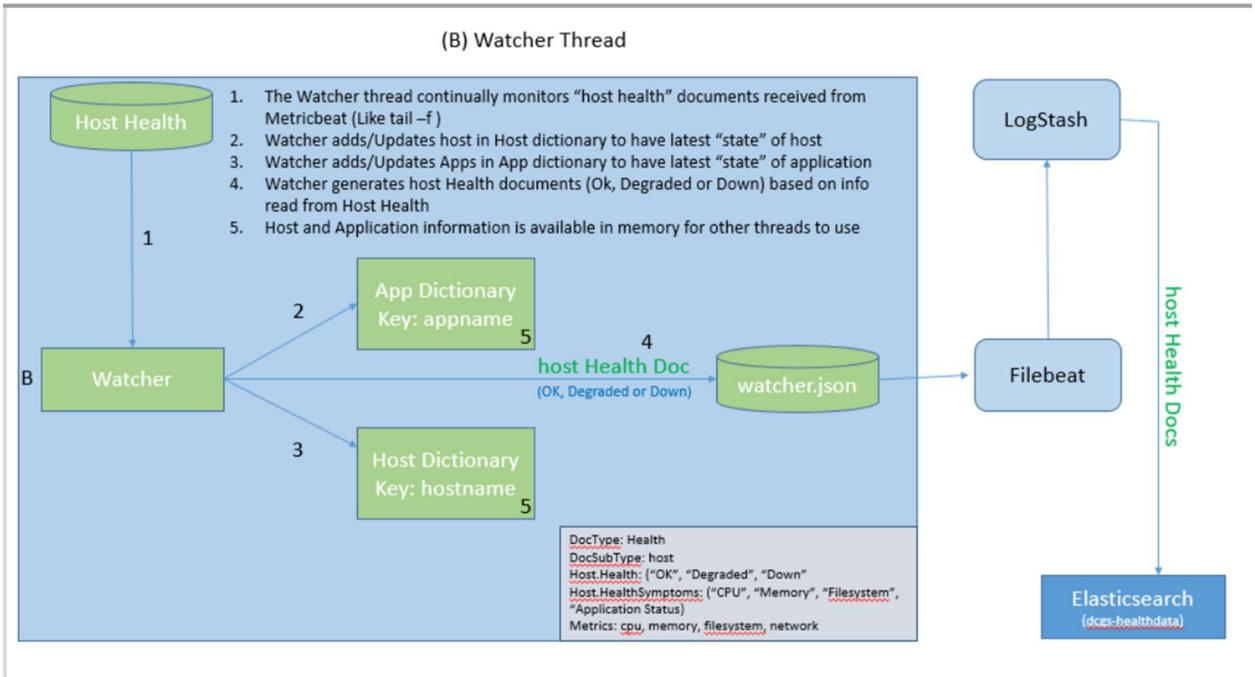


Figure 21 Flow B, Watcher Thread

Figure 22 starts out querying Heartbeat information from Elasticsearch. To be specific, this is ICMP information for all hosts at the site where the instance of Logstash is running.

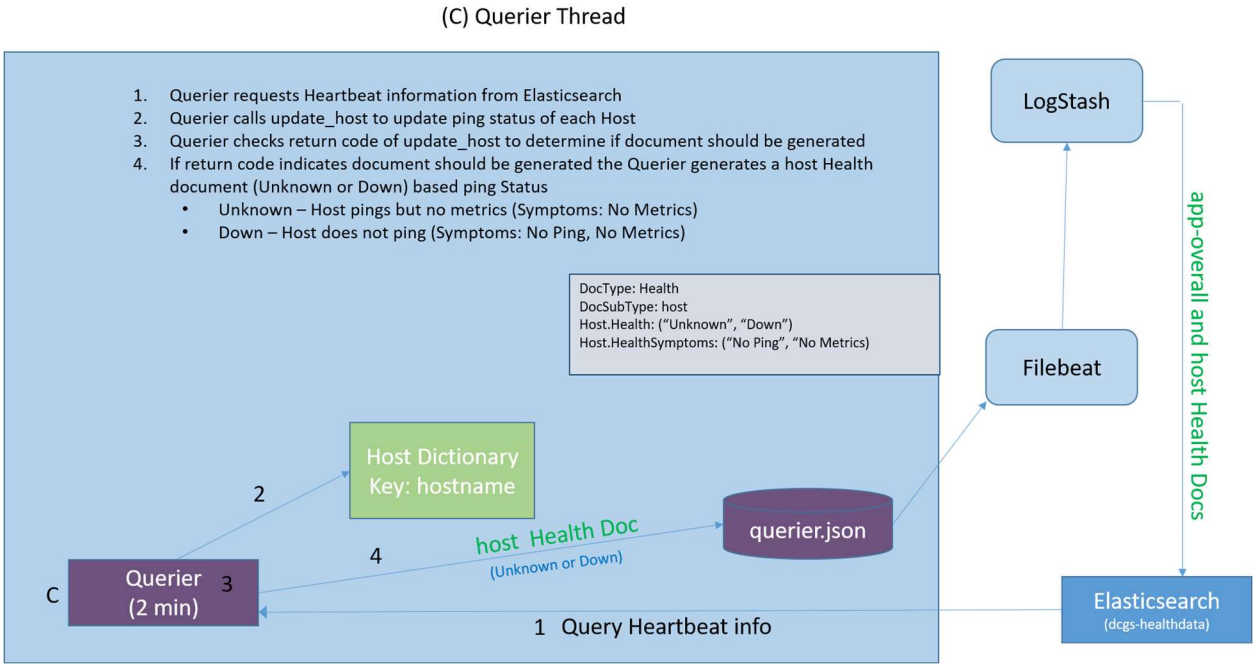


Figure 22 Flow C, Querier Thread

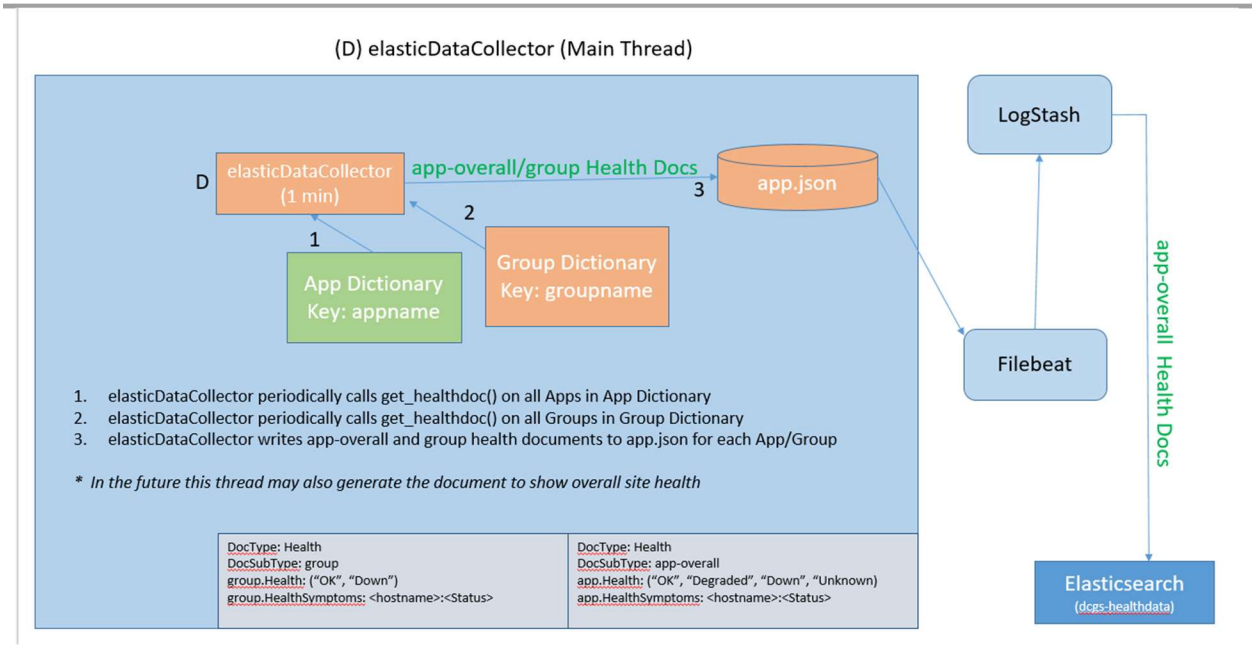


Figure 23 Flow D, elasticDataCollector Main Thread

ICMP pooling is documented in the initial 7.9.1 release documentation; the flow for determining which hosts to ping is shown in Figure 24.

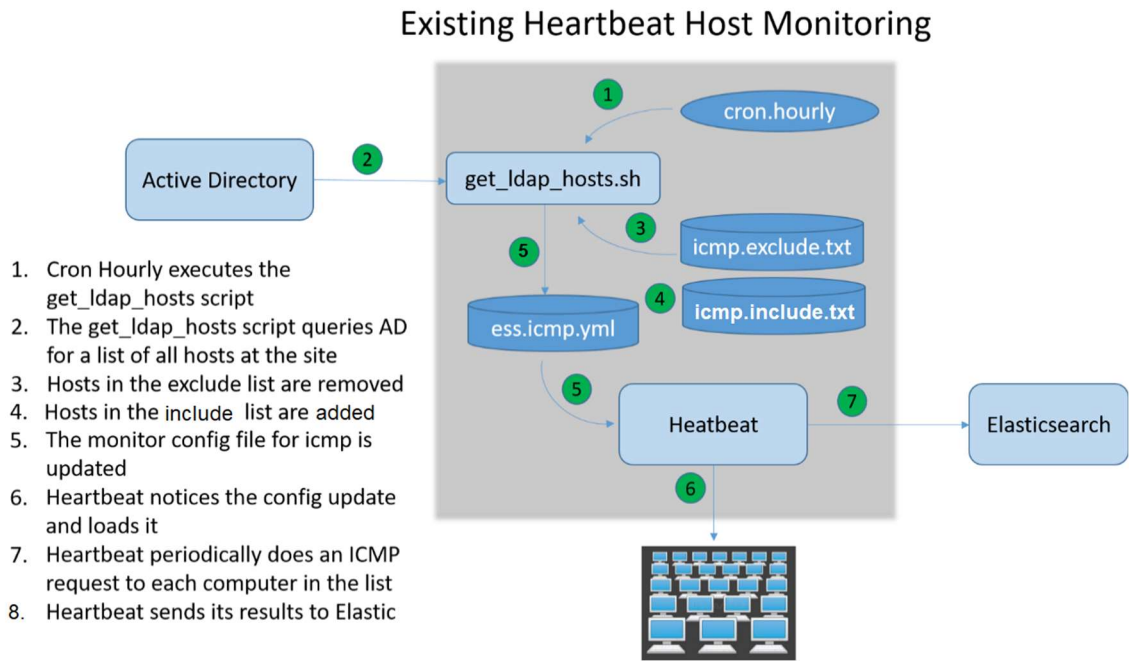


Figure 24 Existing Heartbeat Host Monitoring

Applications and Groups to be monitored for each site are specified in the following configuration files.

- **appsconfig.ini:** Defines applications that need to be monitored. The applications generally span multiple hosts. The header of the appsconfig.ini is shown here and details the syntax of the file. Example configurations for hubs and sites will be provided with the installation. The contents of this configuration file must be updated and verified for the each Logstash instance that is installed.

```
# Configuration file for applications to monitor
#
# Syntax:
# [Application Name]
# hostname=effect on overall application
# hostname=effect on overall application
# hostname=effect on overall application
#
# Note: Application name must be just as it's defined in the
#       metricbeat.yml configurations for the application
#       The syntax on the metcbeat.yml is:
#            name_decription
#       the application name is up to the first '_' char
#
# Example:
# [MyApp]
# host1:Degraded
# host2:Down
#
# - The maximum effect host1 can have on the overall application is
#   causing it to be Degraded.
# - The maximum effect host2 can have on the overall application is
#   causing it to be Down
#
```

Figure 25 appsconfig.ini header

- **groups.ini:** Allows the monitoring of multiple hosts as a group. A minimum number of the hosts must be **OK** for the group to be **OK**. The header of the groups.ini file is shown here and details the syntax of the file. This file must be updated after installation for any groups to monitor at a site.

```
# Configuration file for groups to monitor
#
# Syntax: # [Group Name]
# group_min=minimum number of workstations in group required for group to be OK
# group_hosts= list of hosts in group
#
# Example:
# [MyGroup]
# group_min = 4
# group_hosts = myhost1, myhost2, myhost3, myhost4, myhost5, myhost6
#
# - At least 4 of the 6 listed hosts must be OK for the group to be OK
#
~
~
```

Figure 26 group.ini file header

2.2.4.3 Event Collection

A new class has been added to the elasticDataCollector to collect audit event data from vCenter hosts. This new class is called vSphere as it uses the vSphere API to communicate with the vCenter host at each site. The following diagram highlights in orange the objects of the elasticDataCollector that are used for vCenter Audit event collection. A lighter shade of orange signifies that the object is used for more than just infrastructure monitoring.

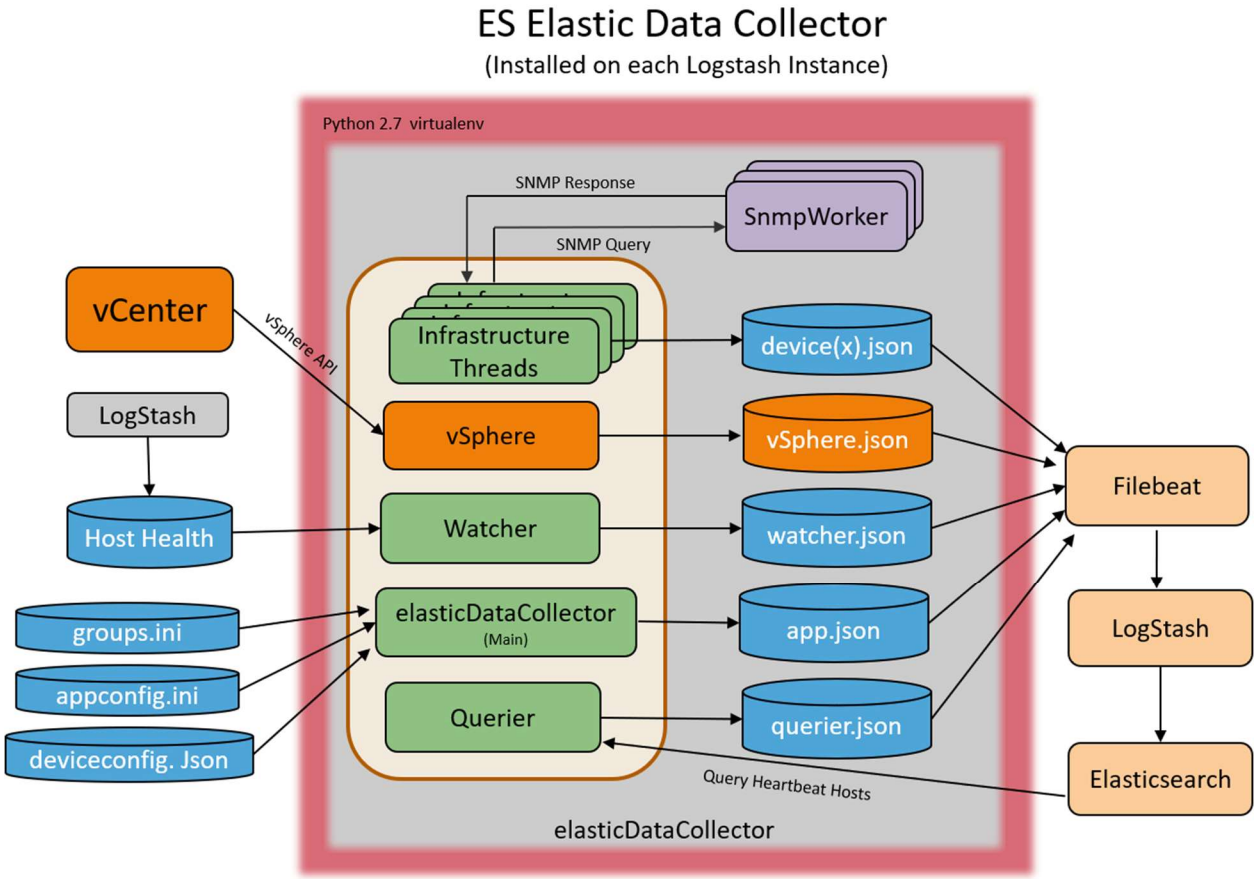


Figure 27 vCenter Audit Event Collection

2.2.5 Metricbeat vSphere Data Collection

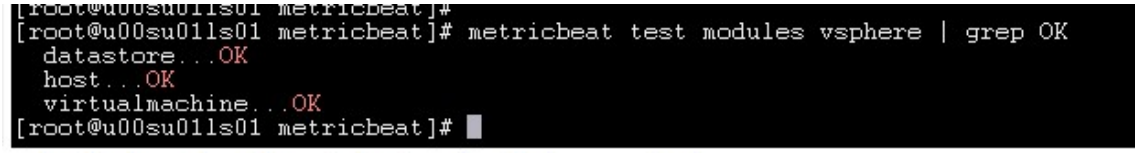
Metricbeat is automatically installed on the Logstash instance at each site and should be configured to collect vSphere statistics for that site. The {xx}_elastic.svc account is used to collect the data using the vsphere module of Metricbeat. The password for the service account is stored in the Metricbeat keystore as "V_PWD". You can't see the password but can verify the secret is present by executing the following:

```
# metricbeat keystore list
```

You can verify the password is correct by testing the module:

```
# Metricbeat test modules vsphere | grep OK
```

The following should display:



2.2.6 Heartbeat

Heartbeat is installed automatically by the **dsil_elastic_servers** Puppet module. Unlike Metricbeat and Filebeat, the configurations are not continually updated by Puppet. On the initial install the baseline monitor files are loaded. The computer names and port numbers in these files must be verified for correctness at each site where Heartbeat is installed. Any additional monitoring should also be added after the initial install. Note that Puppet will not overwrite any changes made to any of the monitors located in the **monitors.d** sub directory.

The initial configuration of Heartbeat will most likely need to be adjusted after its initial installation.

NOTE: The heartbeat.yml file is under puppet control and the location information is updated automatically via the heartbeat pipeline.

2.2.6.1 Heartbeat Monitors

Heartbeat uses 3 different types of monitors to determine if your servers/services are reachable:

- **ICMP:** Uses an ICMP Echo Request to ping the configured hosts.
- **TCP:** Connects via TCP and optionally verifies the endpoint by sending and/or receiving a custom payload.
- **HTTP:** Connects via HTTP/s and optionally verifies that the host returns the expected response. PKI certificate start and end dates are also returned with HTTPS requests.

The following HTTPS and TCP monitor configurations are installed by default. The computer names and ports in these monitors must be verified for correctness after the initial install. Some servers/services are only available at the hub so there are extra monitors configured for that location.

- At ECH:
  - ess.http.hub.yml
  - ess.http.site.yml
  - ess.tcp.hub.yml
- At each Site:
  - ess.http.site.yml
  - ess.tcp.site.yml

The default ICMP monitor only pings the Logstash instance that Heartbeat is installed on. To get a list of hosts to ping for each site's Logstash instance, the **get_ldap_hosts.sh** script is executed hourly by the

cron.hourly script **heartbeat.cron**. The get_ldap_hosts.sh queries Active Directory to get a list of all computers that start with the site's designator. This script can also be executed manually to verify the configurations. Run the following to modify the **/etc/heartbeat/monitors.d/ess.icmp.yml** file, updating the list of computers to ping for the Logstash instance:

```
# /etc/heartbeat/get_ldap_hosts.sh
```

Edit the /etc/heartbeat/monitors.d/ess.icmp.yml file to view and verify the hosts to be pinged. If there are any hosts in the list that should not be pinged, they can be added to the exclusion list.

To add hosts that should not be pinged to the exclusion list, edit the **/etc/heartbeat/icmp_exclude.txt** file and add hostnames (one per line) for them to be excluded. After adding hosts to the file, you can run the **get_ldap_hosts.sh** script manually or wait for it to be run automatically. The hosts will no longer be in the list contained in the ess.icmp.yml file.

If there are any hosts that should be pinged whether or not they are returned by Active Directory, they can be added to the inclusion list. To add hosts that should be pinged to the inclusion list, edit the /etc/heartbeat/icmp_include.txt file and add hostnames (one per line) for them to be included. After adding hosts to the file, you can run the get_ldap_hosts.sh script manually or wait for it to be run automatically. The hosts will be added to the list contained in the ess.icmp.yml file.

Once configuration is complete, verify Heartbeat data is received by selecting the Kibana **Discover** tab and selecting the **Heartbeat-*** indexes. You should see "hits" populating on the selected graph to confirm Heartbeat data is being received.

Figure 28 Verify Heartbeat data is received

2.3 Daily Checks

The health of the Elastic cluster should be checked each day to ensure that data is flowing into the cluster and that users are able to access the data.

2.3.1 Cluster Health

The Elastic cluster is made up of multiple nodes. The health of the cluster can be checked in multiple ways, the easiest of which is to use Kibana. However, there are times when access to Kibana may be

unavailable. Before checking the status, let's discuss what possible status you will see. The 3 possible statuses for Elastic are as follows:

1. **Green:** All primary and replica shards are allocated. Your cluster is 100% operational.
2. **Yellow:** All primary shards are allocated, but at least one replica is missing. No data is missing, so search results will still be complete. However, your high availability is compromised to some degree. If *more* shards disappear, you might lose data. Think of yellow as a warning that should prompt investigation.
3. **Red:** At least one primary shard (and all its replicas) is missing. This means that you are missing data; searches will return partial results and indexing into that shard will return an exception.

NOTE: If cluster health does not show Green, refer to the troubleshooting sections to resolve.

Here are 3 ways to check cluster health:

1. From the **Stack Monitoring** tab under **Management** in Kibana.

   As of the Elastic 8.11.3 release, clusters exist at each hub (on production). Each cluster can be monitored from either hub. Selecting Stack Management first shows a summary Cluster page as shown in Figure 29 Elastic Stack Monitoring *Cluster Listing*. Selecting a cluster shows more details.

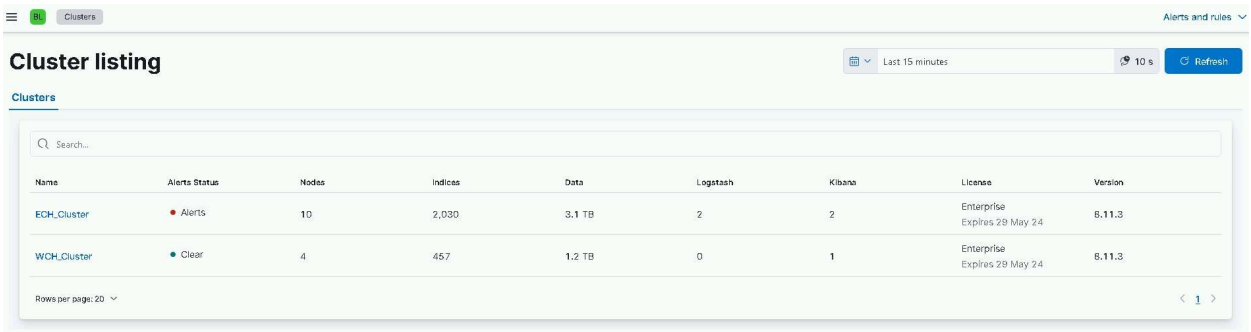


Figure 29 Elastic Stack Monitoring Cluster Listing

Figure 30 shows an example of these details showing a cluster of 10 nodes in **Healthy** status.

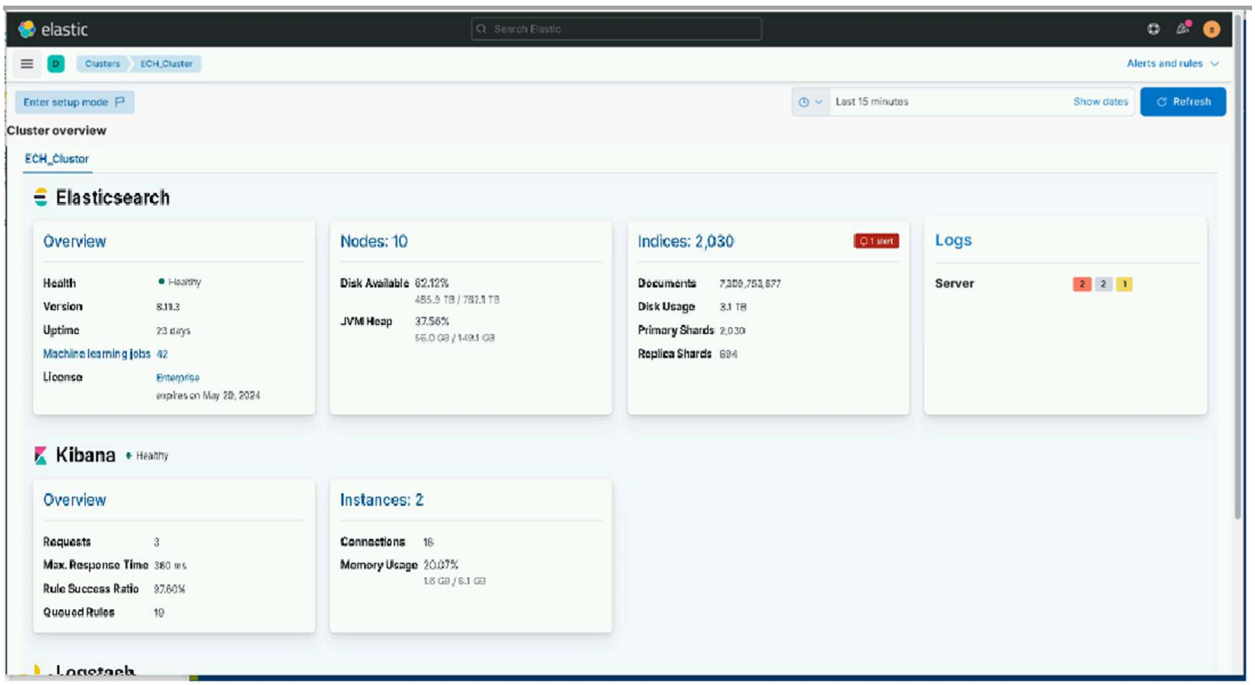


Figure 30 Elastic Stack Monitoring tab showing a Healthy Status

2. From the **Dev Tools** tab of Kibana. Run the following command from the **Dev Tools** tab:

```
GET _cluster/health
```

It returns:

```
{
  "cluster_name" : "ECH-Cluster",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 10,
  "number_of_data_nodes" : 6,
  "active_primary_shards" : 518,
  "active_shards" : 1037,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100.0
}
```

3. From the Linux command line. If Kibana is not working properly, your only option to check the status of Elastic is from the command line.

NOTE: There are several instances where the cluster's health is fine but there is an issue with Kibana.

```
# curl -k -u {username} https://elastic-node-1:9200/_cluster/health?pretty
```

NOTE: If you have trouble getting a response from "elastic-node-1" try another node

2.3.2 Node Status

The cluster may be Green but are all the nodes of the cluster up? Elastic has built-in fault tolerance and will automatically recover if a node goes down in the cluster. So, it is possible to have a cluster status of Green but be missing nodes that should be part of the cluster. The monitoring tab (shown in Figure 30) also has a "**Nodes**" section. This will give you a count of the number of nodes that are currently active in your cluster. Verify the number of nodes is correct. You can get a list of the clusters nodes by selecting the **Nodes** link on the **Cluster Management** page. The results will look similar to the following:

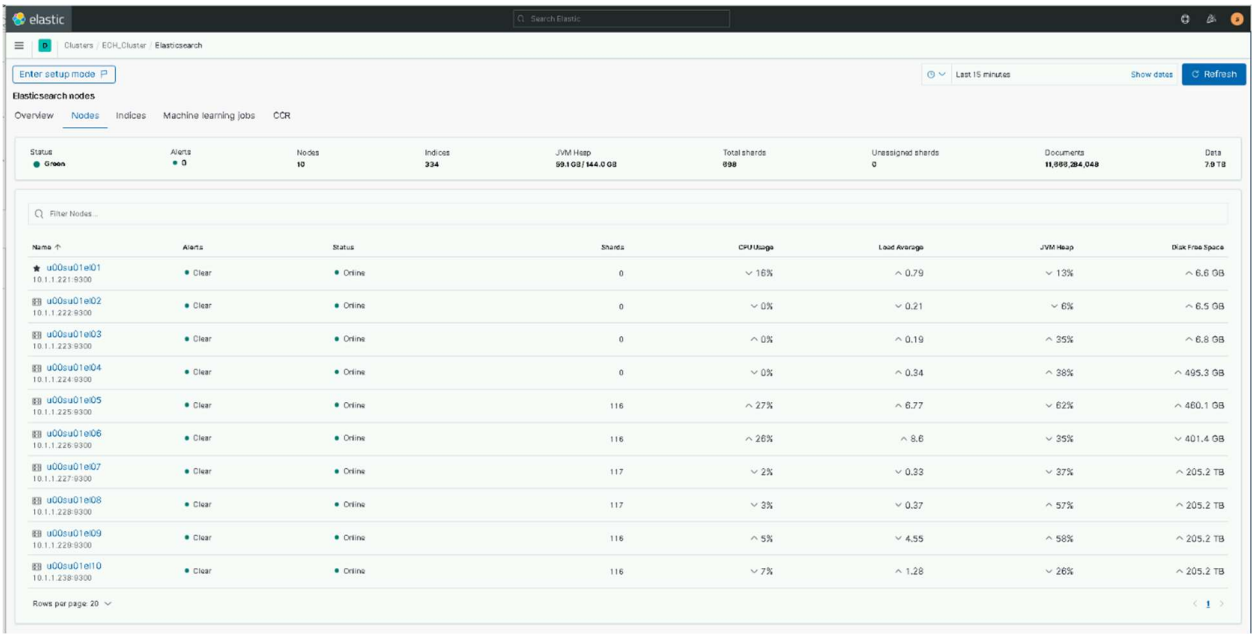


Figure 31 Elastic Node Status

NOTE: If a node is missing, refer to the troubleshooting section to resolve.

2.3.3 Logstash Nodes Running Properly and Data Ingested Properly

Knowing that the Elastic Cluster is running properly is important and should be done first. After making sure the Cluster is healthy, you should also verify that all Logstash instances are up and running and sending data properly to the cluster.

2.3.3.1 Verify All Logstash Instances Talking to Cluster

An easy way to do this is also found on the **Stack Monitoring** tab (shown in Figure 30). Verify that the number of Logstash nodes is what is expected. Selecting the **Nodes** link under Logstash on the Stack Monitoring page will list the Logstash nodes that are currently sending data to Elastic. **Selecting a node,**

then the Pipelines tab will show the ingest rates from each pipeline for that node (or selecting the Pipelines link on the main Stack Monitoring page will show pipeline traffic from all nodes combined).

2.3.3.2 Verify Expected Data Types Being Ingested

Seeing a Logstash node up and running may not be enough information to verify the data we are expecting is being ingested. The best way to do this is to create an **Events Received by Collector Type** dashboard.

Note that this dashboard was available in versions prior to 7.9.1 but was not included in the latest releases. The dashboard is easy to create so if you don't have one to monitor events received it is a good exercise to create one.

1. Create a visual to monitor events by DCGS_site for each data type.
2. Create a dashboard and place all the event monitoring visuals on it.

Following is an example of a simple visual configuration to monitor the amount of Metricbeat data from each site:

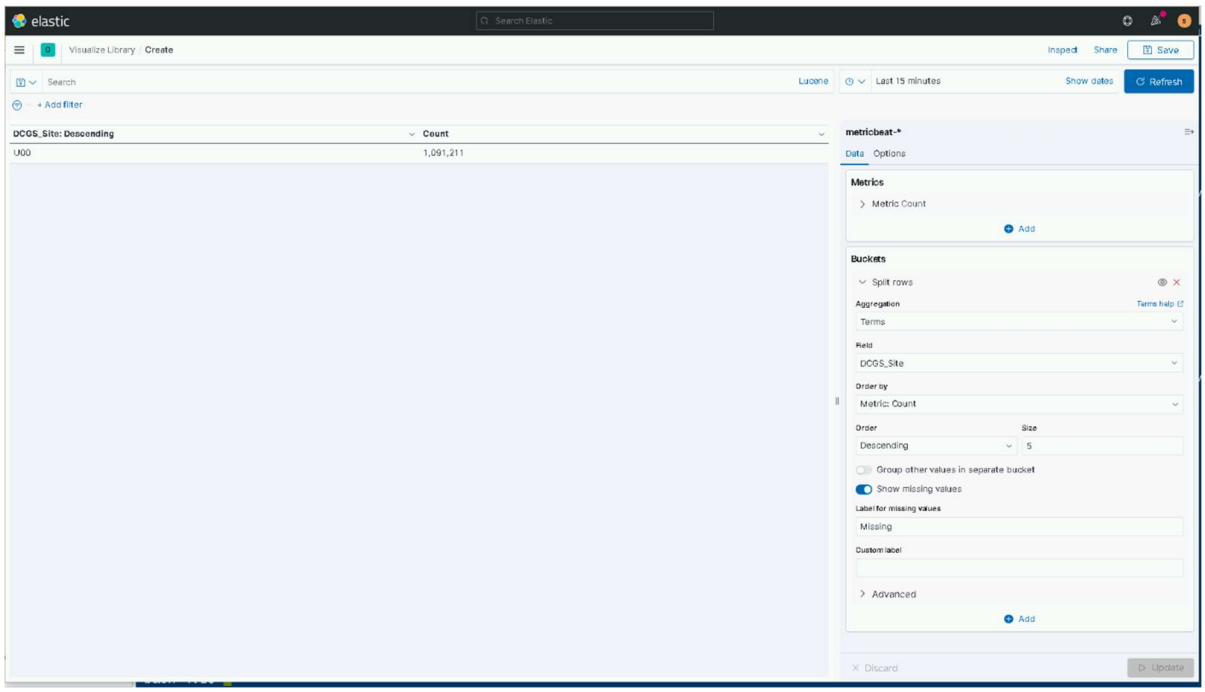


Figure 32 Example visual to monitor Metricbeat events

NOTES:

- Currently all sites are sending metricbeat, winlogbeat and linux-syslog data. Some are also sending other data types. If you don't know what datatypes to look for, ask the Elastic SME at the DSIL for guidance.
- If a Logstash node is missing, or not sending the expected data, see the troubleshooting section to resolve.

2.3.4 Disk Space Issues

Elastic and Logstash may have trouble running if the machine they reside on runs out of disk space for any reason. Log files stored in /var/logs can sometimes cause disk space issues on Linux machines and should be checked regularly. The easiest way to do this is to use the OA DCGS Health & Status dashboard in Kibana. By default, the **disk utilization** visual on this dashboard shows the 10 machines with the highest disk usage across the entire enterprise.

You can enter a filter for the dashboard (in the Search bar) to only Elastic and Logstash machines. Following is an example for an unclassified environment, u00. You would need to change the "u" to the correct prefix for your environment.

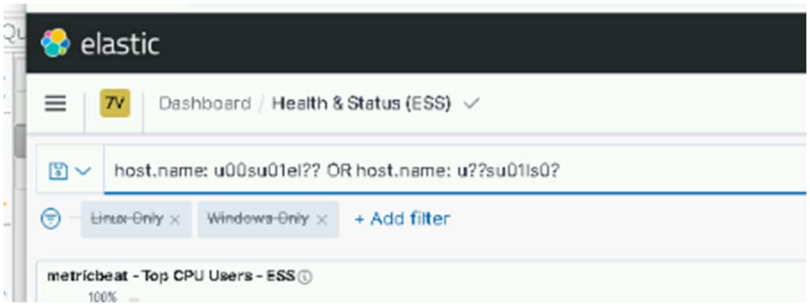


Figure 33 Example filter for Elastic OR Logstash hosts

NOTE: The filter uses Lucene syntax so be sure to turn off the KQL option before entering.

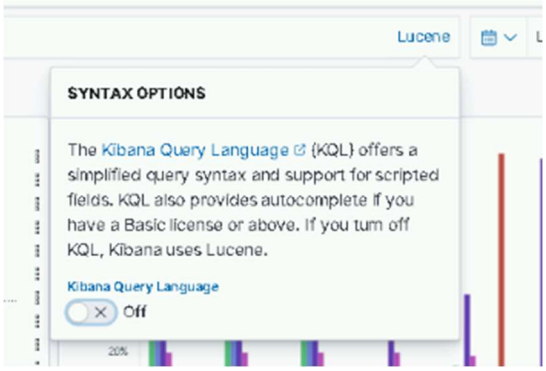


Figure 34 turning off KQL syntax for queries

NOTE: If any machine shows a partition at or reaching capacity, you will need to log in to that host and resolve the disk space issue.

Things to check:

1. Size of /var/log/messages file.
   If there is a misconfiguration or another possible issue on a host the /var/log/messages file may be overwhelmed with entries causing the /var/log partition to fill up. If this file seems too large you can try forcing a log rotation by doing the following:

```
# cd /etc/logrotate
# logrotate -f syslog
```

2. Check the log directory sizes for Logstash and Elasticsearch depending on the problem machine:

   Logstash logs directory - /var/log/logstash
   Elasticsearch logs directory - /var/log/Elasticsearch

   Both of these directories should be limited to 2GB in size via log4j settings.

2.3.5 elasticDataCollector

The elasticDataCollector is a custom Python application that runs as a service on every Logstash VM. Detailed information about the elasticDataCollector can be found in the *ES-018 - Elastic Logging and Aggregation Cluster (ELAC) - Upgrade to 7.12.1 Instructions* document. This document provides details on verifying the elasticDataCollector is running properly. If the elasticDataCollector is not running or is having difficulty on a Logstash instance, the information in the **dcgs-healthdata-iaas-ent** index may be incorrect for the site. The IAAS-ES-SYTEM-Dashboard and its sub-dashboards may also not have the correct information for the problem site.

2.3.5.1 Verify elasticDataCollector Service Running on All Logstash Instances

The elasticDataCollector should always be running on every Logstash instance. The dsil_elastic_servers puppet module does have logic to ensure that elasticDataCollector is always running, but you can create a custom visual to verify if the service is running on all the Logstash VMs. A visual to do this is not included in the release, but it is recommended that one be created. Following is an example of a visual that would show the elasticDataCollector running on each Logstash instance. This example is a good starting point for your visual.

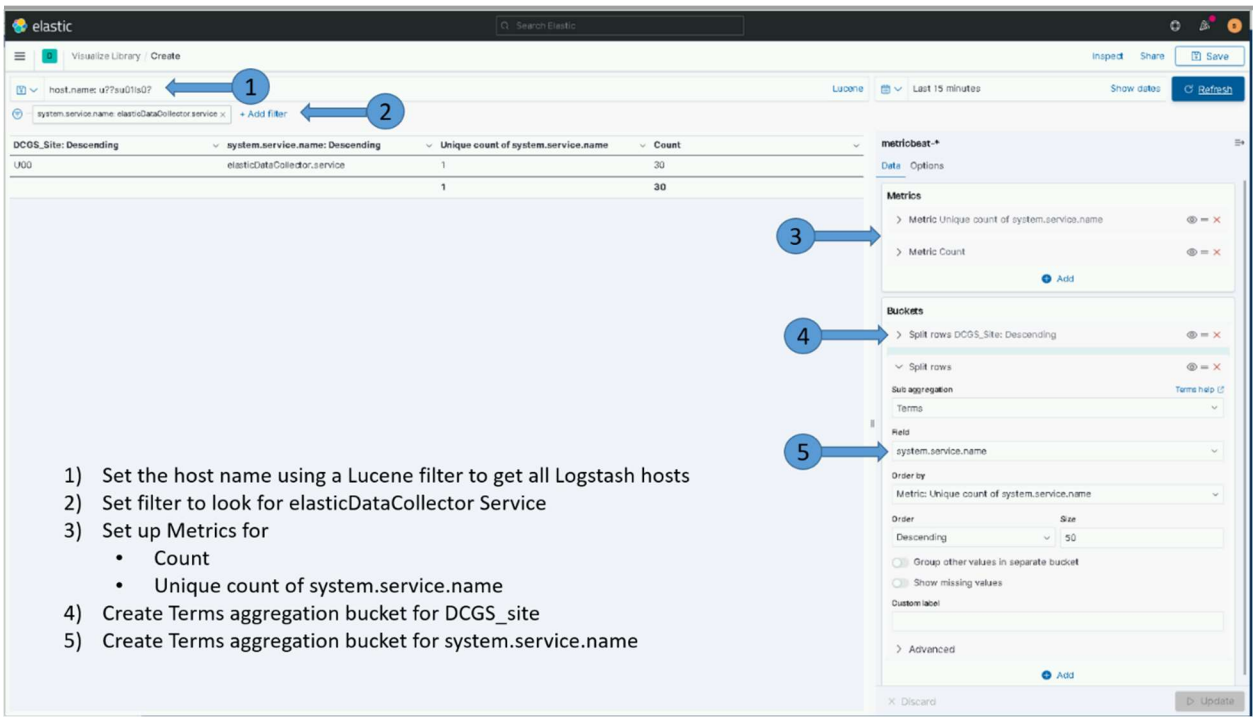


Figure 35 Example visual to monitor elasticDataCollector service on Logstash VMs

2.3.5.2 Verify elasticDataCollector Is Not Generating Critical Errors or Exceptions

The elasticDataCollector is a Python application, and just like any other application, may encounter errors or exceptions while running. All output from the elasticDataCollector is writing to the /var/log/messages file on the Logstash instance and is ingested into the linux-syslog index in Elastic. To monitor if any of the elasticDataCollectors are exhibiting problems, a saved search should be created that shows any output from the Python application. This saved search is not supplied with the release, but it is recommended that one be created. Following is an example of how to set up a search for Python output from all the Logstash instances.

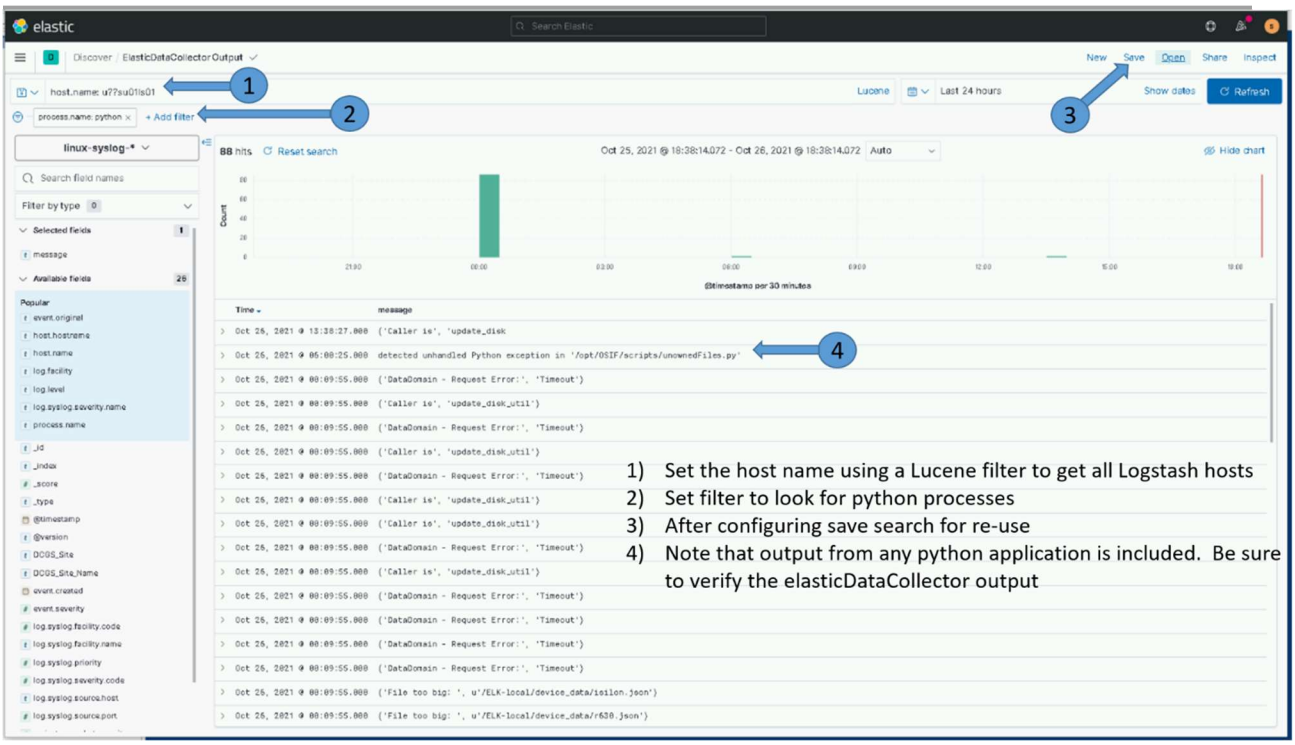


Figure 36 Example saved search to monitor Python applications on Logstash VMs

The elasticDataCollector will output informational data while it is running. Some common outputs are:

- ('File too big:', <path and filename>) - This is normal and is a notification that output files are being rotated because they have exceeded the maximum size set in constants.py
- ('<Device> - Request Error:', Timeout') – The named device had a timeout on a request. Although a valid error, we have seen this happen periodically on some devices like the DataDomain. This should not be a concern unless the timeout occurs on all attempts.

If the elasticDataCollector output shows an exception it may need to be restarted. The following is an example of the elasticDataCollector output with an error that would require a restart of the service.

```
Oct  8 15:01:57 u00su01ls01 python: ('DataDomain - Request Error:', 'Timeout')
Oct  8 16:14:02 u00su01ls01 python: ('Caller is', 'Traceback (most recent call last):
Oct  8 16:14:02 u00su01ls01 python: File "/etc/logstash/scripts/Device.py", line 280, in run
Oct  8 16:14:02 u00su01ls01 python: self.work()
Oct  8 16:14:02 u00su01ls01 python: File "/etc/logstash/scripts/CiscoSwitch.py", line 415, in work
Oct  8 16:14:02 u00su01ls01 python: self.update_sensors()
Oct  8 16:14:02 u00su01ls01 python: File "/etc/logstash/scripts/CiscoSwitch.py", line 233, in update_sensors
Oct  8 16:14:02 u00su01ls01 python: self.sensor_info[key].entSensorValueTimeStamp == "0"
Oct  8 16:14:02 u00su01ls01 python: AttributeError: Document instance has no attribute 'entSensorValueTimeStamp'
Oct  8 16:57:23 u00su01ls01 python: update_disk_util')
Oct  8 16:57:23 u00su01ls01 python: ('DataDomain - Request Error:', 'Timeout')
Oct  8 16:57:23 u00su01ls01 python: ('Caller is', 'update_disk_util')
Oct  8 16:57:23 u00su01ls01 python: ('DataDomain - Request Error:', 'Timeout')
Oct  8 16:57:23 u00su01ls01 python: ('Caller is', 'update_disk_util')
Oct  8 16:57:23 u00su01ls01 python: ('\nThread: ', 'ech-nexus5k', ' Died unexpectidly, the application needs restarted')
```

```
Oct  8 16:57:23 u00su01ls01 python: ('\nThread: ', 'ech-nexus5k', ' Died unexpectidly, the application needs restarted')
```

At 16:12:02 the output first shows that an exception in the code has happened and gives the stack trace information of where the exception occurred. The application is broken into threads, and some exceptions, although noteworthy, do not cause a thread to exit. However, in the previous example, at 16:57:23, the application started logging that the thread for the **ech-nexus5k** device died unexpectedly and that the application needs to be restarted. If you see any exceptions or errors like those, the elasticDataCollector service for the Logstash VM named in the output should be restarted.

NOTE: Any exception information should also be recorded and sent to the Elastic Development team so the issue can be investigated and remediated in future releases.

3 PKI Certificates

PKI Certificates are used to setup TLS communication on DCGS. Communication that is encrypted includes the following:

- Communication between Elasticsearch nodes
- Data sent from Logstash to the Elasticsearch cluster
- Data sent from beats and other collectors to Logstash
- Requests from Logstash to Databases and other endpoints

Use the following to guide you when new certificates are required because the current ones are expiring or the certificate authority has changed.

3.1 Obtain new PKI Certificates

When the certificates used by Elasticsearch and Logstash are going to expire or new certificates are needed because of a certificate authority (CA) change you must obtain new certificates. Once obtained, the certificates should be placed in the **install/certs** directory on the repo server so they are available for distribution.

NOTE: Backup/move existing certificates that are currently in the install/certs directory, do not overwrite.

3.1.1 Elastic Certificates (includes Kibana)

Certificates are needed for each Elasticsearch node. Elastic Certificates contain the following:

CN: hostname of Elastic VM (ex: u00su01el01.ech.dcgs.mil)

Aliases:
- fully qualified hostname (ex: u00su01el01.ech.dcgs.mil)
- hostname (ex: u00su01el01)
- hostname.{first segment of domain} (ex: u00su01el01.ech)
- elastic-node-{x} (ex: elastic-node-1)
- elastic-node-{x}.{first segment of domain} (ex: elastic-node-1.ech)

Additional Aliases if Kibana runs on the VM:
- kibana
- kibana.{first segment of domain} (ex: kibana.ech)
- kibana.{fully qualified} (ex: kibana.ech.dcgs.mil)

A convenience script is provided to make the creation of the Elastic Server Certificate requests easy for the installer. To create PKI certificate requests for Elastic to run on the system:

1. Log in to any existing Linux server at the site where the Elastic Cluster will be installed and do the following from your home directory:

```
# curl -k
https://satrepo/pulp/content/oadcgs/Library/custom/Elastic_Client/Elast
ic_Files/install/make_elastic_csrs.sh | bash
```

2. When the script completes, 3 directories will be present in the location where it was run.
   - Reqs: This directory holds the CSR Request information in text format
   - Keys: The private key associated with the certificate request for each Elastic node
   - CSRs – The actual PKI Certificate Request for each Elastic node.

3. The *.key and *.csr files should be submitted to the certificate authority for the system the Elastic Cluster is being installed on to obtain public certificates for each node. For systems submitted to the JWICS certificate authority (i.e. CTE High or Enterprise High), also include a text file named SANS.txt listing the SubjectAlternativeNames listed in the CSR (for each CSR).

NOTE: In this installation, Kibana will run on Elastic nodes and use the certificate for the node it runs on.

4. Once the Elasticsearch node certificates have been obtained, both the new certs and the keys for each node should be copied to the **install/certs** directory of the Elastic repo on the repo server ({xxx}su01ro01). Make a backup of the certificates that are already there before overwriting.
5. If needed update the elastic_cachain.pem file in the install/certs directory with the new root certificate from the certificate authority for the system being installed. (See Root Certificates below)

PKI Certificates must be in the following format for the installation scripts to work properly:

Public Cert: {hostname}.crt	examples: u00su01el01.crt, u00su01el02.crt
Private Keys: {hostname}.key	examples: u00su01el01.key, u00su01el02.key

3.1.2 Logstash Certificates

A Logstash instance runs at each DCGS site to collect data for that site. A PKI certificate is needed for each Logstash Instance. Do this to obtain a Logstash certificate request for each site.

Logstash Certificates contain the following:

CN: hostname of Logstash VM (ex: u00su01ls01.ech.dcgs.mil)

Aliases:

- fully qualified hostname (ex: u00su01ls01.ech.dcgs.mil)
- hostname (ex: u00su01ls01)
- hostname.{first segment of domain} (ex: u00su01ls01.ech)
- logstash (ex: logstash)
- logstash.{first segment of domain} (ex: logstash.ech)

A convenience script is provided to make the creation of the Logstash Server Certificate requests easy for the installer. To create PKI certificate requests for Logstash to run on the system:

1. Log in to any existing Linux server at the site where the Logstash Server is to be installed and do the following from your home directory:

```
# curl -k
https://satrepo/pulp/content/oadcgs/Library/custom/Elastic_Client/Elast
ic_Files/install/make_logstash_csr.sh | bash
```

2. When the script completes, 3 directories will be present in the location it was run
   - Reqs: This directory holds the CSR Request information in text format
   - Keys: The private key associated with the certificate request for the Logstash instance
   - CSRs: The actual PKI Certificate Request for the Logstash instance.

3. The *.key and *.csr files files should be submitted to the certificate authority for the system the Elastic Cluster is being installed on to obtain public certificates for each node. For systems submitted to the JWICS certificate authority (i.e., CTE High or Enterprise High), also include a text file named SANS.txt listing the SubjectAlternativeNames listed in the CSR (for each CSR).

4. Once the Logstash certificate has been obtained, both the new cert and the key for each Logstash instance should be copied to the **install/certs** directory of the Elastic repo on the repo server ({xxx}su01ro01) for the associated site. Make a backup of the certificates that are already there before overwriting.

PKI Certificates must be in the following format for the installation scripts to work properly:

Public Cert: {hostname}.crt	examples: u00su01ls01.crt
Private Keys: {hostname}.key	examples: u00su01ls01.key

3.2 Certificate Authority (CA) changed

This will happen if there is a new CA or an update to an existing CA. If the root certificate did change then the following will need to be done before using certificates issued from the new CA. If the new certificate were issued from the same CA then this section can be skipped.

3.2.1 Distribute new CA

1. Windows Clients - The cachain.pem file provided as part of SCCM package needs to be updated to include new CA root certificate and must be distributed to all Windows hosts. Consult with the SCCM SME to ensure the cachain.pem file that is part of the SCCM Elastic package is updated with the new root certificate. The hosts are automatically updated once a day or the SCCM SME can do a manual update to distribute the new certificate faster.
2. Linux Clients - The ca-bundle.crt file needs to be updated on all Linux hosts via puppet. Consult with a puppet SME and ensure that the new CA certificate is added and distributed by the osif_pki module. This module will add the new certificate into the /etc/pki/ca-trust/source/anchors directory and execute an update-ca-trust command on all Linux to update the ca-bundle.crt with the new certificate.

3. Elastic nodes – The cachain.pem in the **/etc/elasticsearch/certs** directory on each node must be updated to contain the new root certificate. The elastic_cachain.pem file in the install/certs directory on the repo server should have been updated with the new certificate authority and can be copied to each host.
    a. Elasticsearch will detect the new cachain.pem file and update automatically
    b. On nodes where Kibana is running
        i. The cachain.pem in the **/etc/kibana/certs** directory must be updated to contain the new root certificate. In this case you can just copy the cachain.pem that was updated in the **/etc/elasticsearch/certs** directory.
        ii. After updating the cachain.pem for kibana you must restart kibana

```
# systemctl restart kibana
```

4. Logstash hosts – The cachain.pem in the **/etc/logstash/certs** directory on each Logstash instance must be updated to contain the new root certificate. The elastic_cachain.pem file in the install/certs directory on the repo server should have been updated with the new certificate authority and can be copied to each Logstash instance.
    a. After updating the cachain.pem the Logstash instance must be restarted

```
# systemctl restart logstash
```

3.2.2 Restarting Beats on Linux Hosts

When SCCM distributes the new cachain.pem file to the windows hosts the beats that are running on each host are restarted so they start using the new certificate right away. On Linux hosts when puppet updates the ca-bundle.crt the beats will not automatically detect the change and need to be restarted manually to use the new certificate for validations. Puppet runs every 30 minutes and normally only restarts a beat if there was a configuration change on the host. To modify the Elastic Clients puppet module to restart all beats installed on every puppet run do the following.

1. Login to the puppet console
2. Select “Node groups” 🡒 “OEM Nodes” 🡒 “Elastic Search Nodes” 🡒 “Elastic Clients”

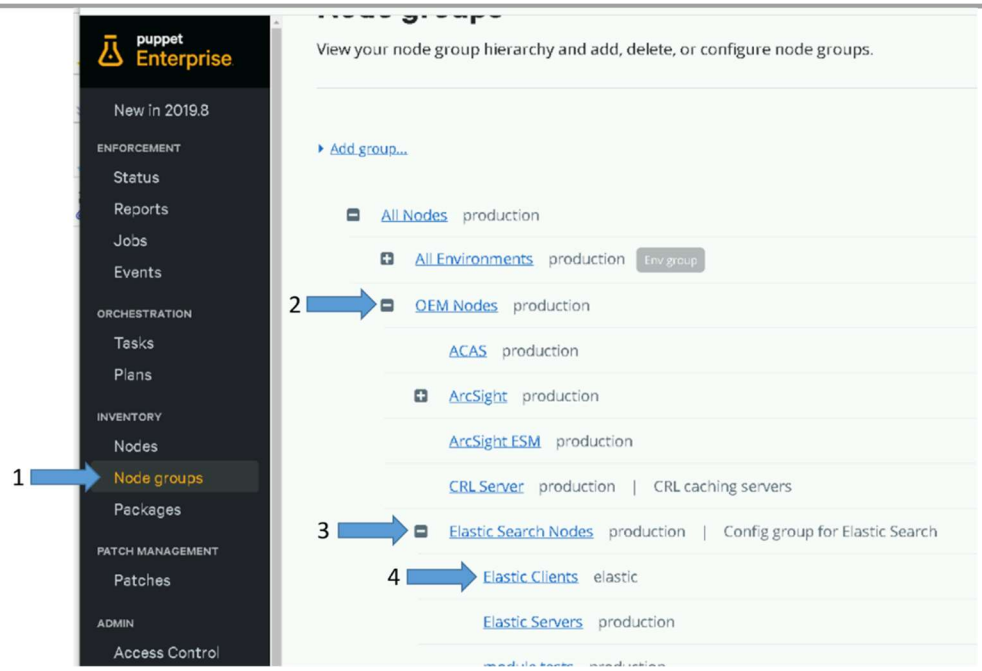


Figure 37 - Access Elastic Clients Module

3. Select "Classes"

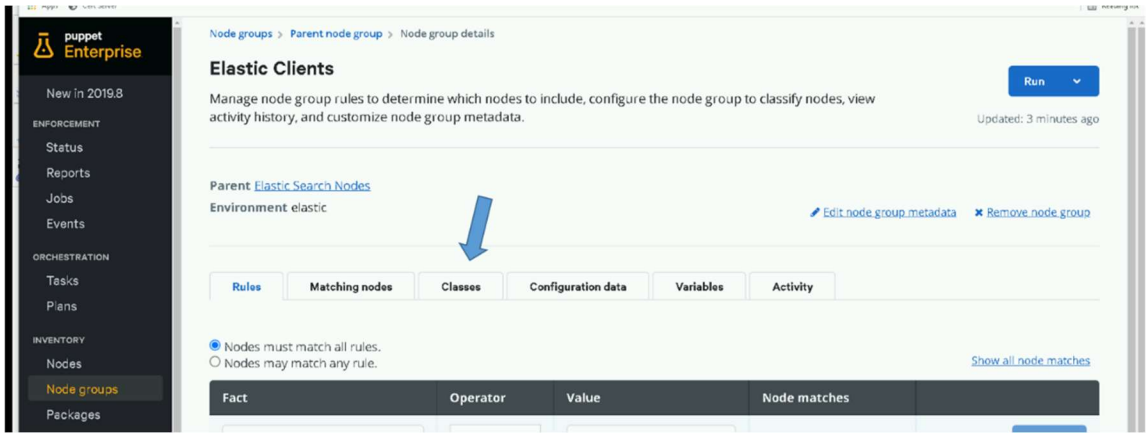


Figure 38- Select classes for Elastic Clients

4. Select "restart_beats" parameter

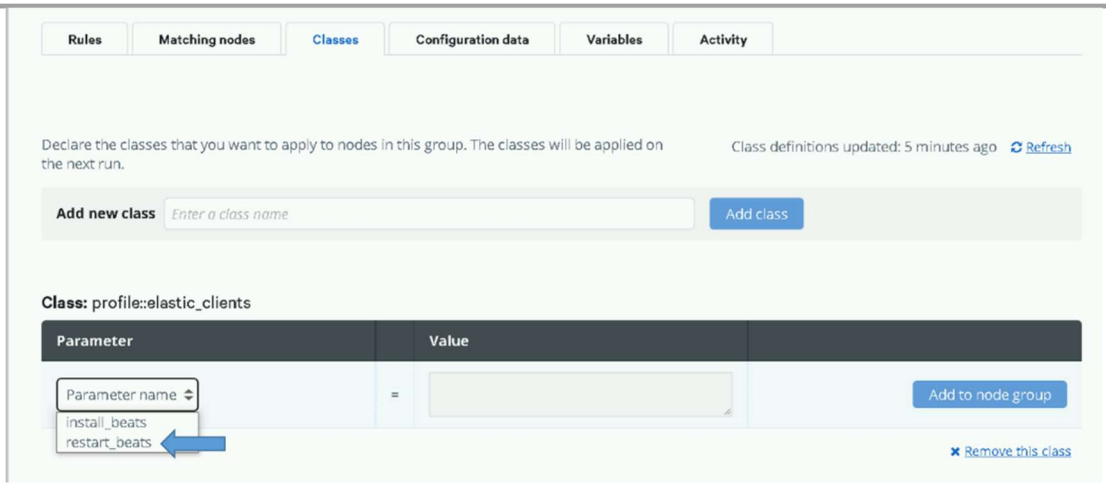


Figure 39- Select restart_beats parameter

5. Set restart_beats parameter to “true” and select “Add to node group”

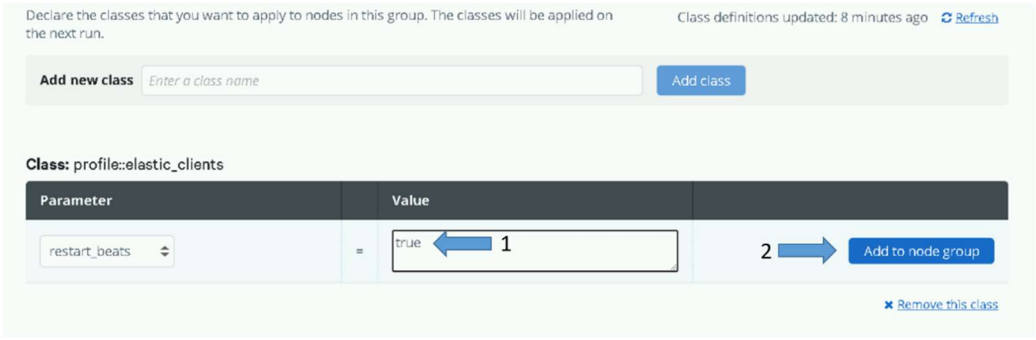


Figure 40- Set restart_beats to "true" and select "Add node to group"

6. Select “Commit 1 change”

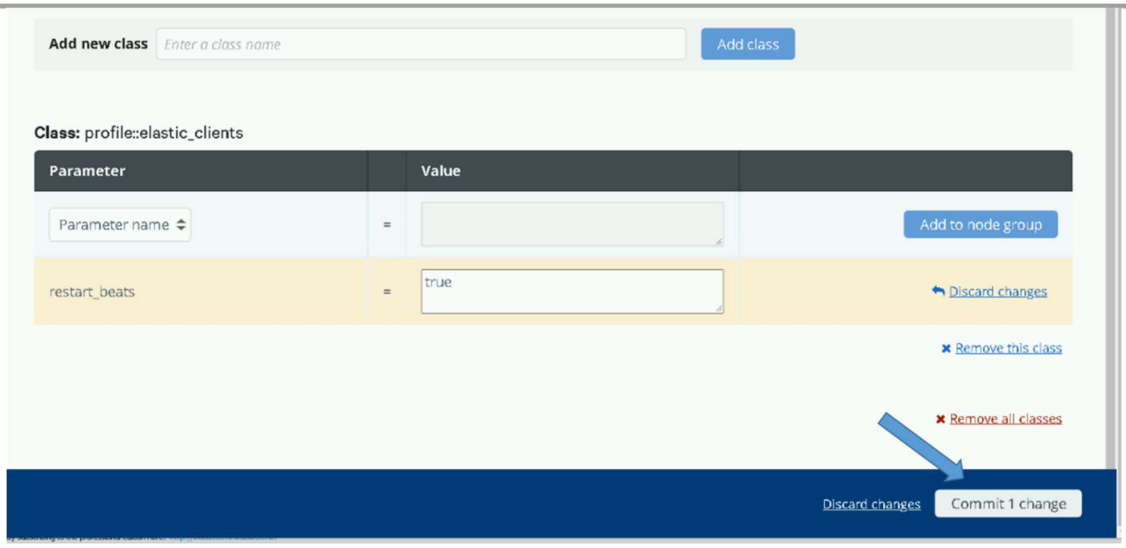


Figure 41- Select "Commit 1 change"

7. restart_beats parameter is now set to true

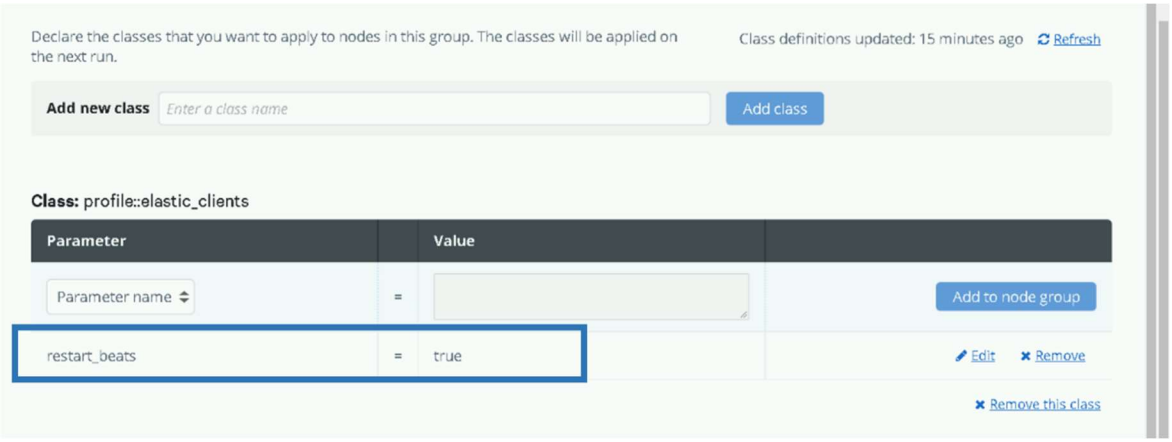


Figure 42- restart_beats parameter is set to "true"

8. Now you can either use the puppet console to setup a job that runs puppet on all hosts or just wait for it to run automatically. Having the beats restart on every run does not affect data collection but the restart_beats parameter should not be permanently left in this configuration. You can validate that beats are communicating correctly with Logstash by checking the /var/log/logstash-plain.log log file on each Logstash host. Note that this log file is ingested into Elasticsearch so you can use the Discover tab on the Filebeat index to look for bad_certificate messages.

Below is an example of how to setup filters on the Discover tab to look for bad_certificate messages.

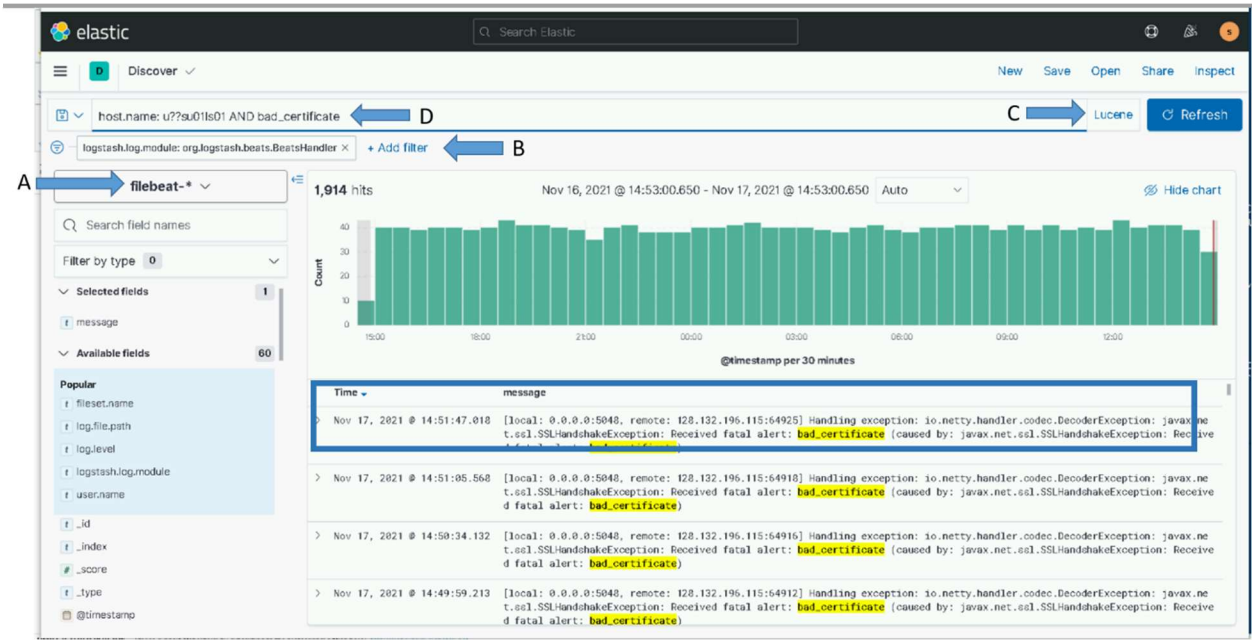


Figure 43- Search for bad_certificate using Kibana

A. Select "filebeat-*" index pattern
B. Setup filter: logstash.module: *org.logstash.beats.BeatsHandler*
C. Select "Lucene" for Query language
D. Add filter for Logstash hosts and bad_certificate : *host.name: u??su01ls01 AND bad_certificate*

Note that if some hosts are still having issues communicating with Logstash there could be an issue with puppet running on the host. You can login to the host and try running puppet manually (puppet agent –t) to see if there is an issue. If you find hosts that have issues running puppet contact the puppet SME to get them resolved. If they cannot be resolved you may need to update the ca-bundle.crt and restart the beats manually for the problem hosts.

9. After you are satisfied that all hosts are communicating with Logstash properly using the new certificates you can go back to normal puppet runs without having the beats restarted on each run. To do this simple remove the parameter setting that was made above. The default value for this parameter is "false" so if it's not set manually in the puppet console the value will be "false".

Select "Remove" next to the parameter on the "Classes" tab of the Elastic Clients module

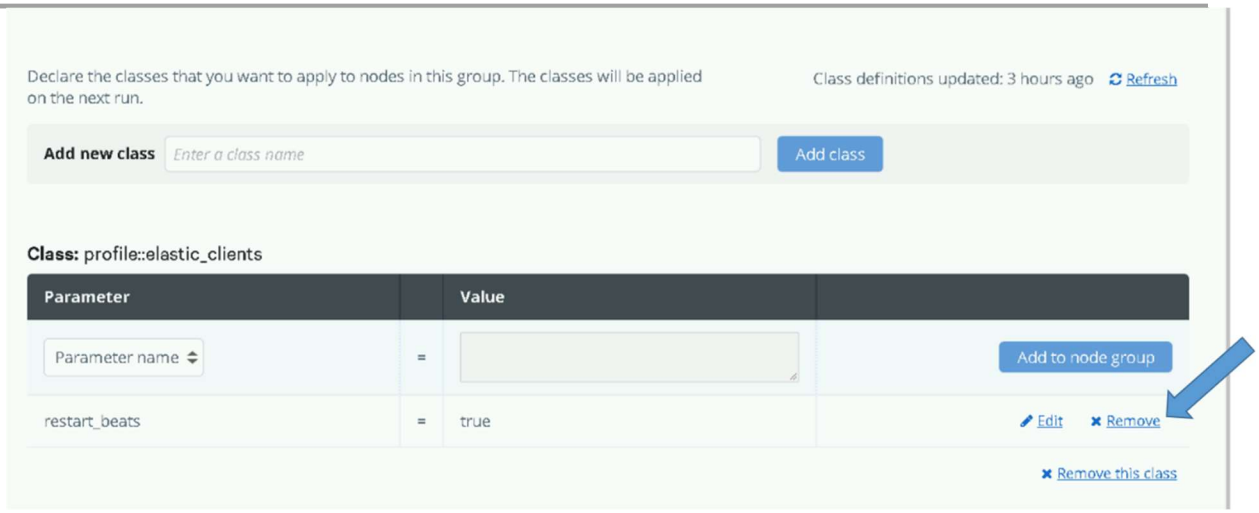


Figure 44- Remove restart_beats parameter

10. Commit the removal by selecting the "Commit 1 change" button

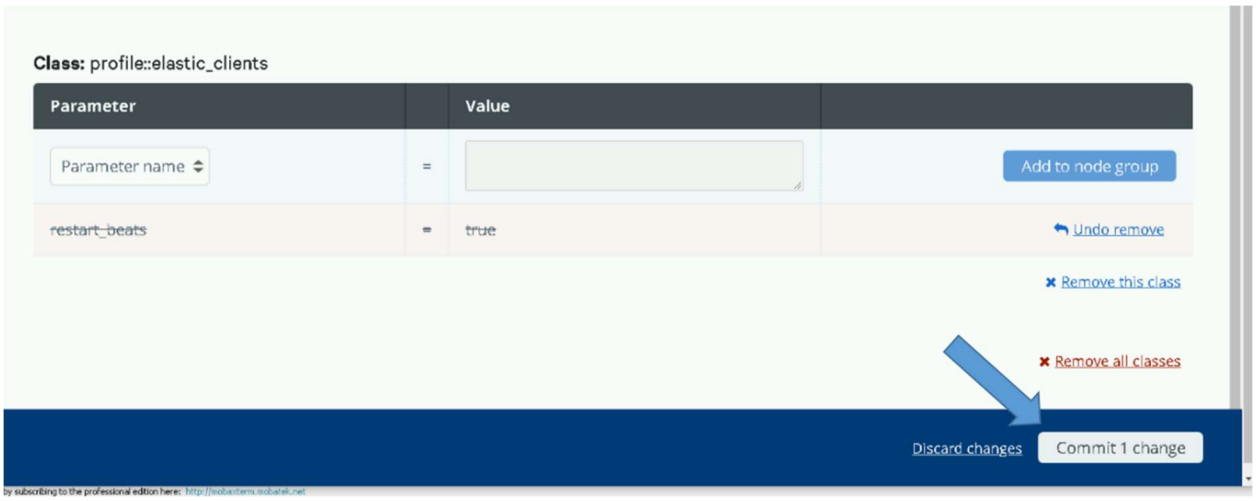


Figure 45 - Commit removal of restart_beats parameter

11. Beats will no longer be restarted on puppet runs

3.3 Updating certificates on Elastic hosts

The new certificates for the Elasticsearch nodes must be placed in the **/etc/elasticsearch/certs** directory on each node. Backup the existing certificates temporarily in case there are issues with the new certs.

1. Copy the new certificate and key files for each node from the **install/certs** on the repo server to the /**etc/elasticsearch/certs** directory. Elasticsearch will detect the updated certificates automatically.

2. On nodes where Kibana is running
    a. Copy the new certificate and key files for the node from the /**etc/elasticsearch/certs** directory to **the /etc/kibana/certs** directory
    b. Restart Kibana – It does not automatically detect the new certificate

3.4 Updating certificates on Logstash hosts

The new certificates for the Logstash hosts must be placed in the **/etc/logstash/certs** directory on each node. Backup the existing certificates temporarily in case there are issues with the new certs.

1. Copy the new certificate and key files for each Logstash host from the **install/certs** on the repo server to the /**etc/logstash/certs** directory on the host.
2. Create new pkcs8.key for the Logstash instance

    # openssl pkcs8 -topk8 -in /etc/logstash/certs/<hostname>.key -out /etc/logstash/certs/<hostname>_pkcs8.key -passout pass:<password>

    Note: If you do not know the ssl key password consult an Elastic SME
3. Restart Logstash

    # systemctl restart logstash

4. If new root certificate was distributed restart heartbeat to ensure it picks up new cert

    # systemctl restart heartbeat-elastic

3.5 Check for errors

Check for SSL errors in the following logs to ensure everything is working correctly with the new certificates.

- Logstash - /var/log/logstash/logstash-plain.log
- Elasticsearch nodes - /var/log/elasticsearch/XXX_Cluster.log (XXX will be the site name of the cluster, example ECH_Cluster.log)

4 Elastic Service account

The elastic service account (xx_elastic.svc) is used by the cluster and at each site for the following:

Elastic Cluster

The Elasticsearch service (application) is run on each VM (node) in the cluster as the elastic service account.

- During installation, Elasticsearch is modified to be run by the Elastic service account. Elastic relies on the Kerberos ticket for the Elastic service account to be automatically updated by the SAKM scripts.

Each Site

The Elasticsearch service account at each site is used on the Logstash VM for the following:

- The get_ldap_hosts script used to dynamically configure heartbeat uses the service account to query hosts from Active Directory. This script relies on the Kerberos ticket for the Elastic service account to be automatically updated by the SAKM scripts.
- Metricbeat vSphere module – Password is stored in Metricbeat keystore as "V_PWD"
- Several pipelines use the service account to query databases. The password for this is stored in the Logstash keystore as "es_svc_acct_password"
- The Elastic Data Collector uses the service account password for the following:
    - To query data from VCenter
    - To query data from devices that can use active directory accounts for access. Currently this is the Isilon and XtremIO devices.

4.1 Updating the service account password for a DCGS site

When the elastic service account password needs to be updated at a site the following must be completed to ensure that Elastic continues to operate successfully.

4.1.1 SAKM keytab updates on all hosts

The service account Kerberos ticket is used on Elasticsearch nodes and Logstash hosts. The keytab file holding the service account password is used by SAKM to periodically update the ticket and must be updated with the new password on all hosts.

- If the site is the location of the Elastic cluster (example: ECH, ISEC) the keytab file used by SAKM needs to be regenerated with the updated password on each Elasticsearch node in the cluster. Do the following on each Elastic node:
- Also do the following on the Logstash host at the site of the service account password change.

    1. Login to the host and sudo to root
    2. Sudo to service account

su – xx_elastic.svc

3. Remove existing keytab file (replace xx with site designator, example: 00 or 24)

   $ rm /usr/local/etc/sakm/xx_elastic.svc/xx_elastic.svc.**keytab**

4. Recreate keytab file with updated service account password
   NOTE: DOMAIN must be in all caps. (example: DCGS.MIL)

   $ **ktutil (click enter)**
   ktutil: **addent -password –p** xx_elastic.svc@<DOMAIN> **-k 1 -e aes256-cts-hmac-sha1-96 (hit enter)**
   Enter Password for <service account name>@<DOMAIN> (you will not see the entry, and this password will not be automatically checked for complexity on the domain)
   > **wkt** /usr/local/etc/sakm/xx_elastic.svc/xx_elastic.svc.**keytab (enter)**
   > **exit**

5. Change permissions for the keytab.

```
$ chmod 700 /usr/local/etc/sakm/xx_elastic.svc/xx_elastic.svc.keytab
$ chown xx_elastic.svc /usr/local/etc/sakm/xx_elastic.svc/xx_elastic.svc.keytab
$ exit
```

6. Test the keytab is working

```
$ kinit XX_elastic.svc -k -t /usr/local/etc/sakm/XX_elastic.svc
/XX_elastic.svc.keytab
```

No errors should be returned.

7. If not last node in cluster return to step one for next node

4.1.2 Service account password updates on Logstash VMs

Perform the following on the Logstash host (ls01) at the site where the service account is updated:

4.1.2.1 Elastic Data Collector

Use the following steps to update the service account password stored for use by the data collector.

Note: To execute a portion of this update you must have access to MobaXterm. If this tool cannot be used contact an Elastic SME for guidance on an alternative.

- Update vsphere.dat – Used by vSphere API to pull audit data from vCenter
  1. Login to the Logstash VM for the site (xxsu01ls01)
  2. sudo to root

3. update the password used by the Vsphere class

   # cd /etc/logstash/scripts

   # python updatepasswd.py

   You will be prompted to enter the service account password 2 times to ensure it is entered correctly.

- Update any devices that are using the service account for access. Currently this is any XtremIO/Isilon device. Use configurator to update password for XtremIO and Isilon access. Instruction for this are provided in most install instructions deliveries.

To use this tool you must be able to open a window from the Logstash box. The easiest way to do this is by using MobaXterm which has an embedded X Server. To verify X11 is set up correctly, do the following:

1. Connect to the Logstash VM with a new MobaXterm session.
2. Elevate to root – "sudo su"
3. Verify X11-forwarding is enabled. There will be a green check box next to **X11-forwarding**, as shown in the following figure.

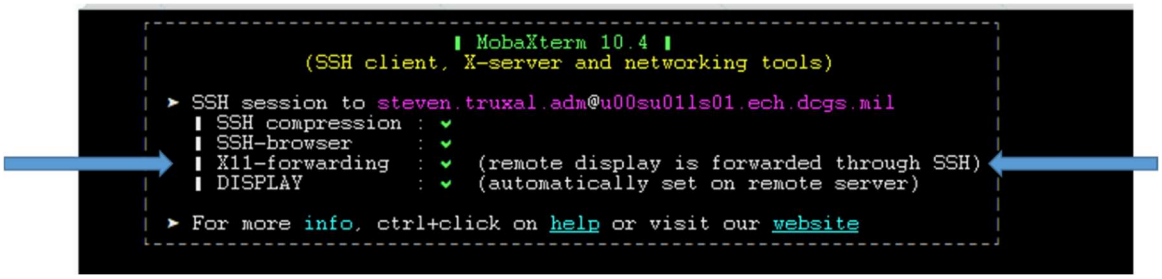


Figure 46 X11 Forwarding Setup Successfully

NOTE: When Puppet is running, X11-forwarding will be automatically disabled. To run the configurator you must manually enable X11-forwarding by executing the following steps.

1. Execute the following command:
   ```
   # sed -i 's/X11Forwarding no/X11Forwarding yes/g' /etc/ssh/sshd_config
   ```
2. `# systemctl restart sshd`
3. Retry the previous test to verify X11 forwarding is set up correctly.

NOTE: if the steps above are still not getting X11-fowarding enabled one last thing to try is to ensure the AddressFamily setting is correct.

1. # cd /etc/ssh
2. # vi sshd_config
3. look for the line setting for "AddressFamily". If it is comment out, then uncomment and verify it is set to "inet"

Should be: AddressFamily inet

4. Save changes (:wq)
5. # systemctl restart sshd
6. Retry the previous test to verify X11 forwarding is setup correctly.

IMPORTANT: If you cannot get X11-forwarding enabled, **STOP**, and consult a Linux SME for help.

Edit Collector Configuration

NOTE: The previous step must be successful to continue. The GUI will not display if X11-forwarding is not enabled on the Logstash VM.

1. Log in to the Logstash VM with your .adm account (Do not assume root until after this command) and list the xauth cookies.

```
xauth list
```

You may see multiple cookies if X11-forwarding is enabled on other hosts. Take note of the cookie for this host; you will see the ls01 host name at the beginning.

NOTE: If there are multiple entries for the ls01 host the last entry is most likely the one you will use. To make sure, execute **echo $DISPLAY** to get the correct number of the display for your SSH session. Use the cookie line that has both **ls01** and the display number.

Figure 47 xauth list example with logstash host

2. Sudo to become root.
   `# sudo su`
3. Copy the entire xauth cookie line and add it to the roots xauth cookies.
   `# xauth add <cookie>`
   Example:

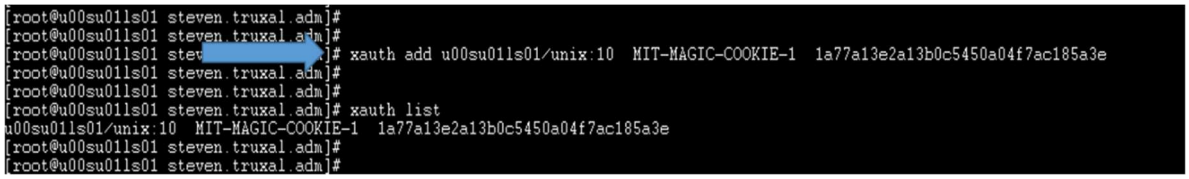


Figure 48 xauth add <cookie>

4. Run the configurator GUI.

```
# cd /etc/logstash/scripts
# . ./venv/bin/activate
# python ./configurator.py
```

5. The device configuration window, **The Configurator**, displays.
NOTE: If the window does not come to the foreground, look for an icon in the taskbar.

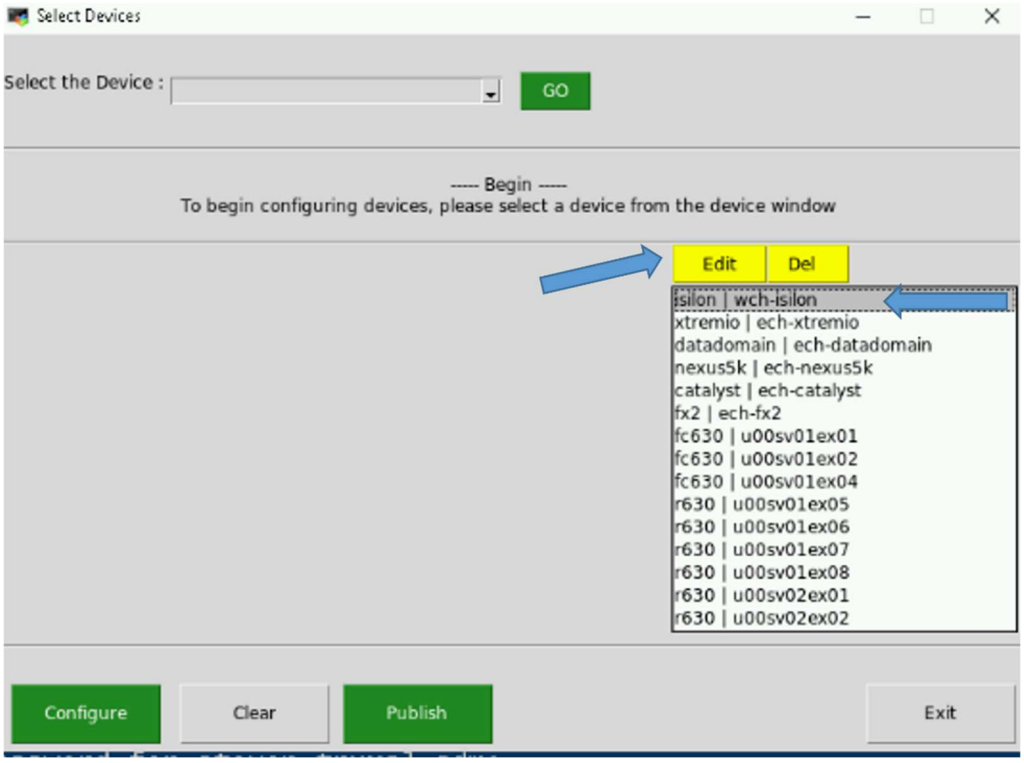


Figure 49 Device Configuration GUI

NOTE: If the data collector was already installed on a previous version the devices that are currently set up for monitoring will be displayed. If there are no devices configured then there will be no service account passwords to update.

6. Select the Isilon or XtremIO to update and then select “Edit”

7. Remove the asterisks hiding the current password and type in the updated password and select "Save"

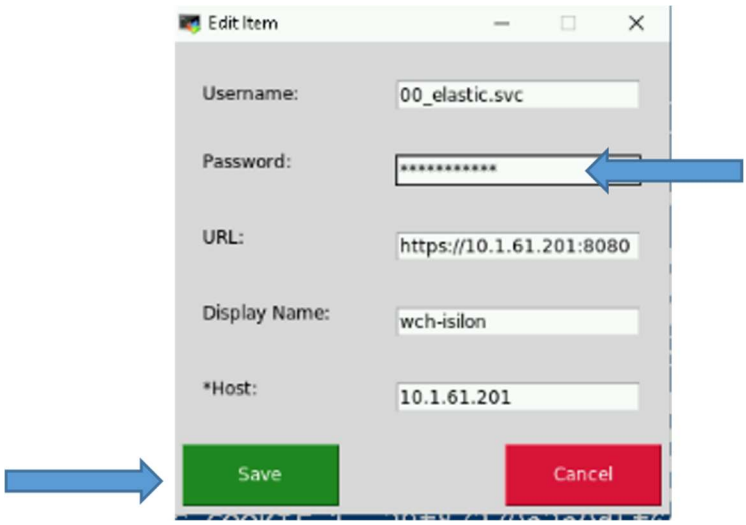


Figure 50 Update password

8. Repeat this for all devices that use the service account for access.
9. press **Publish** to test your updated configuration.

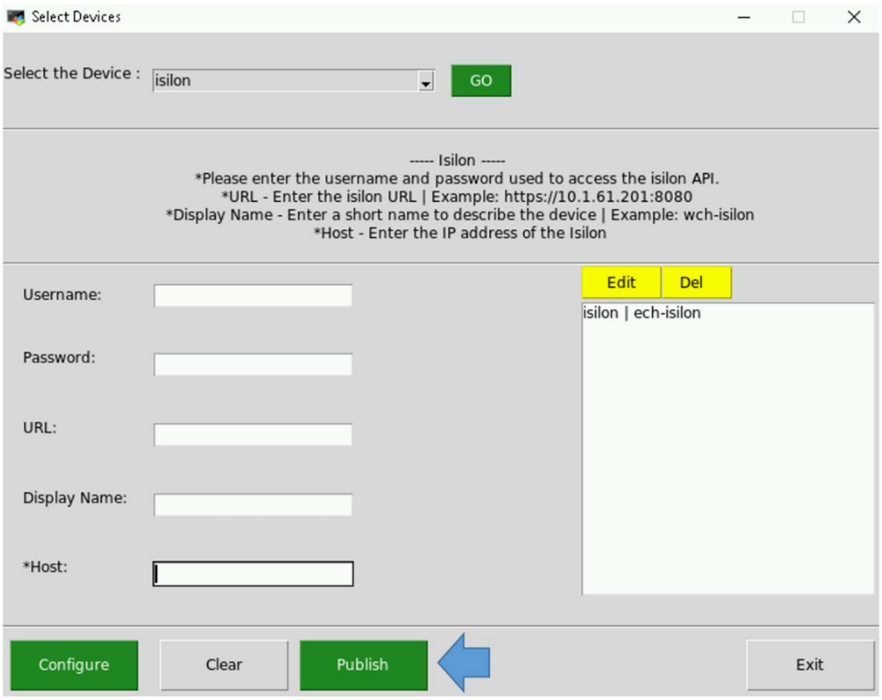


Figure 51 Publish Device Configuration

10. The Configurator will verify the values you have entered by trying to access the device(s). If all goes well your configuration will be updated and a success message will be displayed.

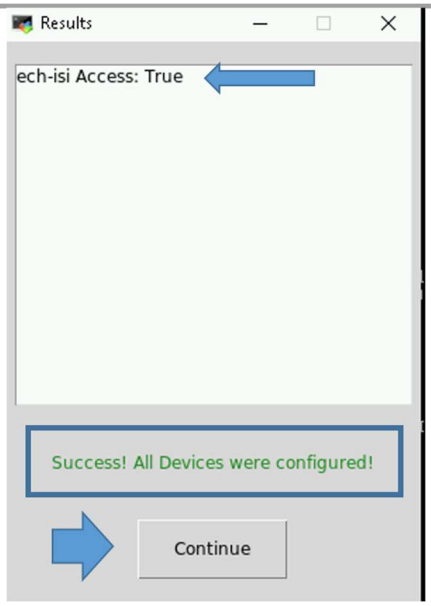


Figure 52 Successful Publish

a. If any of your access parameters are incorrect the **Publish** will fail, and the configuration file will not be updated.

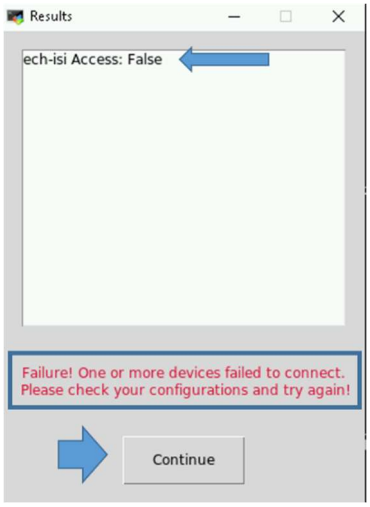


Figure 53 Unsuccessful Publish

11. Select the **Continue** button in the **Results** window to continue. If you entered information incorrectly and the **Publish** failed, you can either **Edit** or **Del** the device that failed.

NOTES:

- If any of the devices fail, the configuration file is NOT updated. Access to all devices must be successful to create/update a device configuration file.
- Every successful publish will restart the elasticDataCollector service so the current configuration is read in and collection from the updated devices is started.

12. If the **Publish** failed, you can use the **Edit** feature again to correct the issue. For security reasons the password is not shown, so if everything else looks correct try re-entering the password. To review this feature, select the device you'd like to modify and press the **Edit** button.
13. If the **Publish** was successful then click **Exit.**

NOTES:

- You can run the configurator GUI at any time to either view, modify, or add devices for this Logstash instance to monitor.

  Every successful publish will restart the elasticDataCollector service so the current configuration is read in and collection from the devices is started.

4.1.2.2 Metricbeat Keystore

vsphere.yml - Uses service account to access vSphere - password is stored in Metricbeat keystore
Remove the existing "secret" in Metricbeat keystore and re-add.

1. Login to Logstash VM for the site (xxsu01ls01)
2. sudo to root
3. remove entry from Metricbeat keystore
   ```
   # metricbeat keystore remove V_PWD
   ```
   Note: You may see a message that "the keystore doesn't exist", in this case it will be created in the next step.

4. Re-add entry to Metricbeat keystore
   ```
   # metricbeat keystore add V_PWD
   ```
   Note: If you see a message that "the keystore doesn't exist. Do you want to create it? …" enter **y** to create it.
   (Enter new service account password when prompted - Be careful, it does not prompt you twice)

5. Validate new password is working correctly by testing the vSphere module and ensuring that it is retrieving data.
   ```
   # metricbeat test modules vsphere
   ```
6. If step 5 is successful, then restart Metricbeat. If not return to step 3 and try again.
   ```
   # systemctl restart metricbeat
   ```
   Note: this may take a few minutes to complete but will return to the prompt when done.

4.1.2.3 Logstash Keystore

Logstash Pipelines (eracent, idm, sccm, sqlServer) use service account to access databases. The ES_SVC_ACCT_PASSWORD entry must be updated in the Logstash Keystore

1. Login to Logstash VM for the site (xxsu01ls01)
2. sudo to root
3. remove entry from Logstash keystore

```
# /usr/share/logstash/bin/logstash-keystore remove
es_svc_acct_password --path.settings /etc/logstash
```

4. Re-add service account password to Logstash Keystore

```
# /usr/share/logstash/bin/logstash-keystore add
es_svc_acct_password --path.settings /etc/logstash
```

5. Restart Logstash

```
# systemctl restart logstash
```

6. Verify no errors using new password in logstash log file

```
# tail -f /var/log/logstash/logstash-plain
```

7. Exit Logstash VM when complete.

5 Troubleshooting

5.1 Kibana Troubleshooting

5.1.1 Web Page Does Not Appear or is Not Working Properly

First verify Elasticsearch is running properly (2.3.1 Cluster Health). If the Elastic cluster health is Red, Kibana may have issues communicating. After verifying the health of the Elastic Cluster, proceed to the following troubleshooting steps.

If you cannot connect to the web page, you'll need to log into the elastic node containing the Kibana service. On OA DCGS this is dependent on the cluster size. See the following table for which elastic node Kibana is installed on in your cluster.

Table 9 Elastic Nodes where Kibana is installed

# of Nodes in Cluster	Elastic Nodes where Kibana is installed
7	Node 5 and Node 6
10	Nodes 7 and Node 10
15	Node 10 and Node 15

Log in to the node that should be running Kibana and execute the following Linux command:

```
# systemctl status Kibana
```

If the VM has a version of Kibana installed it should show some basic info about the product. Now that we can see the product is there, if the status command shows Kibana is not running then the service will need to be restarted. First, take note of any errors indicated in the output.

Try to restart the service:

```
# systemctl start Kibana
```

Give it a minute and try to open the Kibana webpage again. To verify Kibana started successfully and stays running, execute the following command:

```
# systemctl status kibana
```

Verify Kibana is running and has been running since you started it earlier. If it runs for a small period of time and then restarts or exits, then there is another issue. At this point you will have to check the Kibana log file to determine what the issue is. The kibana.log file is located in **/var/log/kibana**. You can examine the log file for errors or try tailing it when you attempt to start Kibana to determine the issue.

NOTE: If you cannot determine the issue, please contact the Elastic SME at the DSIL for more guidance.

5.1.2 Grok Debugger Gives Error "Something Went Wrong"

Starting with the 6.8 version of Kibana, a JavaScript call to the function "trimEnd" was added to the grok_debugger.js code. This call is not available in older browsers. So, if you are using a browser before the following you will see this error:

Table 10 Browser Error

Browser	Must have at least
Firefox	Version 61
Chrome	Version 66
IE 11	Does not seem to work here

If you need to use the grok debugger this can easily be fixed by removing the calls to "trimEnd" from the javascript code in /usr/share/kibana/optimize/bundles/kibana.bundle.js. Open the file with vi and search for "trimEnd()". DO NOT REMOVE the calls to "trimEndRegexp"! The javascript code is on one continuous line in this file so it's not the prettiest to look at. There are 2 calls to "trimEnd" to delete:

- this.grokdebuggerRequest.rawEvent = rawEvent.trimEnd();
- this.grokdebuggerRequest.pattern = pattern.trimEnd();

Change them both to what they were before version 6.8:

- this.grokdebuggerRequest.rawEvent = rawEvent;
- this.grokdebuggerRequest.pattern = pattern;

This change can be found in the Kibana repository on GitHub here:
https://github.com/elastic/kibana/commit/3180052e6fec2beda11bd524cf36f1a986e73037

5.2 Elasticsearch (Cluster) Troubleshooting

5.2.1 Verify Elasticsearch is Running on a Node

On DCGS, Kibana communicates with the node it runs on. If that node is not available, then Kibana will not function properly. Also, in earlier versions of Elasticsearch, Kibana will not function at all until the cluster has a health of at least 50%. This is important to know because in some cases when the Kibana node goes the cluster is still running fine, but web access is lost.

For example, a 3-node cluster can usually function with the loss of one node, but if that node is the one Kibana communicates with your web access will not function. You can still access the cluster using "curl" commands as described in later sections of this document.

To see if a node is running on a particular VM you can run the following command:

```
# systemctl status Elasticsearch
```

If elastic is not running on the VM, see the next section 5.2.2 Missing Elastic Node for possible resolution.

If elastic is running but there are still issues, use the commands in 5.2.3 Check Elastic Cluster Health from Linux Command Line to troubleshoot.

5.2.2 Missing Elastic Node

If you have identified an instance of Elasticsearch that is not running, check the following:

1. Disk space

   Verify that the node does not have any disk space issues. If so correct and attempt to restart the node.

2. NFS share issues

   All Elastic nodes should have the elastic share on the Isilon mounted as /ELK-nfs. This share is currently used for elastic snapshots and is the data location for "WARM", "COLD", and "FROZEN" nodes. Elastic runs as the XX_elastic.svc account so it can access the elastic share on the Isilon (elac).

   Verify that the /ELK-nfs share is mounted and is accessible by the xx_elastic.svc account.

   Log in to Elastic node:

   ```
   # sudo su
   # su - XX_elastic.svc
   # cd /ELK-nfs
   ```

   A valid Kerberos ticket is required to access the Isilon share. If the service account cannot access this directory it may be because of an expired ticket. Each Elastic node should have SAKM installed to periodically renew the service accounts Kerberos ticket. You can check the status of the Kerberos ticket using the following command.

   ```
   # klist
   ```

   If the Kerberos ticket is expired, or there is no ticket at all, there is probably an issue with the Service Account Kerberos Management (SAKM) configuration on the node. Refer to the SAKM installation instructions or consult with a DCGS Linux SME to troubleshoot.

3. Isilon issues

   In previous versions of Elastic there has been an issue where the NFS share becomes inaccessible or attempts to write data to the share fail. The reason for this failure has been investigated by Dell but is still unknown. One possibility is that the Isilon cannot keep up with the volume of data that Elastic is writing and sometimes fails. If you log in to an Elastic node when it is having this issue you may not even be able do a directory listing on the /ELK-nfs share. You may see an I/O error when attempting. If you can cd to /ELK-nfs and attempt to write data to a file, this may also give an I/O error. Look in the **/var/log/elasticsearch/<cluster>.log** file and search for **Remote I/O error**. If the node is experiencing this issue you will see this multiple times in the log. This error usually causes the node to fail and sometimes Elastic will exit.

   Also note that this issue may be seen on more than one node at a time, which can cause a major issue with the cluster.

To recover when you see the /ELK-nfs partition inaccessible:

a. Make sure Elasticsearch is not running: `systemctl stop Elasticsearch`
You can do a status first; it may already be stopped.
b. Unmount the /ELK-nfs share.
c. Remount the /ELK-nfs share: `mount -a`
d. Verify that the /ELK-nfs share has been remounted and is accessible by the elastic service account.
e. Restart Elasticsearch: `systemctl start Elasticsearch`
f. Take steps to make sure Elastic Cluster returns to 100% health.

Sometimes the previous instructions fail at step 2 or 3. If you are unable to unmount/remount the share, reboot the box; this should recover NFS share. If you do reboot, verify that the /ELK-nfs share is mounted when the box comes back online. Elasticsearch should start automatically on the reboot, but you should verify. After that, ensure the cluster returns to 100%.

UPDATE: This issue has not been seen in G7 since the Isilon was upgraded to OneFS 9.1.0.10.

4. Elasticsearch not starting on reboot.

We have seen issues where Elasticsearch does not start after a reboot due to invalid permissions on the file **/usr/lib/tmpfiles.d/elasticsearch.conf**. This seems to only happen if the node is rebooted without stopping Elasticsearch first. If you log in to a node and see that Elasticsearch was not started after a reboot this could be the issue.

The solution is to delete the /**usr/lib/tmpfiles.d/elasticsearch.conf** file. Elastic should start on reboot after the file is removed.

NOTE: Elastic will also start normally after logging in if **systemctl start elasticsearch** is run after logging into the problem VM.

The two log locations where elastic info could be are:

- /var/log/messages - This will contain the logs of most all applications running on the box.
- /var/log/elasticsearch/ - This is a folder path where several different logs pertaining to Elasticsearch reside; check these for relevant info. The log to look at first is <Cluster_Name>.log.

5.2.3 Check Elastic Cluster Health from Linux Command Line

When Kibana is not able to talk to Elastic or you are having issues getting access to the web page you may have to use the Linux command line to determine if the cluster is up and running. Use the following commands to help determine the status of the cluster. When looking at the commands notice the similarity. Any command that can be executed from the Kibana console can also be executed using curl.

5.2.3.1 Display Cluster Health

```
# curl -k -u {username} https://elastic-node-1:9200/_cluster/health?
```

5.2.3.2 List Nodes of Cluster

```
# curl -k -u {username} https://elastic-node-1:9200/_cat/nodes?pretty
```

5.2.3.3 List Indexes in Cluster

```
# curl -k -u {username} https://elastic-node-1:9200/_cat/indices?pretty
```

5.2.4 Unassigned Shards

If the cluster is yellow, it's likely that they're unassigned shards. This happens from time to time and is not a serious issue.

Running commands from the Kibana **Dev Tools** tab is the easiest place to resolve unassigned shards. It can also be done by running the same commands from the Linux command line using curl. We'll use Kibana in the following examples.

First let's examine the output of the cluster health. Run the following command to get the health:

```
GET _cluster/health
```

This cluster health command will show basic status of the cluster, including the number of nodes, shards, whether there are unallocated shards, etc. An example output showing unassigned shards is shown here:

```
{
  "cluster_name" : "ECH-Cluster",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 518,
  "active_shards" : 1037,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 5,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 100.0
}
```

You can see all the shards and determine which are unassigned with the following. (Adding the "?v" on the end of the command turns on verbose output, which will show the column headers.)

```
GET _cat/shards?v
```

The list will show all shards and states. The following is a list of possible shard states:

- Started – The shard is working normally.

- Relocating – The shard is being moved from one node to another.
- Initializing – The shard in getting ready, on its way to started.
  NOTE: This fails sometimes, which results in an unassigned state.
- Unassigned – The shard is not assigned to a node and is currently unusable. The reasons for an unassigned shard are:
  - ALLOCATION_FAILED: Unassigned as a result of a failed allocation of the shard.
  - CLUSTER_RECOVERED: Unassigned as a result of a full cluster recovery.
  - DANGLING_INDEX_IMPORTED: Unassigned as a result of importing a dangling index.
  - EXISTING_INDEX_RESTORED: Unassigned as a result of restoring into a closed index.
  - INDEX_CREATED: Unassigned as a result of an API creation of an index.
  - INDEX_REOPENED: Unassigned as a result of opening a closed index.
  - NEW_INDEX_RESTORED: Unassigned as a result of restoring into a new index.
  - NODE_LEFT: Unassigned as a result of the node hosting it leaving the cluster.
  - REALLOCATED_REPLICA: A better replica location is identified and causes the existing replica allocation to be canceled.
  - REINITIALIZED: When a shard moves from started back to initializing.
  - REPLICA_ADDED: Unassigned as a result of explicit addition of a replica.
  - REROUTE_CANCELED: Unassigned as a result of explicit cancel reroute command.

To see the reason, execute the following command:

```
GET _cat/shards?v&h=index,shard,prirep,state,unassigned.reason
```

Find the unallocated shards and manually reallocate them, but before you start allocating shards, turn off cluster allocation to stop any automatic rebalancing that might occur. Run the following commands.

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": "none"
  }
}
```

Now you can run the following to start manually allocating unassigned shards. You must fill in the proper data that matches your problematic shards. Note that both index and node require **“”** to show they are strings, but shard does not need them as it is looking for an integer.

```
POST _cluster/reroute?retry_failed=true
{
  "commands": [{
    "allocate_replica": {
        “index”: “{index_name}”,
        “shard”: {shard_#},
        “node”: “{Elastic_node_allocating_to}”
     }
```

```
    }]
}
```

For example:

```
POST _cluster/reroute?retry_failed=true
{
  "commands": [{
    "allocate_replica": {
        "index": "metricbeat-8.9.1-2023.12.25",
        "shard": 2,
        "node": "elastic-node-3"
     }
  }]
}
```

After running the command, if you execute a _cat/shards command, you should see the state of the shard set to initializing. You can also watch the progress of the initialization in the Shard Activity area on the **Overview** page of the **Monitoring** tab. Note that you will also see the number of shards initializing in the _cluster/health response.

Allocate the unassigned shards one at a time until the cluster health is returned to 100%.

When finished allocating unassigned shards you must re-enable shard allocation by running the following command (**DO NOT FORGET TO DO THIS**). If this is not done, the cluster will soon be back to yellow.

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": null
  }
}
```

5.2.5 ILM Issues

Index Lifecycle Management ensures that indexes are moved through storage stages in the cluster and eventually deleted. If an index is having ILM errors, then it may get stuck and possibly never deleted. It is important that index lifecycle policies are monitored to ensure all indexes are moving through their policies correctly. If an index is showing an ILM error then the issue must be resolved.

5.2.5.1 Attribute box_type causing ILM to fail

Elastic has been updated to use node roles instead of node attributes to determine shard allocation locations. In the process we have found issues if older indexes still have the "box_type" attribute defined. If you have indexes that are failing to move to the next step of ILM this may be the issue.
Check to see if the box_type attribute is still in the index settings:

GET <index>/_settings

If the “box_type” attribute exists on an index, then this setting should be removed.
Run the following command to remove this attribute:

```
PUT <index>/_settings
{
   “routing.allocation.require.box_type” : null
}
```

This should allow the index to move to the next ILM Phase. If you continue to have an issue with an index after removing the “box_type” attribute contact an Elastic SME for additional guidance.

5.3 Logstash Troubleshooting

All data from every site is sent to the Logstash instance for that site, where it gets tagged as that site’s data before being sent into the Elastic cluster. There are times when a Logstash instance has issues and cannot send data to Elastic. Most of the time these issues are caused by an initial problem with the cluster that, if left unattended, can eventually cause issues on the Logstash instances. For example, if the Elastic Cluster is unreachable by a Logstash instance for a long time, the Logstash instance may have a problem with too many files open and needs to be restarted. This issue can be compounded by the fact that the VM logs errors into the /var/log/messages file, which can eventually fill up the VM’s disk. A Logstash instance having a /var/log partition at 100% is one of the main issues seen.

The best way to keep from having problems with the Logstash instances is to perform the Daily Checks outlined previously in Section 2.

If you are having trouble with starting Logstash or keeping it running, the best way to determine the problem is to look in the Logstash-plain.log file located in the /var/log/logstash directory.

NOTE: The logstash logs are also ingested into Elastic and can be viewed in the filebeat index.

5.4 Elastic Data Collector Extended Details

Verifying that the elasticDataCollector is running properly is part of the Daily Checks outlined previously in Section 2. This section provides more details on the configuration, input, and output file locations for the elasticDataCollector.

The elasticDataCollector runs as a service on each of the Logstash VMs. The service is called **elasticDataCollector.service**. If the service is stopped, Puppet will restart it. Keep this in mind if you are trying to stop it for some administration reason.

The data collector is a Python application that runs in a Python virtual environment set up during installation. The virtual environment configuration and Python files are located in the /etc/logstash/scripts directory. The urllib3 and Elasticsearch Python modules are installed in the virtual environment for use by the data collector.

Other Python modules installed on the Logstash VM are:

- net-snmp-python
- python-requests

- python-virutalenv
- python-cryptography

The Elastic Data collector runs from the /etc/logstash/script directory but also uses the following directories:

- /etc/logstash/MIBS: Contains all Management Information Databases for devices that the data collector interrogates.
- /ELK-local: Location of directories used by data collector:
    - elasticDataCollector: Location of data collector configuration files
    - metrics_in: Location where Logstash writes host health received from Metricbeat agents. This data is read by the elasticDataCollector to help evaluate the health of hosts and applications.
    - metrics_out: This is the output directory for host, group, and application documents created by the elasticDataCollector. Documents generated are written to files in this directory. The Filebeat agent on Logstash monitors this directory and sends any data written there to Logstash for ingestion into Elasticsearch.
    - device_data: This is the output directory for device documents. Documents generated are written to files in this directory. The Filebeat agent on Logstash monitors this directory and sends any data written there to Logstash for ingestion into Elasticsearch.

Other files of interest in the /etc/logstash/scripts directory:

- deviceconfig.json: This is the configuration file created by the configurator. The configurator is used to setup the devices to be monitored from each Logstash VM at each site. Although this file can be edited manually, it is recommended to use the configurator application to add/modify device configurations. Instructions to do this can be found in the *ES-018 - Elastic Logging and Aggregation Cluster (ELAC) – Upgrade to 7.12.1 Instructions* document.
- querier.dat: Holds username and encrypted password for Querier objects to access Elasticsearch.
- Xtremio.last_event_querytime: Used by XtremIO object and holds the last time events were queried. The time is in ls epoch format.
- Lastupdate: Used by Watcher object and contains the file offset and file that is currently being monitored for data. This allows the Watcher to pick up where it was on a restart of the elasticDataCollector without re-processing data that it has already processed.

5.5 Metricbeat Script Processor

There is custom DCGS processing done in metricbeat in a script processor that is called for each document. The script processor is called appmonitor_linux.js on Linux VMs and app_monitor_win.js on Windows VMs.

See Script Processor | Metricbeat Reference [master] | Elastic for more information on script processors.

For more information on the appmonitor DCGS script processor, see *ES-018 - Elastic Logging and Aggregation Cluster (ELAC) – Upgrade to 7.12.1 Instructions*.

6 Node Maintenance

Occasionally a node might have to be rebooted or brought down for maintenance. It is never good to just reboot any of the nodes that are part of the Elastic cluster. Use the following procedures prior to doing anything with an Elastic node. If you are unsure of how to proceed, contact the Elastic SME at the DSIL for guidance.

6.1.1 Rebooting an Elastic Node

If you want to reboot a node for any reason, you should take steps to prepare Elastic first. To prevent a major impact on the cluster, only one node should be rebooted at a time. When you shut down a node, the allocation process waits for `index.unassigned.node_left.delayed_timeout` (by default, one minute) before starting to replicate the shards on that node to other nodes in the cluster, which can involve a lot of I/O. Since the node is shortly going to be restarted, this I/O is unnecessary. You can avoid racing the clock by disabling allocation before shutting down the node. Run the following commands from the Kibana console to disable allocation before rebooting:

1. Disable shard allocation:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": "none"
  }
}
```

2. Execute the following command until there are no failures:

```
POST _flush
```

3. Stop Elastic on the node and reboot.

   Shut down the node that you want to reboot by logging into the host and executing the following Linux command:

```
# systemctl stop elasticsearch
```

   Now you can reboot the host by executing the following command or using Vcenter:

```
# reboot
```

4. Verify the rebooted node rejoins the cluster.

   After the host is rebooted, verify that the elastic node rejoins the cluster by executing the following command from the Kibana console and looking for the node in the list.

```
GET _cat/nodes
```

   NOTE: If the node never shows up, log in to the node and verify that elasticsearch restarted properly. If elasticsearch is not running, see 5.2.2 Missing Elastic Node for recovery.

5. Re-enable shard allocation:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": null
  }
}
```

6.1.2 Shut Down Entire Cluster

Shutting down the entire cluster should be a last resort but there are times when this may need to be done. For example, if the FX Chassis that hosts the Elastic VMs is being brought down for any reason, the Elastic Cluster would have to be shut down. Even though the cluster will most likely survive an unorderly shutdown, it is much better if we bring things down gracefully so there is much less time spent bringing the cluster back online.

1. Stop all Logstash ingest into the cluster.
   Since the cluster will be shut down, it's a good idea to stop as much ingest traffic as possible. An easy way to do this is to stop the Logstash instances that are sending data. Note that if the cluster is not going to be down long you may not want to do this. Logstash will queue data when it cannot send it to Elastic but it's not a good idea to leave Logstash running for more than a couple of hours when it doesn't have an endpoint to send its data to.
   Log in to every Logstash node and execute the following Linux command:
   `# systemctl stop Logstash`
2. Disable shard allocation. Run the following command from the Kibana console:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": "none"
  }
}
```

3. Execute the following command from the Kibana console until there are no failures:
   `POST _flush`
4. Log in to each Elastic node and execute the following Linux command:

```
# systemctl stop elasticsearch
```

5. After all Elastic nodes have been stopped you can shut down the host VMs.

NOTE: When turning the Elastic VMs back on be sure to verify that all nodes join the cluster and then perform normal cluster health checks. After the cluster is healthy (green), turn the Logstash ingest nodes back on.

Log in to every Logstash node and execute the following Linux command:

```
# systemctl start logstash
```

6.1.3 Adding OS Patches on an Elastic Node

DCGS releases patches regularly for RedHat which is the base operating system on the VM where each Elasticsearch node runs. Some patches may call for a reboot of the VM; if that is required verify the cluster is "Green" as described in Section 2.3.1 Cluster Health and then restart the node that has been patched following the steps in section 6.1.1 Rebooting an Elastic Node.

NOTE: Elastic nodes should be restarted one at a time. Verify the rebooted node returns to the cluster and that the cluster returns to "Green" after each node reboot

6.1.4 Decommission an Elastic Node

If you need to remove a node from the cluster for an extended time (or forever), you will want to move all the shards that are on the node to other nodes before shutting it down. Execute the following command, replacing the "x.x.x.x" with the IP address of the node you are removing from the cluster. After executing the command, you must give the cluster time to relocate the shards that are currently located on the node you are removing. Note that the shard relocation can take some time so be patient.

NOTE: There are also _name and _host parameters for this command but we have found the _ip option to work the best.

```
PUT _cluster/settings
{
  "transient" : {
    "cluster.routing.allocation.exclude._ip" : "x.x.x.x"
  }
}
```

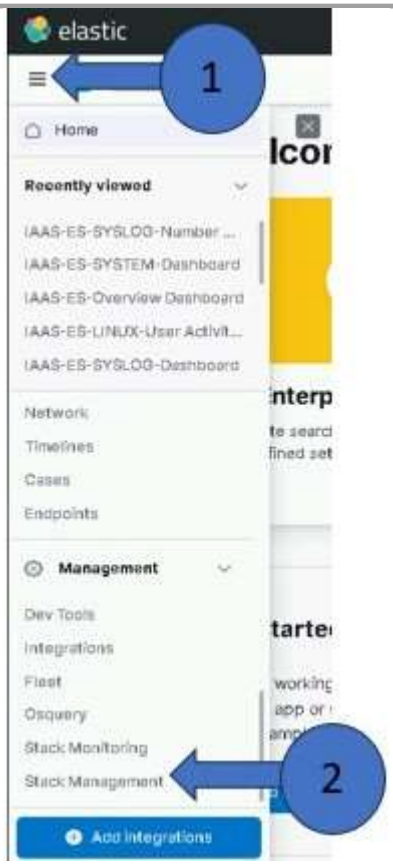
You can monitor the shards allocated to the node by selecting the node from the **Nodes** screen of the **Stack Management** page in Kibana. Once all shards have been relocated from the node it is safe to shut the node down. The rest of the nodes will automatically detect that it is no longer part of the cluster.

6.1.5 Stop ILM for Old Indexes

As Elastic is upgraded, the indexes for the beats-collected data change because the indexes have the beats version numbers. These older indexes are setup for ILM and will continue to rollover and create new indexes even after all beat collectors have been upgraded. This will eventually result in multiple indexes that have no data in Elastic. To stop these indexes from being created, the only old index that needs to be deleted is one with is_write_index set to true.

6.1.5.1 Determine if Any Indexes are Candidates

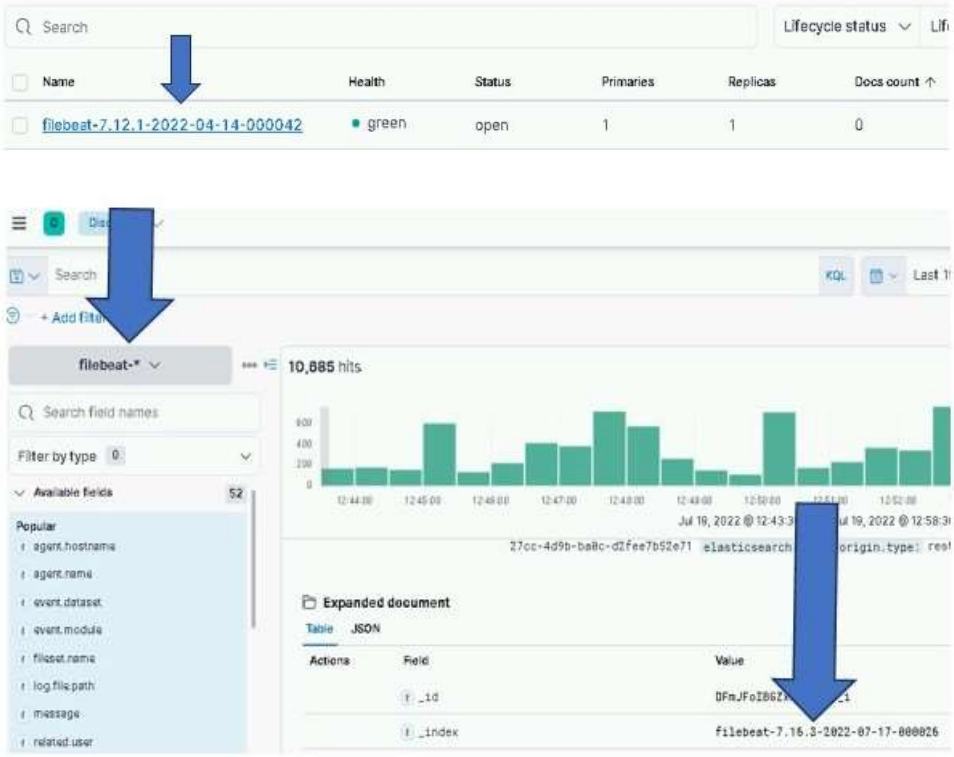
1. Navigate to the Stack Management page



2. Click Index Management

3. Sort by Docs count

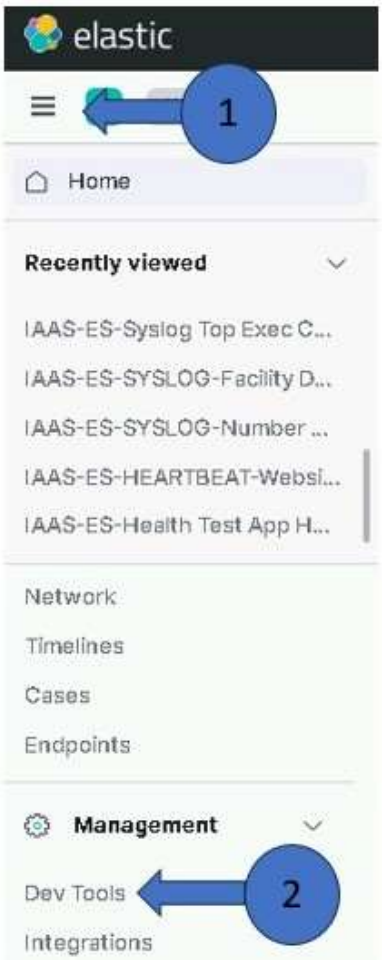
4. Correspond the versioning of the empty indexes with the current version of the index through the Discover tab



5. Verify that the empty indexes are an older version than what is currently being used (7.12 versus 7.16 for example). If the empty indexes are an older version, then we are clear to remove the write_index. If the versioning is the same between the indexes with 0 documents and the current version found in the discover tab, **STOP AND CONTACT AN ELASTIC SME.**

6.1.5.2 Delete Old Write Index

Once verified the empty indexes are old, navigate to the Dev Tools console

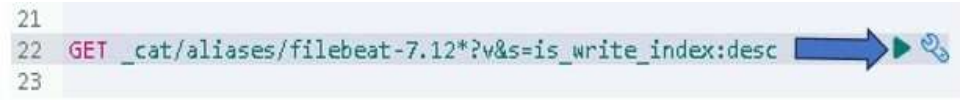


Type:

`GET _cat/aliases/nameofindexwithversion*?v&s=is_write_index:desc`

Replace the *nameofindexversion* with the beat and version that has 0 documents. For example, filebeat-7.12*

Press the green Play button to execute the command



The right side of the Dev Tools shows the output of the command. The index with “is_write_index=true” is the ONLY one that needs to be deleted. Copy the index name and then run a DELETE command on it

```
alias           index                            filter routing.index routing.search is_write_index
filebeat-7.12.1 filebeat-7.12.1-2022-07-14-000055 -      -            -              true
filebeat-7.12.1 filebeat-7.12.1-2022-04-21-000043 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-06-02-000049 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-04-28-000044 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-06-23-000052 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-04-14-000042 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-05-26-000048 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-05-12-000046 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-06-16-000051 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-05-05-000045 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-05-19-000047 -      -            -              false
filebeat-7.12.1 filebeat-7.12.1-2022-06-09-000050 -      -            -              false
```

```
DELETE filebeat-7.12.1-2022-07-14-000055
```

This is the expected output:

```
{
  "acknowledged" : true
}
```

This will stop the creation of the 7.12.1 empty indexes. The existing indexes will continue their path through ILM and eventually be automatically removed from Elastic.

7 Useful Commands from Kibana Console

Below are some commands that can be useful from the Kibana Console. Note that all these commands can also be executed from the Linux command line using curl.

Table 11 Kibana Console Commands

Command	Description
`GET _cluster/health`	Show overall health of cluster.
`GET _cat/nodes`	List nodes of cluster.
`GET _cat/indices`	List indexes.
`GET _cat/indices/metric*`	List indexes beginning with "metric" (can be any string)
`GET _cat/shards/{index name}`	List all shards or if {index_name} is specified shards of that index.
`GET _cluster/settings`	Show cluster settings.
`GET _xpack/license`	Show license information.
`GET _xpack/security/role/{role_name}`	Show security role information for all or specified name.
`GET _cluster/stats`	Show cluster wide statistics.
`GET _cluster/allocation/explain` **{** **"index" : "{index name}",** **"shard" : 0,** **"primary" : false** **}**	Provide explanations of shard allocations in the cluster. {index name} - Name of index you would like an explanation for. shard – id of shard you would like an explanation for
`POST <index name>/_rollover`	Creates new index for <index name> using ILM (Should only be used for testing purposes)

NOTES:

- Adding "?v" to most commands will give column headers in the output.
  - Example: `GET _cat/nodes?v`
- Many commands support Help by adding an ?help on the end.
  - Example: `GET _cat/nodes?help`
- You can specify what columns (data) will be returned from the command by specifying headers. See the "?help" for the command to see the available columns.
  - Example: `GET _cat/nodes?v&h=ip,port,heapPercent,name`
- You can sort the output by column by adding an &s=<column name> on the end of the command. Note that multiple columns can be specified. You can also change the sort order to descending by specifying: desc as part of the command
  - Example: `GET _cat/indices?v&s=index`
  - Example: `GET _cat/indices?v&s=index:desc`

See Compact and aligned text (CAT) APIs | Elasticsearch Guide [master] | Elastic for more information.

7.1 DOD PKI Certificates

All DOD PKI certificates must be obtained and copied to the **install/certs** directory in the Elastic repository. The certs directory is under the install directory **install/certs**. This folder should contain the certificates for all Elastic and Logstash VMs that are being installed. (Created in 5.1)

The **elastic_cachain.pem** file containing the Root CA and Sub CA used to issue all Elastic certificates exists in the **certs** directory in the Elastic repository. This file must be in place for successful execution of the Elastic installation scripts.

PKI Certificates must be in the following format for the installation scripts to work properly:

Public Cert: {hostname}.crt	examples: u00su01el01.crt, u00su01ls01.crt
Private Keys: {hostname}.key	examples: u00su01el01.key, u00su01ls01.key

7.2 License Expiration

- # cat elasticLicense.json
  1. Select the "expiry_date_in_millis" value from the data printing out. This will be a long number representing time in milliseconds.
  2. Use the number obtained in the following command:

date –d @$((<number obtained>/1000))

Example: date –d @$((1622332799999/1000))

Sat May 29 23:59:59 UTC 2021

The example shows a license that expires on May 29th 2021

7.3 Accounts and Passwords

- xx_elastic.svc service accounts exist for each site and have been given the correct privileges.
- Service account should be a member of the following groups:
  - fs – xx app-elac full control (xx = site number)
  - domain users
  - ent infrastructure read only
  - dcgs service accounts
- You have the password for the **xx_elastic.svc** account
- You have the password for the Elastic bootstrap account **elastic**

7.3.1 Elastic Certificates (includes Kibana)

Certificates are needed for each Elasticsearch node. Elastic Certificates contain the following:

CN: hostname of Elastic VM (ex: u00su01el01.ech.dcgs.mil)

Aliases:

- fully qualified hostname (ex: u00su01el01.ech.dcgs.mil)
- hostname (ex: u00su01el01)
- hostname.{first segment of domain} (ex: u00su01el01.ech)
- elastic-node-{x} (ex: elastic-node-1)
- elastic-node-{x}.{first segment of domain} (ex: elastic-node-1.ech)

Additional Aliases if Kibana runs on the VM:

- kibana
- kibana.{first segment of domain} (ex: kibana.ech)
- kibana.{fully qualified} (ex: kibana.ech.dcgs.mil)

A convenience script is provided to make the creation of the Elastic Server Certificate requests easy for the installer. To create PKI certificate requests for Elastic to run on the system:

1. Log in to any existing Linux server at the site where the Elastic Cluster will be installed and do the following from your home directory:

   ```
   # curl -k
   https://satrepo/pulp/content/oadcgs/Library/custom/Elastic_Client/Elast
   ic_Files/install/make_elastic_csrs.sh | bash
   ```

2. When the script completes, 3 directories will be present in the location where it was run.
   - Reqs: This directory holds the CSR Request information in text format
   - Keys: The private key associated with the certificate request for each Elastic node
   - CSRs – The actual PKI Certificate Request for each Elastic node.

3. Edit the *.csr files in the CSRs folder to:
   - Uncomment the keyUsage and extendedKeyUsage lines
   - Correct the spelling of “keyEncipherment” on the keyUsage line

4. The *.key and *.csr files should be submitted to the certificate authority for the system the Elastic Cluster is being installed on to obtain public certificates for each node. For systems submitted to the JWICS certificate authority (i.e. CTE High or Enterprise High), also include a text file named SANS.txt listing the SubjectAlternativeNames listed in the CSR (for each CSR).

NOTE: In this installation, Kibana will run on one of the Elastic nodes and use the certificate for that node.

5. Once the Elasticsearch node certificates have been obtained, both the new certs and the keys for each node must be copied to the **install/certs** directory of the Elastic repo on the repo server ({xxx}su01ro01).
6. Replace the (empty) default elastic_cachain.pem file in the install/certs directory with the cachain from the certificate authority for the system being installed.

   NOTE: There is also an empty cachain.pem file delivered with the oadcgs-es-elastic-sccm package that must be updated to hold the correct root and sub-ca certificates during installation.

PKI Certificates must be in the following format for the installation scripts to work properly:

Public Cert: {hostname}.crt	examples: u00su01el01.crt, u00su01el02.crt
Private Keys: {hostname}.key	examples: u00su01el01.key, u00su01el02.key

7.3.2 Logstash Certificates

A Logstash instance runs at each DCGS site to collect data for that site. A PKI certificate is needed for each Logstash Instance. Do this to obtain a Logstash certificate request for each site.

Logstash Certificates contain the following:

CN: hostname of Logstash VM (ex: u00su01ls01.ech.dcgs.mil)

Aliases:

- fully qualified hostname (ex: u00su01ls01.ech.dcgs.mil)
- hostname (ex: u00su01ls01)
- hostname.{first segment of domain} (ex: u00su01ls01.ech)
- logstash (ex: logstash)
- logstash.{first segment of domain} (ex: logstash.ech)

A convenience script is provided to make the creation of the Logstash Server Certificate requests easy for the installer. To create PKI certificate requests for Logstash to run on the system:

1. Log in to any existing Linux server at the site where the Logstash Server is to be installed and do the following from your home directory:

```
# curl -k
https://satrepo/pulp/content/oadcgs/Library/custom/Elastic_Client/Elastic_Files/install/make_logstash_csr.sh | bash
```

2. When the script completes, 3 directories will be present in the location it was run
   - Reqs: This directory holds the CSR Request information in text format
   - Keys: The private key associated with the certificate request for the Logstash instance

- CSRs: The actual PKI Certificate Request for the Logstash instance.

3. Edit the *.csr files in the CSRs folder to:
   - Uncomment the keyUsage and extendedKeyUsage lines
   - Correct the spelling of "keyEncipherment" on the keyUsage line

4. The *.key and *.csr files files should be submitted to the certificate authority for the system the Elastic Cluster is being installed on to obtain public certificates for each node. For systems submitted to the JWICS certificate authority (i.e., CTE High or Enterprise High), also include a text file named SANS.txt listing the SubjectAlternativeNames listed in the CSR (for each CSR).

5. Once the Logstash certificate has been obtained, both the new cert and the key for each Logstash instance must be copied to the **install/certs** directory of the Elastic repo on the repo server ({xxx}su01ro01) for the associated site.

PKI Certificates must be in the following format for the installation scripts to work properly:

Public Cert: {hostname}.crt	examples: u00su01ls01.crt
Private Keys: {hostname}.key	examples: u00su01ls01.keyRoot Certificates

The **elastic_cachain.pem** file containing the Root Certificate Authority (CA) and Sub CA used to issue all Elastic certificates must exist in the certs directory in the Elastic repository. If this file is not present in the **elastic/certs** directory on the repo server, you must create it:

1. Log in to the repo server in your environment ({xxx}su01ro01 box in your environment) and become root:

   ```
   # cd /etc/pki/ca-trust/source/anchors
   ```

2. You should see the following 2 files in this directory:
   - rootca.pem
   - subca.pem

   If they are not there, stop and ask for guidance from a Linux Admin or Elasticsearch SME.

   It is possible that the cachain.pem is already there. In any case, ensure that the source(s) signing all of the certs (there may be different sources for different certs obtained from a single request) are present in the cachain.

3. Create the elastic_cachain.pem file:
   ```
   # cat rootca.pem subca.pem > /var/www/html/yum/elastic/install/certs
   /elastic_cachain.pem
   ```

4. Copy the elastic_cachain.pem file into the extracted SCCM install; e.g., on the fileserver in **<install>\oadcgs-es-elastic-sccm-1.0.0.3\oadcgs-es-elastic-sccm\sccm\shareDir** replacing the dummy cachain.pem

7.4 Exporting Beat Templates for Installation

NOTE: These templates should already be present in the installation package. These instructions are provided if a template is missing or needs to be recreated.

To make installation easier the templates needed for each of the Elastic Beats components are extracted and placed in the **install/templates** folder as part of the installation package. If for some reason these templates are missing or if you ever need to extract them yourself follow these instructions.

Each Beat is distributed with a template, but it is inaccessible unless you install the Beat. Since the installation procedures in this document install the templates from a Linux host it is best to extract the templates by installing the Beat on a Linux host. This is the method for all Beats except Winlogbeat, which only runs on Windows boxes.

Install the Beat on the host and run the following command:

```
# {beatname} export template > est_{beatname}-{version}
```

Example for Metricbeat version 8.9.1:

```
# metricbeat export template > est_metricbeat-8.9.1
```

The same process is used to export the template for Winlogbeat from a windows box, but once exported the file must be converted from Windows format to ASCII text. Once you have created the **est_winlogbeat-x.x.x** template file, copy it to a Linux machine and run the following command:

```
# dos2unix est_winlogbeat-8.11.3
```

NOTE: If dos2unix is not available, it can be installed by executing:
```
# yum install dos2unix
```

7.5 Configure NSX Load Balancer

NOTE: The Network Administrator role on the NSX manager is required to execute this section.

Kibana, the web interface to Elastic, can be accessed by navigating to https://kibana on the DCGS system. This URL directs the user to a load balancer that will forward the requests to the appropriate Kibana instance. Depending on the configuration there can be one or more Kibana instances available to handle

user requests. Before proceeding with the installation ensure that the NSX Load Balancer is configured to handle user requests.

To configure NSX for Elasticsearch, refer to *ES-018 - VMware - NSX-V Load Balancer Deployment Guide.* The load balanced name for each service must be created in DNS; if this was not completed during the deployment of the load balancer, execute the following steps.

Create a **DNS A** record for the Kibana load balancer address that points to the virtual IP for the Kibana service.

Example: FQDN for target host: **kibana.ech.dcgs.mil**

Perform testing and validation of the Kibana DNS A record for the ElasticSearch Kibana load balancer portal FQDN and IP address.

Open a command window and run the following commands:

```
ping kibana.ech.dcgs.mil
```

```
nslookup kibana.ech.dcgs.mil
```

7.6 Service Account Kerberos Management (SAKM)

During installation, Elasticsearch is modified to be run by the Elastic service account. Elastic relies on the Kerberos ticket for the Elastic service account to be automatically updated by the SAKM scripts. Before deploying Elastic or Logstash on any VMs, ensure that SAKM has been installed. If it hasn't been installed, you can utilize the following steps to configure it. The installed SAKM must be at least **refreshticket-1.0.5-1.1.noarch**.

NOTE: To check the installed SAKM version run the following command on the system to check:

```
# yum info refreshticket
```
Verify SAKM Installed

One way to verify SAKM is installed and configured correctly for the Elastic service account is to manually request a new Kerberos ticket using the **refresh_tgt_tickets.sh** script. If this works successfully, SAKM has been configured on the VM. If not, see the following steps to configure SAKM properly for the Elastic service account.

Run the following command on the Elastic VM:

```
# /usr/local/sbin/refresh_tgt_tickets.sh -p XX_elastic.svc -r <REALM>
```

NOTE: Your Realm can be obtained by executing the following command.

```
# realm list –name-only
```

If **success** is returned, then SAKM is configured on the VM.

If access is denied, ensure the permission on the file /usr/local/sbin/refresh_tgt_tickets.sh are set to read and execute for all users (i.e. 755). Incorrect permission may be caused by an improper SAKM version; again ensure you have at least refreshticket-1.0.5-1.1.noarch installed.

7.7 Setup Repo with Core RPMs

NOTE: A Linux administrator will be needed to execute this section.

Before running the installation, the RPMs for the core Elastic components must be copied to the Elastic repo on the DCGS repo server (ex: u00su01ro0) and the repo must be recreated. Copy these components from wherever you decided to stage the Elastic-Elastic Core Components in section 4.2 step 5.

Copy the following RPMs to the Elastic repo (/var/www/html/yum/elastic):

elasticsearch-X.X.X-x86_64.rpm

kibana-X.X.X-x86_64.rpm

logstash-X.X.X-x86_64.rpm

Ensure RPMs have correct owner/group:

```
# chown -R apache:apache *
```

Repo files must have selinux context **httpd_sys_content_t** set. If you copy the RPMs into the directory, they will automatically get this context set. If you move them, they won't. Ensure all files have the correct context set by executing:

```
# ls -lZ
```

Figure 54 Set httpd_sys_content_t

If all files do not have **httpd_sys_context_t** set, execute the following:

```
# restorecon  *
```

Recreate the Elastic repo so it's ready for use:

```
# createrepo ./
# gpg --detach-sign --armor ./repodata/repomd.xml
```

NOTE: There are 2 dashes in front of **detach-sign** and **armor** in the above command.

Confirm overwriting the file (if it already exists)

Enter **y** to overwrite

7.7.1 Verify Service Account (From Each VM)

WARNING: Elasticsearch is modified to run as the **xx_elastic.svc** account during installation. Elastic must run as the service account or it will not be able to write data over the NFS to the Isilon. See prerequisites in the Software/System Inventory/Prerequisites Section for details.

Log in to each Elastic node and sudo to root.

su to the Elastic service account for this site (*{sitecode}*_elastic.svc):

```
# su 00_elastic.svc
```

Verify the service account can access the Isilon share on this node.

```
# cd /ELK-nfs
# ls -la
```

Verify the location is not empty; it should have at a minimum "." and ".."

7.7.2 Elasticsearch Install – Adding a Node

This step will be done on each VM that will become an Elasticsearch node.

Log in to the VM and sudo to root

Verify iptables are set up correctly. This should be handled by Puppet. All Elastic nodes should be included in the "Elastic Servers" Classification on the Puppet server.

`# iptables --list -n` (there are 2 dashes in front of list)

Verify ports 9200 and 9300 are open. You should see a line for *`multiport dports 9200, 9300 /* 106 allow input from other Elastic nodes and clients */`*.

NOTE: If these ports are not open stop and verify hosts are set up correctly in Puppet.

To determine if data is stored locally on this node, check the table in Section 2.1.1 for the cluster size to be installed. If it will be, you must verify that there is an **ELK-local** directory already created for the data.

Is this a **Master** node? The ELK-local directory for a master node is located at root level on the OS disk. If the directory does not exist, create it.

```
# mkdir /ELK-local
# chown XX_elastic.svc /ELK-local
```

Elastic nodes that store "hot" data or machine learning nodes usually have a 2nd local disk for storage. The ELK-local directory should be the mount to the volume group allocated from that 2nd drive for this type of node.

The files system should be set up to mount the **ELK-local** drive automatically in the **/etc/fstab** for this node. Check the **/etc/fstab** file and verify a line similar to the following exists:

```
/dev/mapper/elk_vg-elk /ELK-local  xfs defaults 0 0
```

Doing a "df -h" command on the ELK-local file system should show something like this:

```
[root@u00su01el05 /]# df -h /ELK-local
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/elk_vg-elk  1.5T  258G  1.3T  18% /ELK-local
```

Figure 55 df command

If this mount point does not exist, please consult with a Linux administrator or the OA DCGS Elastic SME on how to create it before proceeding.

NOTE: All nodes should have an **ELK-nfs** mount point to the Elastic share on the Isilon even if they store data locally.

7.7.2.1 Verify VM can see Elastic Repo

Identify the Elastic Repo by listing all the repos. It should now contain 3 items.

```
# yum repolist all
```

List the contents of the Elastic repo to verify that it has the Elasticsearch RPM using the following command.

```
yum repo-pkgs {elastic repo name} list
```

example: `# yum repo-pkgs elastic-search-rpms list`

NOTE: If you do not see Elasticsearch in the repository **do not proceed**. You can attempt a **yum clean all** and try the previous steps again. If you still don't see the Elastic repo, you may need to verify it is set up properly.

7.7.2.2 Verify SSL Settings for ElasticSearch

The installation script executed previously should have populated the **/etc/elasticsearch/certs** directory with the PKI certificates for this node, along with the root certificate authority's public certificate. Verify these certs have been installed correctly.

Change the certificate directory for the node:

```
# cd /etc/elasticsearch/certs
```

Verify certificates are present.

NOTE: If the certificates are not present, stop. This must be resolved before the Elasticsearch service can run.

Verify that /etc/hostname and the hostname command use the same name as the certificate files (i.e server name without domain). If there is a mismatch, hostname and /etc/hostname should be changed to match. (Alternately, if there is a need, links to the certificate files with the used name can be created in the same location.)

Repeat these steps on all nodes before continuing. Test Elasticsearch

```
curl -k -u elastic:elastic https://elastic-node-
{x}:9200/_cluster/health?pretty
```

NOTE: This is a good test to make sure you set the Elastic user password to **elastic** properly.

NOTE: If the above queries do not work, **STOP** and contact an OADCGS SME for guidance.

7.7.2.3 Add License for Elasticsearch

If the cluster is running and is 100% healthy, add the license key by executing the following.

NOTE: This only needs to be done on the initial cluster install or whenever a license update is required.

```
# curl -k https://xxxsu01ro01/yum/elastic/install/updateLicense.sh | bash
```

7.7.3 Kibana

Kibana is the web interface used to visualize data in Elasticsearch. Kibana currently runs in conjunction with Elasticsearch on an Elastic VM. There may be one or multiple instances of Kibana running to support user access to the Elastic cluster. All Kibana instances are accessible via https://kibana from any site. An NSX load balancer is set up to route the traffic to the Kibana to allow consistent navigation for the users.

NOTE: It is possible to still access Kibana instances, directly bypassing the NSX Load balancer, by using the hostname they are running on in the URL. For example, if Kibana is running on u00su01el03 then it can be accessed directly at https://u00su01el03:5601. This direct access should only be used during this procedure for installation checks. It can also be used for debugging issues or maintenance but should not be used by general users.

Use this table to determine which Elastic nodes to install Kibana on.

Table 6 Elastic nodes to install Kibana on

# of Nodes in Cluster	Elastic Nodes where Kibana is installed
7	Node 5 and Node 6
10	Nodes 7 and Node 10
15	Node 10 and Node 15

NOTE: You must be root to install Kibana.

7.7.3.1 Test Kibana (For Each Kibana Node, If Applicable)

This test will access this instance of Kibana by explicitly specifying the name of the VM where it is installed in the URL. General user access to this instance should be controlled by the NSX load balancer using the https://kibana URL once it's configured.

Verify iptables are set up correctly. This should be handled by Puppet. Kibana normally runs on an Elastic node. That node (or nodes if multiple instances) should be included in the **Elastic Servers** Classification on the Puppet server.

```
# iptables --list -n
```

Verify port **5601** is open. If this port is not open, stop and verify hosts are set up correctly in Puppet.

Test Kibana in Firefox from any computer that has network access to the Kibana node.

```
# firefox https://elastic-node-{x}:5601  (example: firefox https://elastic-node-7:5601)
```

If it loads to a Kibana login window, success!

You can now log in to Kibana.

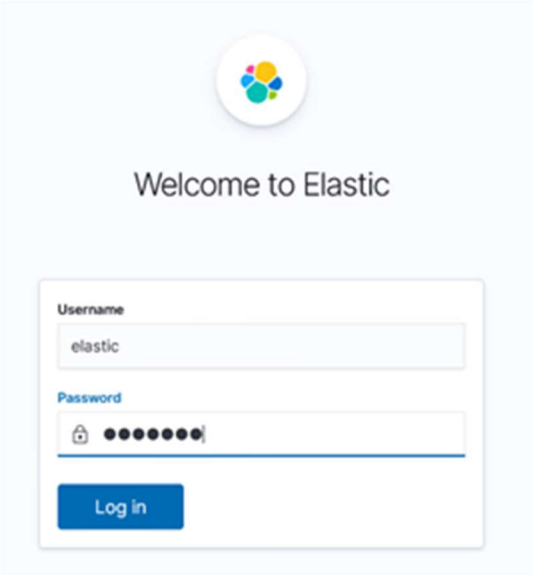


Figure 56 Login Screen

On your first login to Kibana, the **Welcome to Elastic** screen displays. Select **Explore on my own** to proceed.

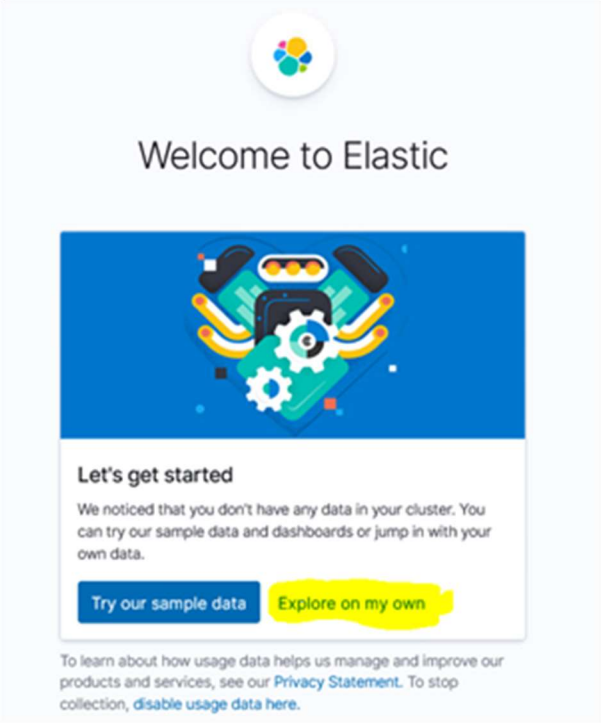


Figure 57 Explore on my own

7.7.3.2 Disable Telemetry (On One Kibana Node)

By default, Usage Collection (also known as Telemetry) is enabled. This must be disabled for DCGS.

Navigate to Kibana in your browser: https://elastic-node-{x}: 5601

Log in to Kibana.

Click the menu (three horizontal lines) button at the top left of the screen. The navigation menu displays.

Scroll to **Management** at the bottom and select **Stack Management**.

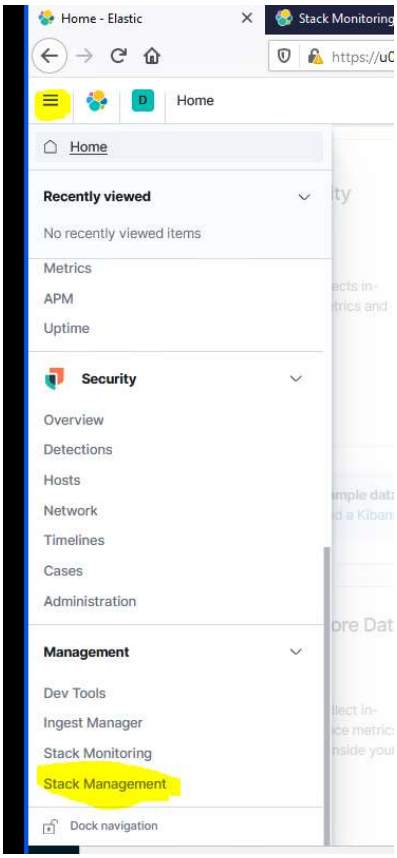


Figure 58 Stack Management

The **Stack Management** page displays. Under Kibana select **Advanced Settings** on the left side.

Enter **telemetry** in the search bar to filter.

If **telemetry:enabled** is **On**, click the slider to turn it **Off**.

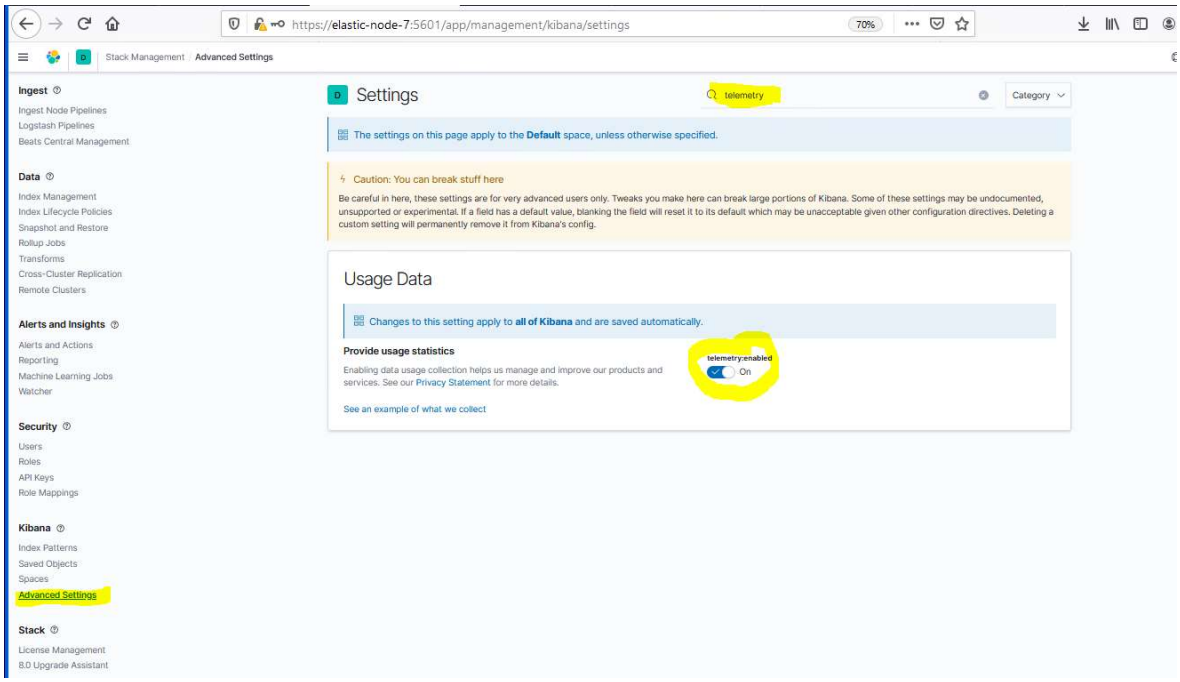


Figure 59 Disable Telemetry

7.7.3.3 Audit Settings

The logging of security-related events such as authentication failures and refused connections is enabled when installing an Elasticsearch node. The audit information will be written to the **/var/log/elasticsearch/<clustername>_audit.json** file, for example, **ECH_Cluster_audit.json**.

7.7.3.3.1 Verify Audit Settings

Verification of the audit settings and other settings in Elastic can be done from the Kibana Dev Tools console. To access the console:

Click the menu (three horizontal lines) button at the top left of the screen. The navigation menu displays.

Scroll to **Management** at the bottom and select **Dev Tools**.

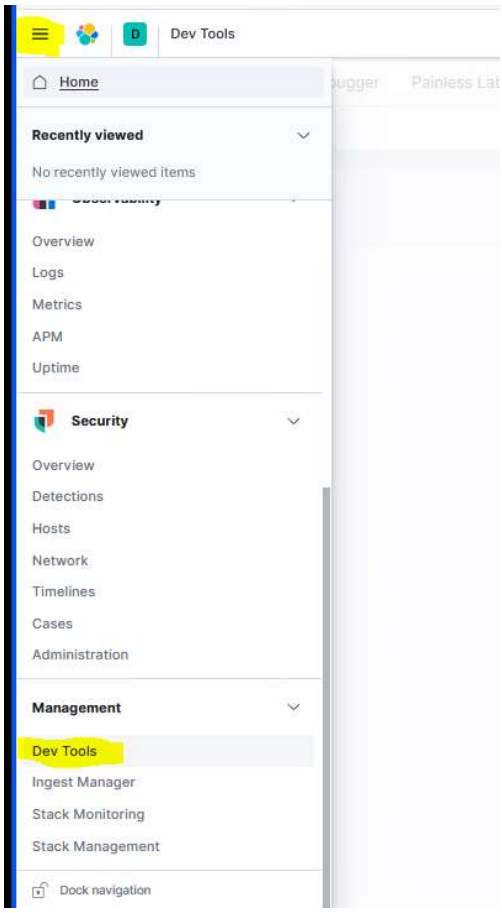


Figure 60 Dev Tools

The **Dev Tools** console displays.

To verify the settings are in place, run the following command from the Kibana Dev Tools console:

```
GET _cluster/settings
```

Verify the following audit settings are contained in the output:

```
"security" : {
        "audit" : {
          "logfile" : {
            "events" : {
              "ignore_filters" : {
                "exclude_admin_users" : {
                  "users" : [
                    "_xpack_*",
                    "logstash_internal",
                    "logstash_admin_user",
                    "logstash_system",
                    "_system",
                    "kibana-*",
```

```
            "querier-*",
          "metricbeat-user"
            ]
          }
        },
        "include" : [
          "anonymous_access_denied",
          "authentication_success",
          "authentication_failed",
          "realm_authentication_failed",
          "access_denied",
          "run_as_denied",
          "tampered_request",
          "connection_denied"
        ]
      }
    }
  }
},
```

7.7.3.4 Configure Index Lifecycle Management (ILM)

The Index Lifecycle Management (ILM) Policies are used to automatically manage the indices in Elasticsearch. In the current implementation each index has a **lifecycle** that starts as hot, moves to warm, and then is eventually deleted. This ensures Elastic does not become overwhelmed with data over time. In this version of Elastic, all indexes will be assigned to the same lifecycle policy, the **dcgs_default_policy**. As requirements change for indexes this may change in the future.

For ILM to be functional, the following must be in place

An ILM policy must be loaded into Elastic.

A template for each index must exist to assign it to the ILM policy.

Each index must be bootstrapped as the initial write index.

Verification that each index is moving through the lifecycle phases should be done.

7.7.3.4.1 Index Template for ILM

In this version, template merging (Indices matching multiple templates) is used to form the final settings containing the **dcgs_default_policy** for all indexes. the **estc_dcgs_defaults** is loaded:

```
{
 "template" : {
   "settings" : {
     "index" : {
       "lifecycle" : {
         "name" : "dcgs_default_policy"
       },
       "routing" : {
         "allocation" : {
```

```
            "include" : {
              "_tier_preference" : "data_hot"
            }
          }
        },
        "mapping" : {
          "total_fields" : {
            "limit" : "10000"
          }
        },
        "refresh_interval" : "5s",
        "number_of_shards" : "1",
        "number_of_routing_shards" : "30",
        "number_of_replicas" : "1",
        "query": {
          "default_field": [ "event.original" ]
        }
      }
    },
    "mappings" : {
      "properties" : {
        "geo" : {
          "properties" : {
            "name" : {
              "type" : "keyword"
            },
            "location" : {
              "type" : "geo_point"
            }
          }
        },
        "DCGS_Site" : {
          "type" : "keyword"
        },
        "DCGS_Site_Name" : {
          "type" : "keyword"
        }
      }
    }
  },
  "version" : 0,
  "_meta" : {
    "managed" : true,
    "description" : "default settings and mappings for all indexes
installed by DCGS Enterprise Services`"
  }
}
```

This assigns the **dcgs_default_policy** as the lifecycle policy and specifies that all ingested data will initially be writing to nodes with the **box_type** designator **hot**.

You can retrieve the contents of this template by executing the following command in the Kibana Dev Tools console:

```
GET _template/estc_dcgs_default
```

7.7.3.4.2 Verify Indexes are Moving Through Lifecycle

To ensure that ILM policies are working, the Elastic Admin must periodically check to make sure all indexes are moving through their lifecycle without issues.

NOTE: This is not something to do during the install but should be done after the cluster has been receiving data for at least a week or more.

Kibana provides an interface for this on the **Index Management** screen.

NOTE: In Kibana, Index Management is located under Stack Management.

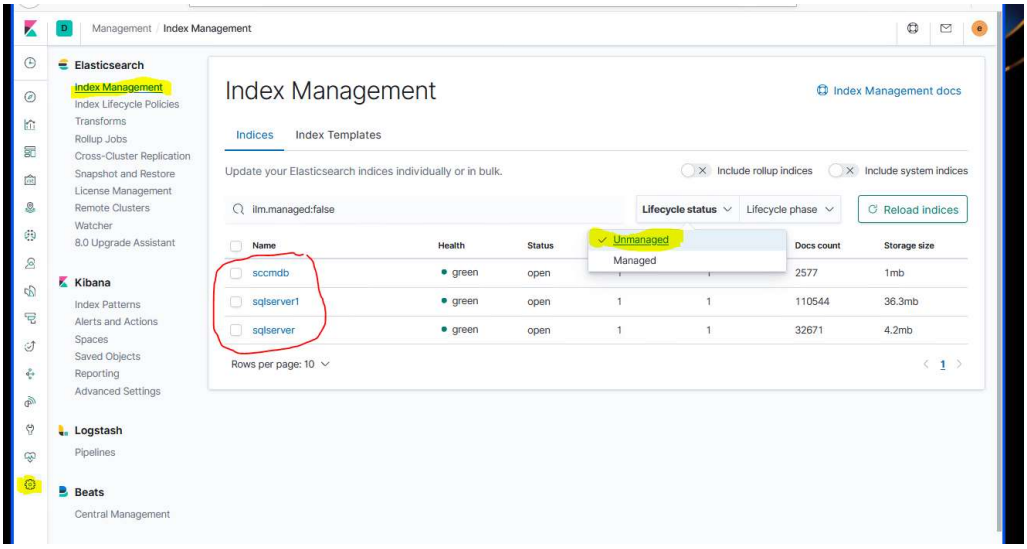


Figure 61 Index Management

In this example, **Unmanaged** has been selected under the **Lifecycle status** drop-down to show all indices that are not currently being managed by ILM. This test cluster has 3 unmanaged indexes (sscmdb, sqlserver1, and sqlserver).

There should be no unmanaged indexes on the production system. If this display shows indexes that are unmanaged, check with an OA DCGS Elastic SME to get direction on how to get all indexes managed by ILM.

Use the **Lifecycle Phase** drop-down to check the phase each index is currently in. The following example shows all indexes in the **hot** phase.

NOTE: In Kibana, Index Management is located under Stack Management.

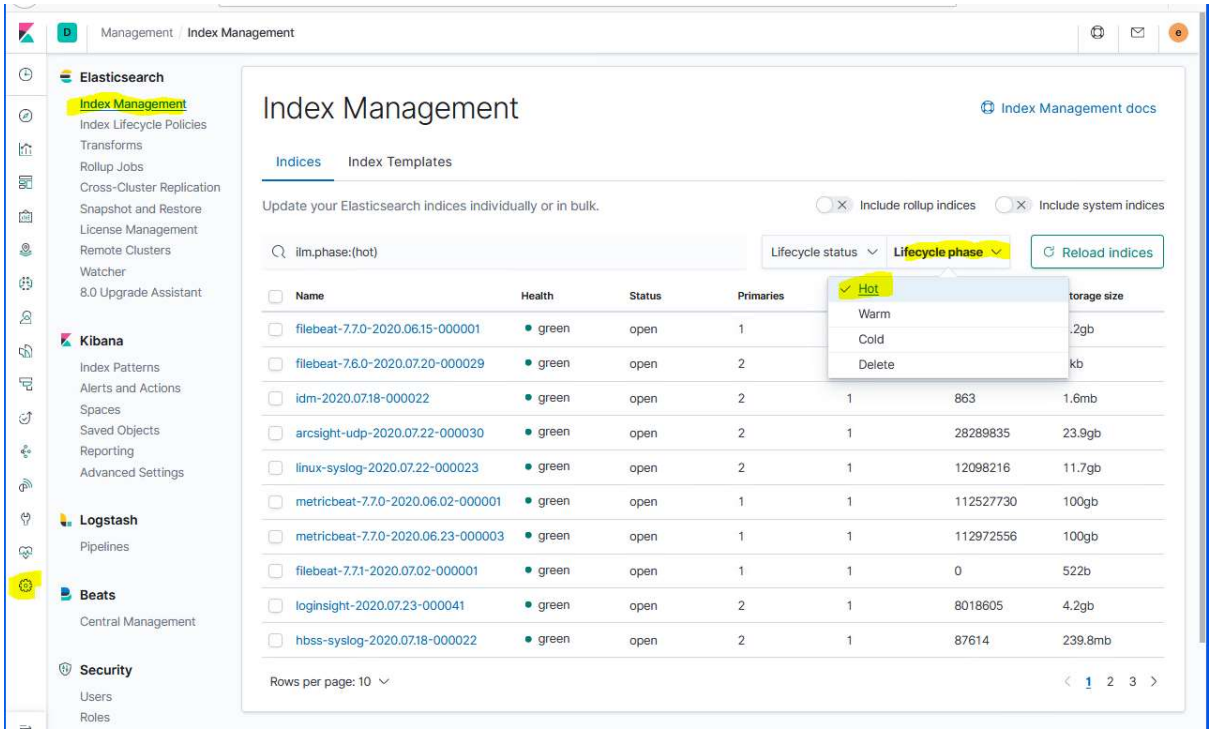


Figure 62 Lifecycle phase

7.7.3.5 Adjust Concurrent Rebalancing (optional)

Control how many concurrent shard rebalances are allowed cluster wide. The default is 2. Note that this setting only controls the number of concurrent shard relocations due to imbalances in the cluster. We have set this value to 10 on the CTE/MTE cluster, which allows faster recovery when there are unassigned shards on cluster startup. The value is easily set using the Kibana Console.

```
PUT _cluster/settings {
    “persistent” : {
       “cluster.routing.allocation.cluster_concurrent_rebalance” : 10
   }
}
```

7.7.3.6 Memory Lock Check

When the JVM does a major garbage collection it touches every page of the heap. If any of those pages are swapped out to disk they will have to be swapped back into memory. That causes lots of disk thrashing that Elasticsearch would much rather use to service requests.

There are several ways to configure a system to disallow swapping. One way is by requesting the JVM lock the heap in memory through **mlockall** (Unix) or **virtual lock** (Windows). This is done via the

Elasticsearch setting `bootstrap.memory_lock`. However, there are cases where this setting can be passed to Elasticsearch, but Elasticsearch is not able to lock the heap (e.g., if the Elasticsearch user does not have **memlock unlimited**). The memory lock check verifies that if the **bootstrap.memory_lock** setting is enabled, the JVM was successfully able to lock the heap. To pass the memory lock check, you might have to configure `bootstrap.memory_lock`.

NOTE: If swapping is not enabled on the machine, memory locking is not needed. If swapping is turned on you can check to see if Elastic was able to prevent memory from being swapped by checking the value of mlockall on each host.

```
GET _nodes?filter_path=**.mlockall
```

Remember, if this returns false, things still may be okay if swapping on the system is disabled. See https://www.elastic.co/guide/en/elasticsearch/reference/6.5/setup-configuration-memory.html for more information.

7.7.4 Logstash

Logstash ingests, enhances, and ships all data to the Elastic Cluster. There is currently one Logstash VM at every site. All data from Beats and other collectors at each site is collected by the Logstash for the site. The **DCGS_Site** field in each document received is set to the site the data is coming from and then the data is sent from that Logstash instance into the Elastic cluster.

The following users/roles are used by each Logstash instance to interact with the Elastic cluster:

Users:

logstash_system – Used for monitoring the Logstash instance
logstash_admin_user – Used to manage centralized pipelines
logstash_internal – Used by Logstash pipelines to write data into Elasticsearch
Roles:
logstash_writer – Used by Logstash to create/write indexes

The required users/roles for Logstash are automatically set up during the installation of the first Logstash instance for the cluster. The passwords for the users are dynamically generated and set in Elastic, and stored in the **logstash.keystore** on the initial Logstash instance. These passwords are never seen during creation and are used to deploy all additional instances of Logstash throughout the system. After the completion of the installation of the initial Logstash instance, the **/etc/logstash/logstash.keystore** file **MUST** be copied to the repo server and placed in the installation directory of the Elastic Repo (elastic/install). The logstash.keystore **MUST** be present for additional Logstash installations or the install will fail.

NOTE: The logstash.keystore must be placed on the repo server for each site.

NOTE: You must be root to install Logstash.

7.7.4.1 Verify Logstash

You can monitor Logstash by looking at **/var/log/logstash/logstash-plain.log**.

Use Kibana to verify this instance of Logstash is up and ready to receive data. Log in to Kibana and select **Stack Monitoring** from the **Management** section of the side navigation menu. A Logstash section showing the number of running Logstash nodes should be visible. The number of Logstash pipelines that are ready to ingest data is also visible.

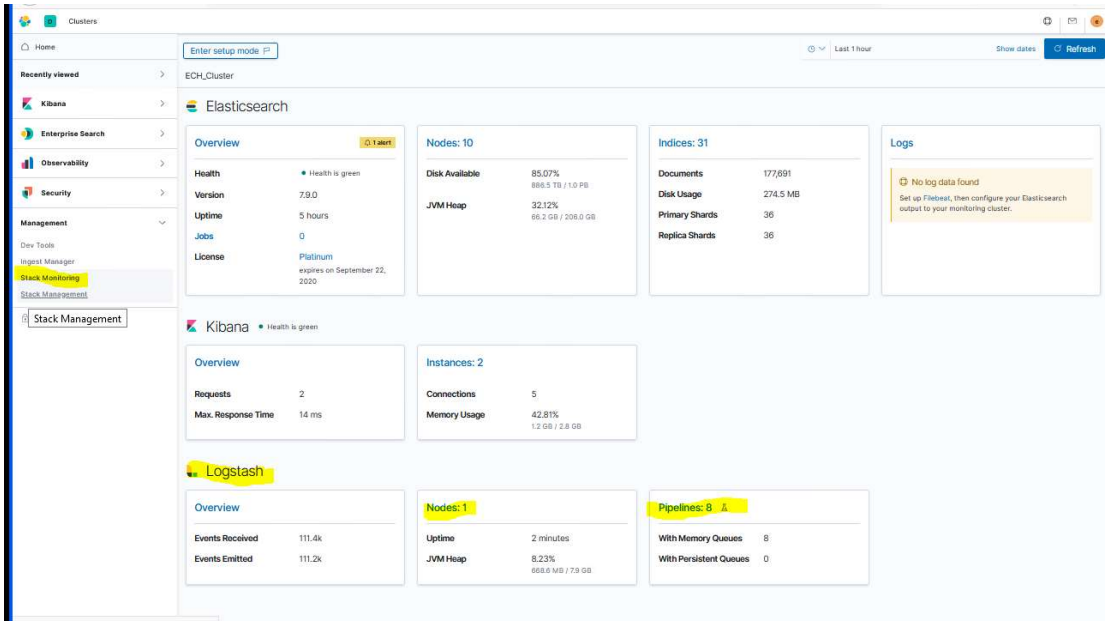


Figure 63 Logstash

Click the **Nodes** link to display the names of the Logstash instance and verify the instance you just installed is listed.

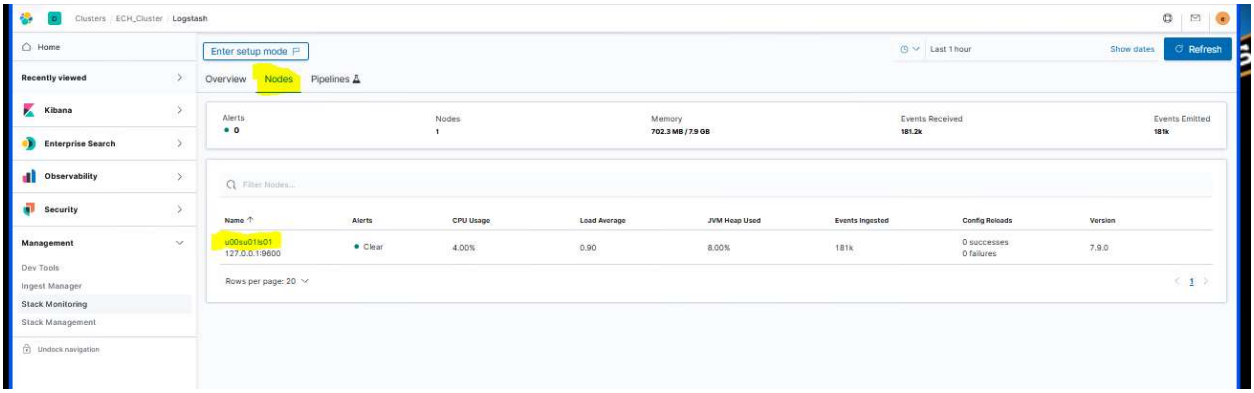


Figure 64 Nodes tab

Select the **Pipelines** option from the top to display the Logstash pipelines that are running for this instance. If the any data is being ingested, the rate of ingest will be shown.

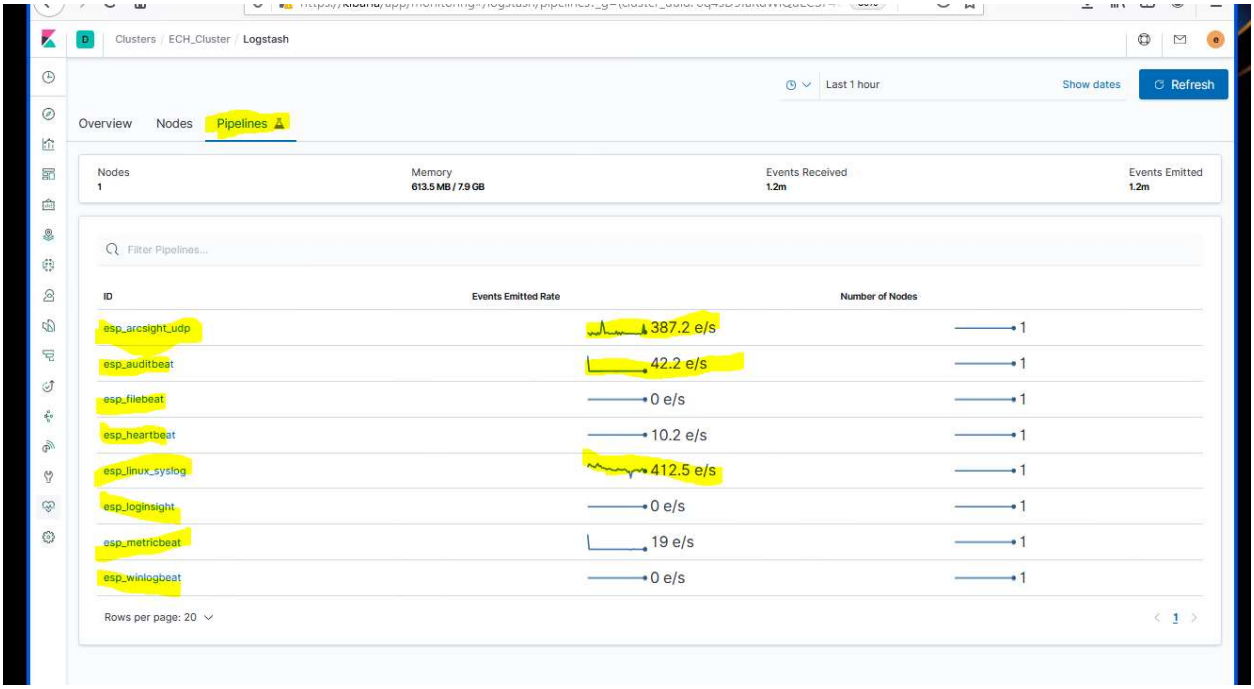


Figure 65 Pipelines tab

Repeat for each Logstash instance.

7.8 Secure Elastic with Break-Glass Password

This section is to be completed after the successful install and checkout of Elasticsearch, Kibana, and all Logstash instances on the entire system.

Now that Elasticsearch has been successfully installed and integrated into Active Directory, all access to Elastic should be done using DCGS accounts. There are two Active Directory groups that are used for cluster administration.

ent elastic admins – Members of this group will oversee installations/upgrades and all configuration aspects of Elasticsearch and its components. This group should be limited to a very small group of people as it gives all privileges in Elastic.

ent kibana admins – Members of this group will oversee day-to-day operations with Elastic, which includes but is not limited to:

Ensuring the cluster is running correctly

Recovering from any cluster or ingest issues

Creating visuals/dashboards

7.8.1 Verify Role Mappings

Prior to checking Active Directory access, check the Kibana Role Mapping (which map AD groups to Kibana Roles):

Log into Kibana using the Elastic account.

Go to **Stack Management** and under **Security**, select **Role Mappings**.

Select the name of one of the Mappings and open the switch to JSON Editor link at the bottom.

Verify the string(s) contain(s) a valid AD group (it is possible that the string content got cut off, so the end may be missing).

Save, if necessary.

Repeat for all Mappings.

7.8.2 Verify Roles

As the installer of Elasticsearch, you should be a member of the **ent elastic admins** group. Verify that you are a member of this group and use your AD account to log in to Kibana.

NOTE: If you are not a member of **ent elastic admins**, you must find someone who is to verify they are able to log in to Kibana.

After logging into Kibana as an **ent elastic admin**, verify your privileges to ensure the role mappings have been created successfully. Execute the following from the Kibana console:

```
GET _security/user/_privileges
```

Verify the privileges are as follows (remember this is for a member of the **ent elastic admins** group):

```
{
  "cluster" : [
    "all"
  ],
  "global" : [ ],
  "indices" : [
    {
      "names" : [
        "*"
      ],
      "privileges" : [
        "all"
      ],
      "allow_restricted_indices" : true
    }
  ],
  "applications" : [
    {
      "application" : "*",
```

```
      "privileges" : [
        "*"
      ],
      "resources" : [
        "*"
      ]
    }
  ],
  "run_as" : [
    "*"
  ]
}
```

If everything looks good, remove unneeded accounts.

NOTE: Do not remove the following 6 accounts, the **kibana_xx**, **logstash_admin_user**, or **logstash_internal** accounts.

7.8.3 Remove Unneeded Accounts

When Elastic was initially set up, passwords were set for the following accounts:

elastic
apm_system
kibana_system
logstash_system
beats_system
remote_monitoring_user

During the installation of Logstash, the password for the **logstash_system** was modified but the rest have not been changed. These user accounts are **Reserved** accounts and cannot be deleted. To protect access to Elastic you must change the passwords to each of these accounts, record the passwords, and store them in a safe as **break-glass** passwords. As indicated previously, access to Elastic should now be accomplished using Active Directory accounts. As an extra measure, we will also disable the accounts that are not needed.

Log in to Kibana using your Active Directory account (member of **ent elastic admins**).

Navigate to **Management** > **Stack Management** on the side navigation menu.

Select **Users** under the **Security** section.

Set Break-Glass passwords for each of the following accounts:

elastic
apm_system
kibana
kibana_system
beats_system
remote_monitoring_user

NOTE: Do not change the **Logstash_system** account password.

Select the account and change the password using the **Edit User** interface.

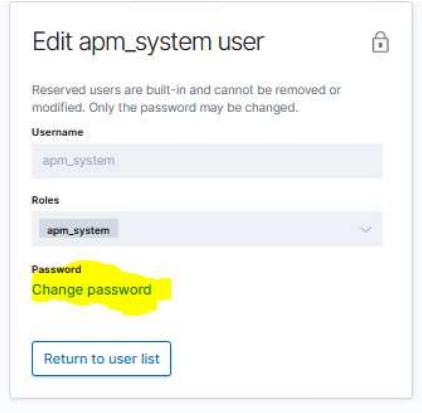


Figure 66 Edit User

Now that the passwords have been changed, disable the accounts we don't use for extra protection. From the Kibana console execute the following commands:

```
PUT _security/user/apm_system/_disable
PUT _security/user/kibana/_disable
PUT _security/user/kibana_system/_disable
PUT _security/user/beats_system/_disable
PUT _security/user/remote_monitoring_user/_disable
```

NOTE: Do not disable the **elastic** accounts as this will be the only way to log in to Elastic if you have issues with Active Directory authentication.

These user accounts will now show **Disabled** on the **Users** page in Kibana.

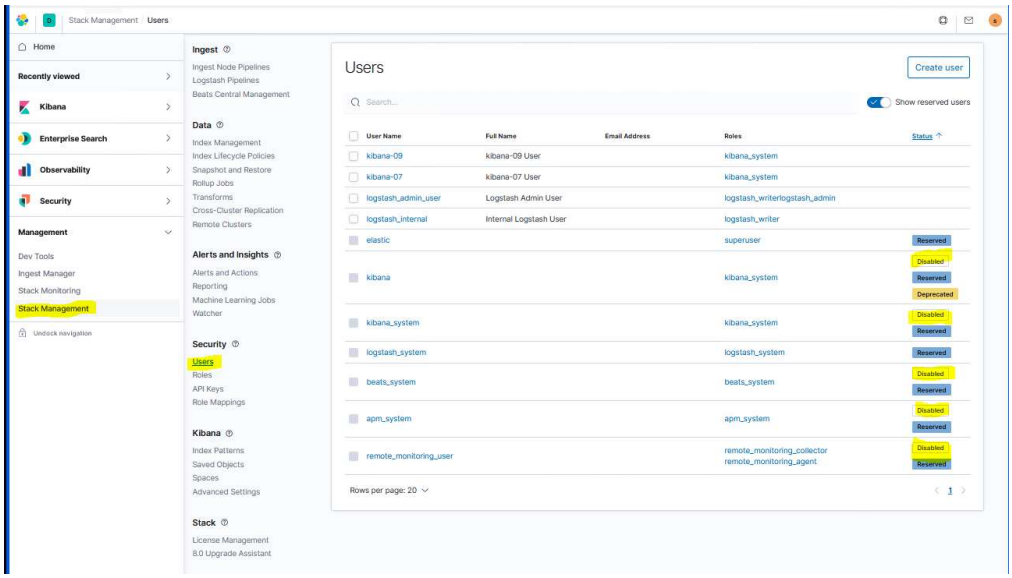


Figure 67 Disabled accounts

.

7.8.4 Verify Beat Templates are Loaded

Verify all beats templates were loaded by running the following command from the Kibana console:

```
GET _cat/templates/est*?v&s=name
```

This will list the templates, sorted by **name**. Verify the templates loaded for the following Beats:

est_filebeat-{version}
est_hearbeat-{version}
est_metricbeat-{version}
est_winlogbeat-{version}

7.8.5 Database Queries

Some of the data ingested into Elastic is queried from different SQL databases on DCGS. The Logstash instance at the site where the database is located is used to query any databases at that site. To execute queries against these databases, the Elastic service account **xx_elastic.svc** is used and must have read privileges for any database that is being queried. Before proceeding with enabling any database ingest, ensure the Elastic service account that will be used to query data is given privileges to read each database. Follow the steps in *ES-018 - Microsoft SQL – Configuring SQL for Elastic Monitoring Instructions* to give permissions to the Elastic service account.

NOTE: Database Administration privileges are required.

The Logstash pipelines used to execute the database queries rely on the Elastic service account password being in the Logstash KeyStore. The key holding the password must be added to the Logstash KeyStore of any Logstash instance that will make database queries.

The key name is **es_svc_acct_password**.

Execute the following command on the Logstash VM that will be using database pipelines to see if the key exists in the KeyStore:

```
# /usr/share/logstash/bin/logstash-keystore list --path.settings
/etc/logstash
```

NOTE: You will see the following errors caused by environment variables in our configuration, but the command still executes properly.

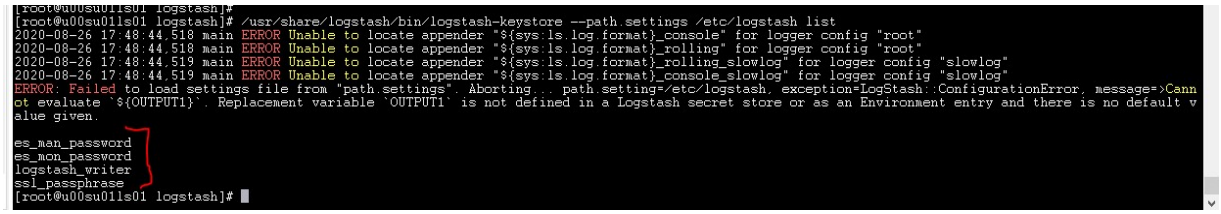


Figure 68 Errors

If the key is not listed, execute the following to add it:

```
# /usr/share/logstash/bin/logstash-keystore add es_svc_acct_password --
path.settings /etc/logstash
```

Enter the service account password when prompted and press Enter. You should then see:

Added 'es_svc_acct_password' to the Logstash KeyStore.

7.8.5.1 SCCM Database

The Logstash pipeline used to read SCCM data is **esp_sccm_database**.

Before turning on ingest for this data type, examine the pipeline in Kibana to ensure that the **jdbc_connection_string** to access the SCCM database is correct for your environment. (It should include the SQL instance and database names). Ensure the **jdbc_user** field is correct (xx_elastic.svc where XX = site number).

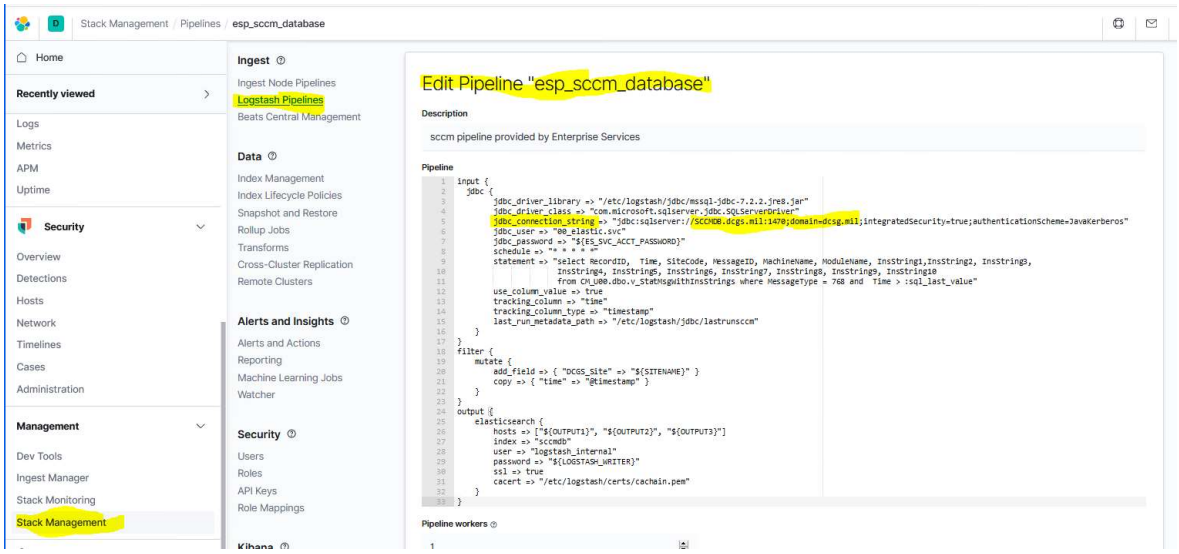


Figure 69 Edit SCCM Pipeline Example

In this version, minimal data is queried from the SCCM database. This data source will be matured in future releases.

To turn on the querying of SCCM data, add the **esp_sccm_database** pipeline to the **xpack.management.pipeline.id** array in the logstash.yml file and restart Logstash.

NOTE: There is only one instance of the SCCM database on the system; this pipeline should only be added to one instance of Logstash. On production it should be added to the ECH instance of Logstash. On CTE/MTE, choose the instance of Logstash that runs at the site where the Elastic Cluster is installed.

Once configuration is complete, verify SCCM data is received by going to Kibana **Discover** and selecting **sccmdb*** indexes in the drop-down under **+Add Filter**. Verify that hits are being received and the

number of hits displayed is increasing. Also confirm that the timestamp (underneath the hit count) is updating to reflect these changes.

Figure 70 Verify SCCM data is received

7.8.5.2 IDM Database

The Logstash pipeline used to read IDM data is **esp_idm_database**.

Before turning on ingest for this data type, examine the pipeline in Kibana to ensure that the **jdbc_connection_string** to access the IDM database is correct for your environment. (It should include the sql instance and database names.) Ensure the **jdbc_user** field is correct (xx_elastic.svc where XX = site number). Also ensure the **statement** field refers to the correct table name.

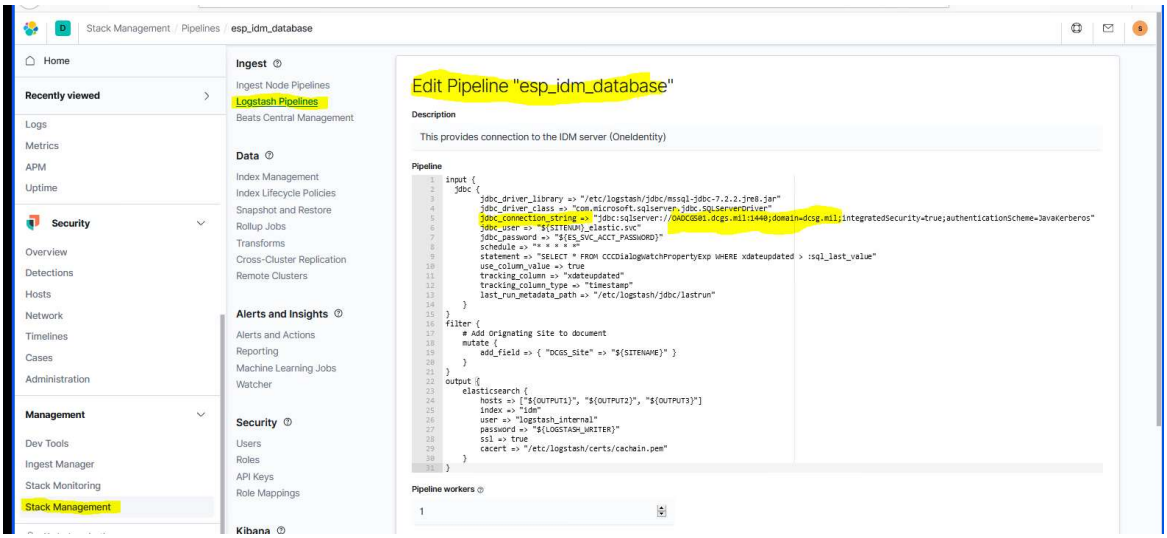


Figure 71 Edit IDM Pipeline Example

In this version, minimal data is queried from the IDM database. This data source will be matured in future releases.

To turn on the querying of SCCM data, add the **esp_idm_database** pipeline to the **xpack.management.pipeline.id** array in the logstash.yml file and restart Logstash.

NOTE: There is only one instance of the IDM database on the system; this pipeline should only be added to one instance of Logstash. On production it should be added to the ECH instance of Logstash. On CTE/MTE, choose the instance of Logstash that runs at the site where the Elastic Cluster is installed.

Once configuration is complete, verify idm data is received by going to Kibana **Discover** and selecting the **idm-*** indexes in the drop-down under **+Add Filter**. Verify that hits are being received and the number of hits displayed is increasing. Also confirm that the timestamp (underneath the hit count) is updating to reflect these changes.

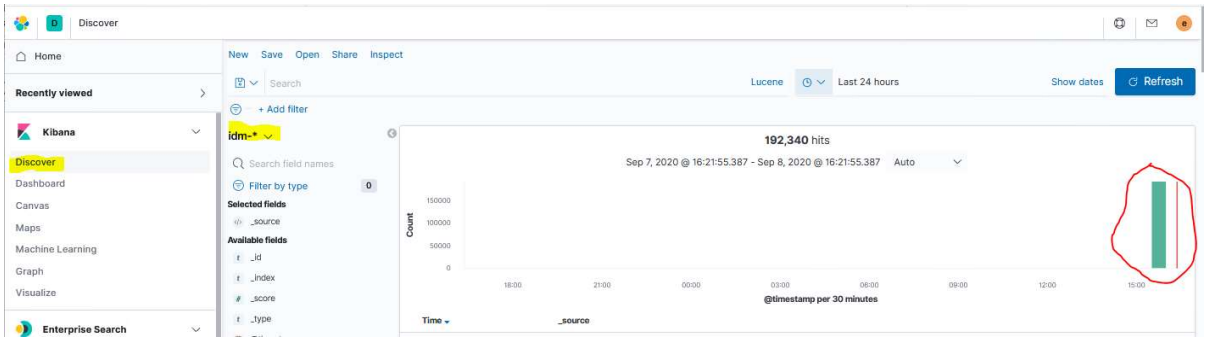


Figure 72 idm- indexes*

7.8.5.3 SQL Server Statistics

The Logstash pipeline used to read SQL Server Statistics data is **esp_sqlServer_stats**.

Before turning on ingest for this data type, examine the pipeline in Kibana to ensure that the **jdbc_connection_string** for each query to access the SQL database is correct for your environment. There are multiple queries used to gather this information so check with an SQL ADMIN to ensure they are correct for each Logstash instance you enable this query on.

To turn on the querying of SQL Server Statistics, add the **esp_sqlServer_stats** pipeline to the **xpack.management.pipeline.id** array in the logstash.yml file and restart Logstash.

NOTE: There are multiple SQL database instances throughout the system. This pipeline should be added to Logstash instances at sites where databases are located. You must check with the SQL Administrator to ensure the connection strings are correct for each site.

Once configuration is complete, verify SQL server data is received by going to Kibana **Discover** and selecting the **sqlserver-*** indexes in the drop-down under **+Add Filter**. Verify that hits are being received

and the number of hits displayed is increasing. Also confirm that the timestamp (underneath the hit count) is updating to reflect these changes.

Figure 73 Verify SQL server data is received

8 System Dashboards

The IAAS-ES-SYSTEM-Dashboard shows over-all status for hosts, applications, groups, data collectors, and devices.

Figure 74 SYSTEM Dashboard

8.1 Status Counts

The first row of visuals on the System dashboard show the numbers of hosts/applications/groups/data collectors separated by reported status. Selecting one of the counts showed will filter the dashboard based on that status (except in the first visual, Total Hosts). Only visuals that have that type of data will display.

Note that there may be a scroll indicator on the right side of these visuals, when more statuses are reported than can fit in the space provided – for example on the last 2 visuals in Figure 75.

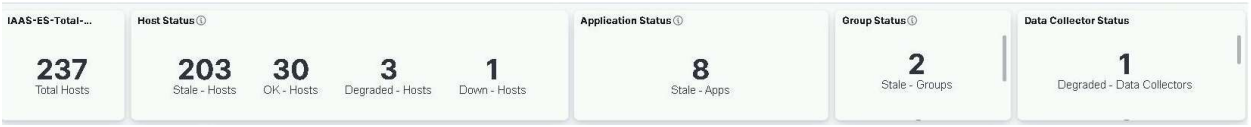


Figure 75 SYSTEM Dashboard - Status Counts

The Data Collector Status visual has a drill-down to the IAAS-ES-Data-Collector-Details dashboard. Click on the 3 dots in the upper-right corner of the visual (Options menu) and select Detailed Status.

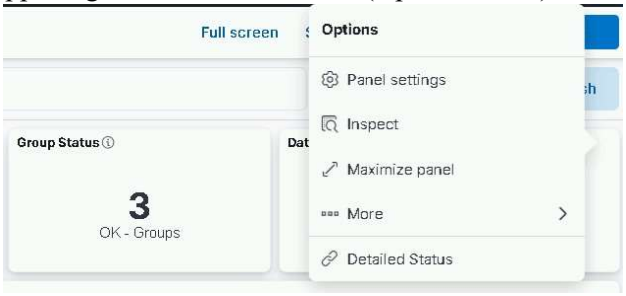


Figure 76 SYSTEM Dashboard - Data Collector Detailed Status

8.2 Syslog & Windows Events

The second set of visuals show pie charts and timeline of Syslog and Windows Events.

The two SYSLOG visuals have drill-downs to the IAAS-ES-SYSLOG-Dashboard. On the pie chart, the drill-down is shown as an option when a log level is selected; on the timeline, the drill-down is available from the options menu in the upper-right corner of the visual.

The two WINLOGBEAT visuals have drill-downs to the IAAS-ES-WINDOWS_Events-Dashboard. On the pie chart, the drill-down is shown as an option when a log level is selected; on the timeline, the drill-down is available from the options menu in the upper-right corner of the visual.

(Because the two timeline visuals are defined with a specific time span (last 4 hours), the data displayed will not change if a time span is selected from within the visual.)

Figure 77 SYSTEM Dashboard - Syslog & Windows Events

8.3 App Status & Problem Hosts

The next section contains visuals for application and host health.

Current App Status shows the health of applications by site, and any symptoms reported. Selecting an Application name drills-down to the IAAS-ES-SYSTEM-Application-Info dashboard.

Current Problem Hosts shows any hosts reporting problems with health and symptoms along with CPU, memory, and disk space usage. Selecting a Host name drill-down to the IAAS-ES-Host Dashboard.

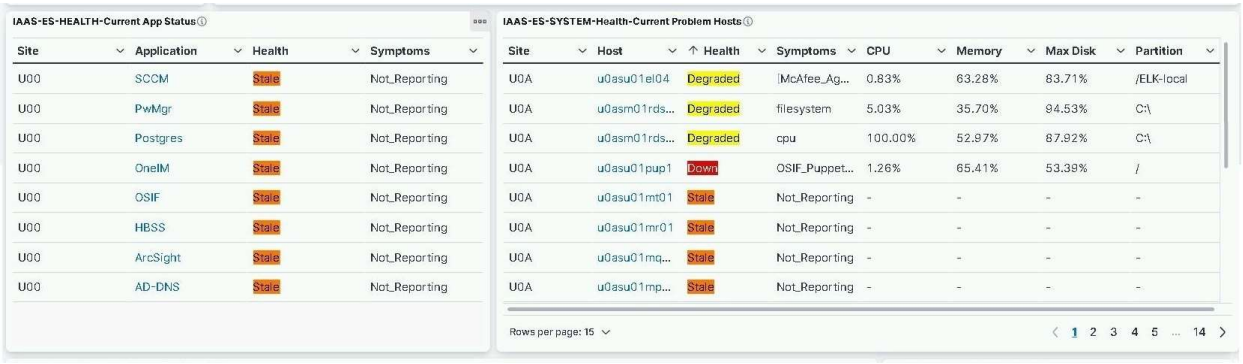


Figure 78 SYSTEM Dashboard - App Status & Problem Hosts

8.4 Problem Apps & Devices

The next section contains visuals for problem applications and device status.

Current Problem Apps shows any applications reporting problems, along with the associated host, and process or service issues. Selecting an Application name drills-down to the IAAS-ES-SYSTEM-Application-Info dashboard. Selecting a Host name drill-down to the IAAS-ES-Host Dashboard.

Device Brief Status shows devices, types, and status; it has a drill-down to IAAS-ES-Infrastructure Status Dashboard from the Options menu (upper-right corner of the visual).

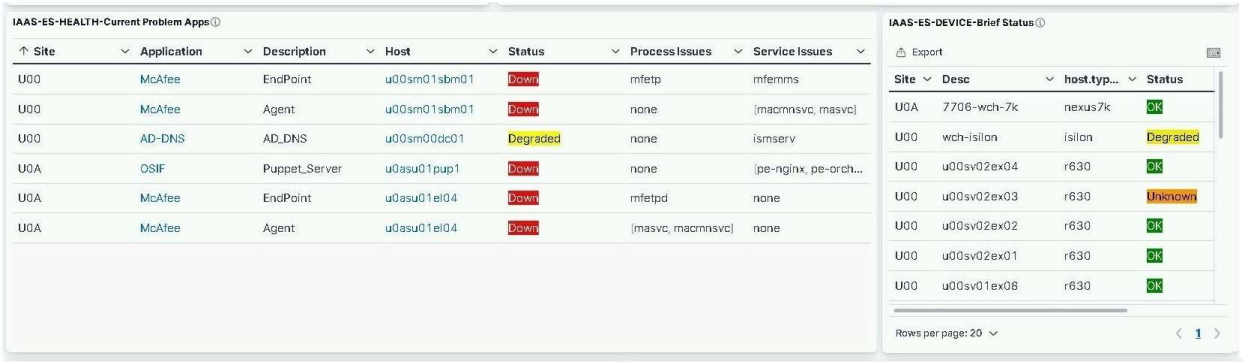


Figure 79 SYSTEM Dashboard - Problem Apps & Devices

8.5 Metricbeat Host CPU & Memory

The next section contains visuals showing the hosts with the top CPU usage and top memory usage. Clicking on a host in either visual will open the IEES-ES-Host Dashboard filtered on the selected host.

Figure 80 SYSTEM Dashboard - Metricbeat Host CPU & Memory

8.6 Groups

The next section shows the status of any host groups defined (for example sets of related servers or mission workstations). See 2.2.4.2 for settings up groups in groups.ini.

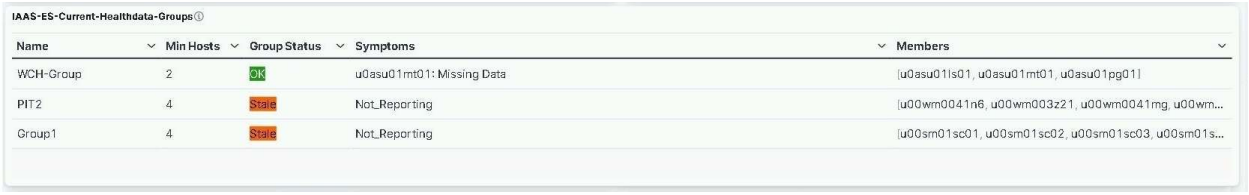


Figure 81 SYSTEM Dashboard - Groups

8.7 Puppet Agent

The next section shows the status of Puppet Agent, showing any systems where Agent has been disables or has failed.

Figure 82 SYSTEM Dashboard – Puppet Agent

.

9 References

ES-018 - Elastic Logging and Aggregation Cluster (ELAC) - Upgrade to 7.12.1 Instructions
ES-024 - Puppet - System Administrator Guide
ES-018 - VMware - NSX-V Load Balancer Deployment Guide
ES-018 - Microsoft SQL – Configuring SQL for Elastic Monitoring Instructions
Appendix A Acronyms

Acronym	
AD	Active Directory
ART	Agile Release Train
CA	Certificate Authority
CN	Common Name
CSR	Certificate Signing Request
CTE	Controlled Test Environment
DCGS	Distributed Common Ground System
DNS	Domain Name Service
DSIL	DCGS System Integration Lab
ECH	East Coast Hub
EPO	ePolicy Orchestrator
FQDN	Fully Qualified Domain Name
GPO	Group Policy Object
HBSS	Host Bases Security System
ICMP	Internet Control Message Protocol
IDM	Identity Management
ILM	Index Lifecycle Management
ISEC	ISR Systems Engineering Center
ISR	Intelligence, Surveillance, Reconnaissance
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
JWICS	Joint Worldwide Intelligence Communication System
KQL	Kibana Query Language
MIB	Management Information Databases
MTE	Managed Test Environment
NRT	Near Real Time
NSX	a virtual networking and security software product family from VMware
OA	Open Architecture
OSIF	OA DCGS Secure Infrastructure Framework
	Operating System Interface Oh Shit I Forgot
RPM	RedHat Package Management
SAKM	Service Account Kerberos Management
SCCM	System Center Configuration Manager
SME	Subject Matter Expert
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSD	Solid State Drive
TCP	Transmission Control Protocol

TLS	Transport Layer Security
UUID	Universally Unique IDentifier
VCPUs	Virtual CPUs
VM	Virtual Machine
WCH	West Coast Hub
WEC	Windows Event Collector