

IMPERVA OPEN REPORTING QUICKSTART GUIDE

Based on ElasticSearch, Kibana, and Logstash

**January 2020**

# Imperva Open Reporting

The product intends to address reporting and visualization needs of customers that are current users of Imperva Database Activity Monitoring (DAM) solution with capabilities to ingest and process massive data. Built using open source technologies (ELK stack), it is intended as a recipe for customers to manage their own reports and dashboards to suit individual reporting use cases. The offering will be open source and community supported in Imperva’s Github repository, allowing for users build on the platform.

### Intended use cases

### 1. Self- Service Visualization and Dashboarding

Kibana provides rich options to view DAM activity data flowing into the system. Querying in Kibana is simple with prompts and suggestions as one builds on the data to query. This can then be used to build visualizations by aggregating and bucketing data based on one or more fields.

### 2. Security Analytics

The dashboards provide the ability to drill down deeper into visualizations to understand patterns and variations in the data. For example, a sudden spike in DDL queries or failed logins is a red flag that can be dug deeper into.

### 3. Audit and Compliance

By setting up a policy in SecureSphere to send all events to LSAR, customers are able to perform audit and compliance on all database activities. Each of the reports or visualizations in the Kibana dashboard can be exported to a CSV raw data file (for free) or a PDF ([requires Elastic platinum license](https://www.elastic.co/subscriptions)). This CSV can be formatted into compliance reports using other BI tools such as Microsoft Excel and Tableau.

### High Level Architecture

The platform relies on syslog events generated by SecureSphere gateways. This can be configured through a policy and followed action on SecureSphere( see Installation section for details). These messages are consumed by the ingestion engine, Logstash. Elasticsearch indexes this data across all of its data nodes optimized for effective searching. Kibana provides a rich reporting and visualization layer with real time querying ability to access this indexed data.

## 

### Terminology

#### Cluster

A group of nodes that work together to power the Elasticsearch.

#### Node

A java process that run the Elasticsearch software. Multiple nodes can be inside a single box (VM or physical machine) when using containers.

#### Index

A group of shards that form a logical data store

#### Shard

A Lucene index that stores and processes a portion of Elasticsearch index.

#### Segment

A Lucene segment that immutably stores a portion of Lucene index. This is a level of granularity that one typically doesn’t have to worry about for most sizing exercises.

#### Document

A record that is submitted to and retrieved from an Elasticsearch index. In this configuration, it represents a single event in a database.

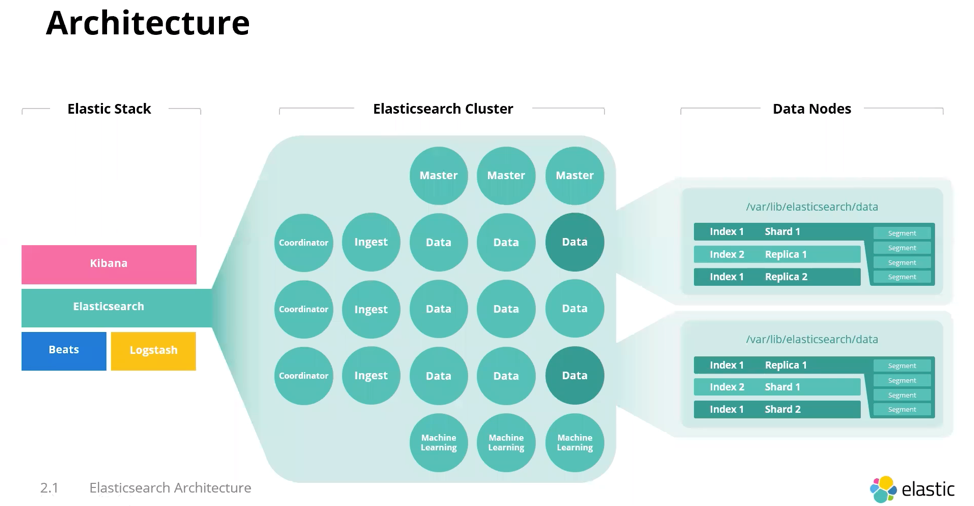
#### 

### Typical Elasticsearch Cluster

#### Types of nodes and their roles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role** | **Description** | **Resources** | | | |
| Storage | Memory | Compute | Network |
| **Data** | Indexes, stores and searches data | **Extreme** | **High** | **High** | **Medium** |
| **Master** | Manages state of the cluster | **Low** | **Low** | **Low** | **Low** |
| **Ingest** | Transforms inbound data (optional) | **Low** | **Medium** | **High** | **Medium** |
| **Coordinator** | Delegates requests and merges search results (optional) | **Low** | **Medium** | **Medium** | **Medium** |

Data Node is the node that drives scaling guidelines. Most other node types can work with very little resources (4-8 GB of RAM) even for large clusters.



### Sizing Parameters

#### Storage

Persistent storage is where the data is actually stored such as a Solid State Drive (SSD) or a Hard Disk Drive. SSDs offer the best performance for “hot” workloads. HDDs are economic and work for “warm” and “frozen” storage. The preferred option is to have Direct Attached Storage (DAS) but high performance Storage Area Network (SAN) works too. Network Attached Storage (NAS) is not recommended for the latency and expensive network protocol overhead.

#### Memory

Memory is where data is buffered and usually represents available RAM. Elasticsearch uses memory in two ways: JVM Heap and OS Cache. JVM Heap stores the cluster, indices, shards, segments and field data. For configuration purposes, set JVM Heap to about 50% of the available RAM. OS Cache is used for caching data, and for optimizing aggregation and sorting operations. The remainder of available RAM goes into this.

#### Compute

Elasticsearch processes data in many ways that can be computationally expensive, such as indexing. The quantity and performance of CPU cores governs the average speed and peak throughput of data operations in Elasticsearch.

#### Network

Bandwidth is rarely a resource that constrains Elasticsearch (unless it’s a very large cluster > 50 nodes). There is some communication between nodes, but this is usually not a limiting factor. Having the highest possible network speed (usually gigabit ethernet) and having all nodes in the same region is preferred.

### Installation

##### Logstash

The Dockerfile located in ./logstash will build the fluentd docker image based on the latest version the first time the docker-compose.yml file is run and install the Elasticsearch output plugin.

The configuration file, ./fluentd/config/logstash.yml, is configured to listen for JSON formatted syslog messages on TCP port 5514, and then write those logs to the Elasticsearch data system in daily indexes titled "lsar". The index name can easily be changed by editing that file, and changing the name of the index.

##### Kibana

The configuration file ./kibana/kibana.yml contains the basic Kibana configuration. Changing the elasticsearch.hosts configuration will allow you to point Kibana at a different host. It is set to listen on the standard TCP port of 5601 on the Docker host. This can be changed in the docker-compose.yml file.

##### Elasticsearch

This instance of elasticsearch is configured to listen on the standard TCP port of 9200. This can be changed in the docker-compose.yml file. When you first bring up the system, there is no index created yet. Once SecureSPhere has been configured to send audit data to the system, an index starting with "lsar-" will be created. A new index will be built daily. Creating a master index of "lsar-" will allow all of the daily indexes to be rolled up into a single index for reporting and visualization purposes.

#### Deployment Mode 1 : Docker - Single Stack

The instructions below are to setup the environment with one node each of Logstash, Kibana and Elasticsearch.

* From the base directory (lsar-p1), navigate to single-node folder

cd single-node

* Let Docker install the required packages

docker-compose build

* Bring the services up

docker-compose up

* [optional] Bring the services up in the background (kill the current running process from previous command)

docker-compose up -d

* Open up a browser and Kibana should load at the following URL

http://localhost:5601

#### Infrastructure code for multi node deployments

The source code comes with Terraform templates to create EC2 instances and Ansible scripts to set up and configure the software.

* From the repo’s base directory, navigate to tools/infrastructure-setup/ansible/playbooks to use the appropriate playbook based on your environment of choice (bare metal / VM, Docker Swarm, or Kubernetes).
* If you are working with AWS EC2, run the following commands from the './terraform/aws' directory you can use our terraform scripts to quickly create EC2 instances
  + To use the terraform scripts you must have an AWS private key set up already. Set the key variable in ‘./terraform/aws/variables.tf’ to the key name.
  + Set the region and number of instances you want
  + In ‘./terraform/aws/instances.tf’ set the instance\_type variable to the machine variant you want to use. (We recommend no less than 16GB of ram and a good network connection)
  + Uncomment the ebs\_block\_device section if you want to use EBS.
  + Initialize terraform and install providers

terraform init

* + Plan and apply the desired setup

terraform plan

terraform apply

#### Deployment Mode 2 : Docker swarm with cluster of nodes

* Navigate to the “swarm” directory from the base directory (lsar-p1)
* Set up your docker swarm

docker swarm init

* From the repo's base directory, build the required images using the following command. You only need to build the images the first time

docker-compose build

* To deploy the stack run the following command:

docker stack deploy --compose-file ./docker-compose.yml lsar

* To see if all containers have started correctly run the command below (NOTE: It may take a few minutes for the services to start)

docker service ls

* Open up a browser and Kibana should load at the following URL

http://localhost:5601

#### Deployment Mode 3 : Install directly on bare metal / VM (containerless mode)

To quickly create a bare metal cluster, use the Ansible play we have created. This removes the hassle of installing and configuring software manually. Before proceeding you must have SSH keys set up for your machines and a user that can gain root access. Use terraform templates

Current Ansible scripts are tested with Amazon Linux machines images, and is expected to work as-is for any RHEL / Cent OS based linux. This may not work with Debian based distributions.

* Add an IP addresses to the different blocks in the hosts file './ansible/inventory/production/bare\_metal\_hosts'.
* You must define 1 node-master.

|  |  |
| --- | --- |
| **Group** | **Explanation** |
| node-master | The initial master node that all other masters/data nodes will communicate with. This node will start without waiting for other nodes. This prevents an endless loop of master nodes waiting for an initial master node to start. |
| master | Nodes listed here will be reserved for Elasticsearch nodes configured as master nodes. Check the Elastic website for the latest recommendations. |
| data | Nodes listed here will be reserved for Elasticsearch nodes configured as data nodes |

##### Running the playbook

* Before running the playbook update the following sections of the playbook to match your infrastructure.
  + In the playbook (ansibe / baremetal-
  + Update *es\_heap\_size in*  to a suitable value (No more than 32GB is recommended by Elastic)
  + Update *es\_data\_dirs* to point to a suitable location on your machines disk
  + If you are not using AWS set the **mount\_volume** variable to false.
  + Under *‘./ansible/config-files/logstash/jvm.options’* change -Xms1g and -Xmx1g to suitable values (more is better but both values must be the same) based on your machines memory.
  + Install Elasticsearch ansible module using command

*ansible-galaxy install elastic.elasticsearch,7.4.1*

* Once you have divvied up your machines and changed the variables, run the following command to create your swarm:

ANSIBLE\_HOST\_KEY\_CHECKING=False ansible-playbook ./playbooks/setup\_bare\_metal\_cluster.yml -i ./inventory/production/bare\_metal\_hosts --private-key=./{{ KEY\_NAME }} -u {{ USERNAME }} --fork 10 -e serial\_number=10

* Install Kibana through Yum (no current Ansible recipe available) in one of the hosts (preferable to use ElasticSearch Master node) from
  + rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
  + sudo yum install kibana
  + Edit **/etc/kibana/kibana.yml** to match tools/infrastructure-setup/confg-file/kibana.yml (add master / coordinator node ip address to elasticsearch.hosts property)
  + Restart Kibana **sudo -i service kibana restart**
* This will install the necessary software and start relevant services.

##### Current limitations to the Ansible script playbook

* Kibana still requires manual installation. However, since this is to be done only on one node, it is easier to be done separately using scripts.
* Re-running the playbook will not reinstall the elasticsearch cluster.

### SecureSphere configuration

#### Setup Action Interface

1. From Admin > System Interfaces > Action Interface > create new action interface.
2. Select TCP for protocol, use the address of the server running the fluentd listener (running on 5514 by default). Syslog log level can be “info”, facility can be left at “USER”
3. Message is message string located in the files folder files/syslog-message.txt (from the gitlab repository)

A screenshot of a computer

Description automatically generated

#### Create Action Sets

1. From Policies > Action Sets, create new action set
2. Use the Action interface built in the previous step for this action set.
3. Apply the policy to your database servers from Audit Policies. Under External logger, pull down the action interface created.

A screenshot of a computer screen

Description automatically generated

### Using Kibana

Kibana is the self-serve reporting platform that offers powerful querying and visualization capabilities. The repository comes with pre-existing canned reports based on some common use cases. However, it is quite simple to create your own reports and visualizations. For more details on Kibana, please visit <https://www.elastic.co/guide/en/kibana/current/introduction.html>

#### Currently available pre-canned reports

The following pre-canned reports are currently available. But it is fairly simple to create your own reports. Please see “Creating your own reports” section for details

##### Summary Views

1. Data Summary
2. Environment Summary

##### Server Analysis

1. Monitored Servers
2. Monitored Databases
3. DB Server Performance

##### Source Analysis

1. Shared DB Users
2. DB Users
3. Source Applications
4. Source Hosts
5. Login Analysis

##### Data Access Patterns

1. Top Queries
2. Query Type Analysis
3. Query Records

##### Privileged Operations

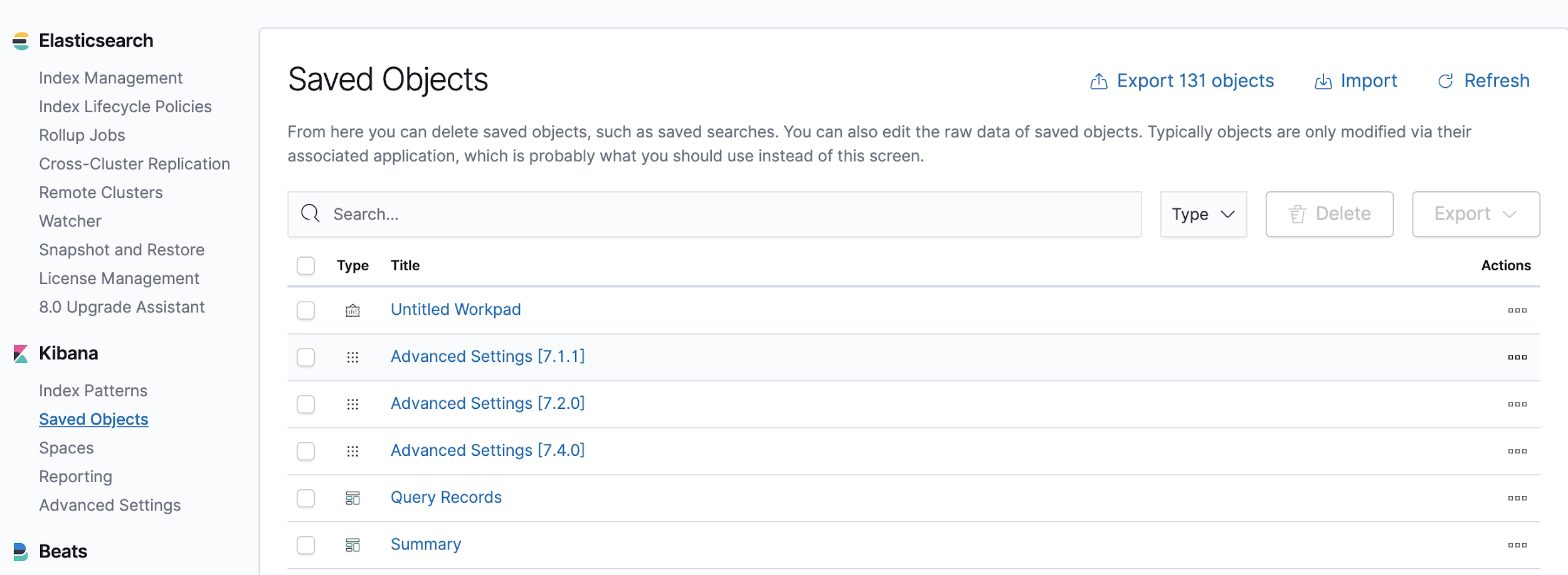
1. Table Drops / Truncates
2. Changes to DB / Schemas

##### Error Analysis

1. SQL Errors
2. Failed Logins

#### Importing existing dashboards

Import the dashboards in **files/lsar-p1-export.json** from Kibana GUI by selecting saved objects under the management menu from the left sidebar.



This will provide access to all canned reports from Imperva

#### Creating your own reports

Searching and building custom visualizations in Kibana is straightforward. It uses Kibana Query Language (KQL) to build appropriate filters to view data. These saved queries can further be visualized through custom visualization with appropriate aggregations and buckets. Please see the links below for more details.

<https://www.elastic.co/guide/en/kibana/current/search.html>

### Support

This product will be open-source and community supported to allow for customers to build this solution to their specific reporting requirements. Imperva customer support will be restricted to providing assistance to export the data using syslog events.

### Limitations

1. While the stack’s deployment and administration are made simple through containers, in-house talent on the ELK stack is expected. Imperva Professional Services might be able to help with the initial deployment and customization of reports based on your organization requirements.
2. The infrastructure footprint on the stack is heavy as can be seen from sizing recommendations. This is the case with most big data search and indexing platforms due to the volume of data. Customers can consider options such as IaaS or Elastic as a service for simplified pay-as-you-go deployments.