



Log Management Solution

Architecture and Implementation Details

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 4/9/2015 | V1 | Harmony Log Management Architecture | Rahul B |
| 5/1/2015 | V2 | Updates from Engineering and Stability Team | Rahul B |
| 5/6/2015 | V3 | Hardware requirements | Rahul B |
| 5/15/2005 | V4 | Updated doc based on discussions with DevOps and Engineering | Rahul B |
| 5/18/2005 | V5 | Hardware details | Rahul B |
| 5/20/2015 | V6 | Installation steps updated | Carlos T |

Purpose 2

Scope and Context 2

Business Objectives and Constraints 2

Objectives 2

Constraints 2

Assumptions 2

Risks 3

Log Collection 3

Solution Overview 3

Architecture Guiding Principles 3

Harmony Component Overview: Supported by Log Management 3

Log Management Component Overview 4

Solution Design 5

Harmony Implementation Approach 5

Application Overview 6

Debbugging Scenario for Engineers (TBD) 6

TargetServers for collecting logs 7

Archieve Policy 7

Expectations for data 7

Collection 7

Transport 7

Storage 7

Analysis 7

Alerting 7

Metric Graphs 8

Implementation Steps 8

Key Team Members 9

Schedule 9

Environment Details For Setup 9

Platform Environment Specification 9

Epsilon Environment Mappings 10

Epsilon Virtual Host Definitions 10

High Availability and Failover 10

Lewisville Datacenter Scenario for Acceptance / Stage 10

Multi Datacenter Failover – Lewisville Production 10

Go-Live Check List 11

Additional Information (AS NEEDED) 11

Key Concepts and Terminology 12

Supporting Documents 12

Purpose

Purpose of this document is to describe the overall Harmony Log Management solution at a high level. It states the problem definition(s) and outlines the design, architecture, and interactions of the Harmony solutions with other components. There are supporting documents, which provide a greater level of detail for some sections within this document. You can find reference of these documents

Scope and Context

Business Objectives and Constraints

Objectives

[Summarize the challenge. What are the business objectives the solution addresses?]

We want to collect logs and necessary transaction data to analyze and mine for trends, statistics, summarizations, or anomalies. Correlate logs, performance metrics, events, custom metrics, and business KPIs.

Few of key objectives

* Centralization
* Parsing
* Storage
* Search

Extended objectives

* Segregation of Duties for audit purpose
* Automate detection
* Detect anomalies using support to metrics engines
* Prevent data leaks
* Failure patterns
* Advance alerting and Generating Reports for Key Stakeholders)
* Easy discovery of root cause
* Automate troubleshooting using graphing

Constraints

Below are the lists of business or technical constraints considered on the log management solution.

Assumptions

Below are the lists of business or technical assumptions considered on the log management solution.

Encryption: We will use SSL encryption for log data where it is needed. For this we will use log forwarder. SHA1

Access: Authentication: Engineers or users of log information will be able to authenticate using network ID. They will be able to create visualizations and necessary queries for future use.

Technical safeguards: Technology, policy, and procedures deploying necessary configurations to the server will be automated using deployment scripts.

Risks

* Data loss during indexer, forwarder or broker being unavailable.
* High unexpected volume of data
* Necessary infrastructure unavailability for scaling necessary modules of log management server
* Data to be truncated after 120 days

Log Collection

* Automated collection of log files
* Supports Windows Event Logs – both .evt and .evtx formats
* Supports Syslog log files
* Configure to clear or not clear log files
* Collects all generated events
* Collects only certain types of events
* Export log data from one source to another

Solution Overview

In this case, our design uses Lumberjack forwards or Shipper logs to Logstash that collects, aggregate, and parse application log data through the broker, and then Logstash feeds this data into Elasticsearch. Once the data is in Elasticsearch, we run searches and aggregations to mine information that is of interest to business scenarios for our clients.

Architecture Guiding Principles

This section provides the key architecture principles used with the Harmony Log solution(s). This includes, but is not limited to: solution standards and protocols, design principal agreements.

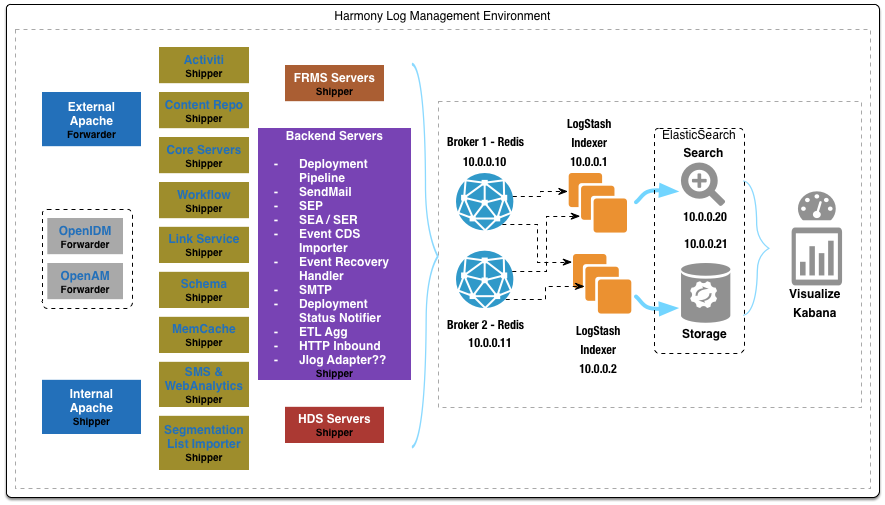
Log Management: Inputs 🡪 Filters 🡪 Output: Collect, parse, store and forward.

Filter: Syntax recommendations (GITHUB: http://github01.epsilon.com/ctorres/Log-Management)

Index Recommendations: Example, Response is used for all response codes. Verbs is used for POST, GET, DELETE options

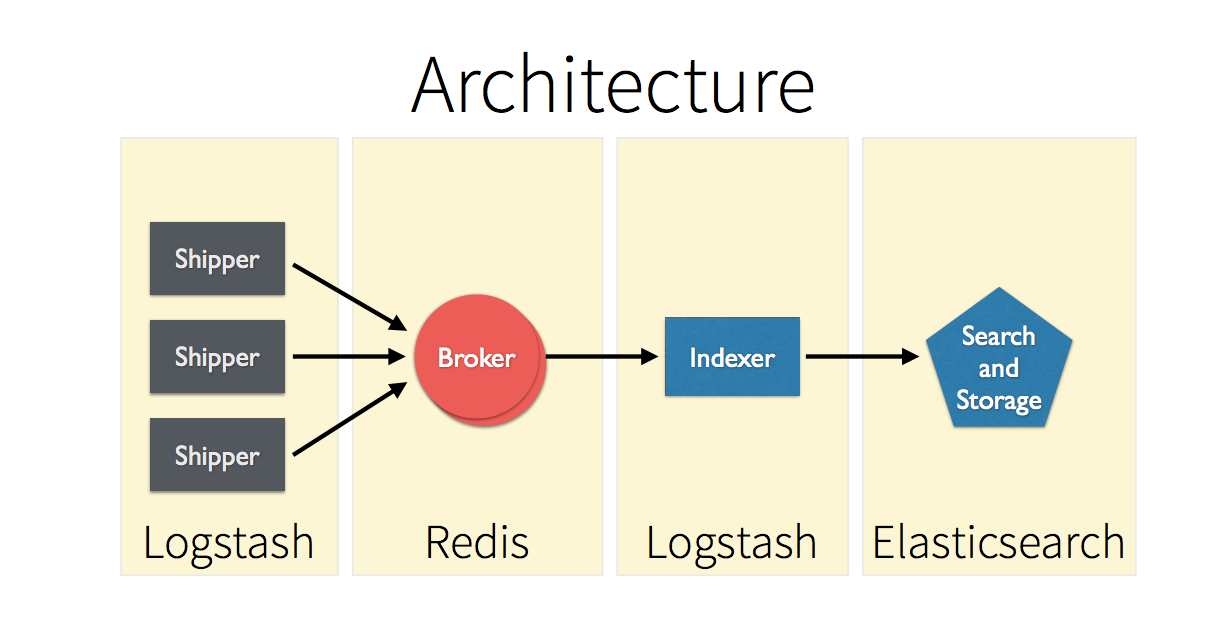
Harmony Component Overview: Supported by Log Management

This section describes a high-level overview of the Harmony log management infrastructure setup.

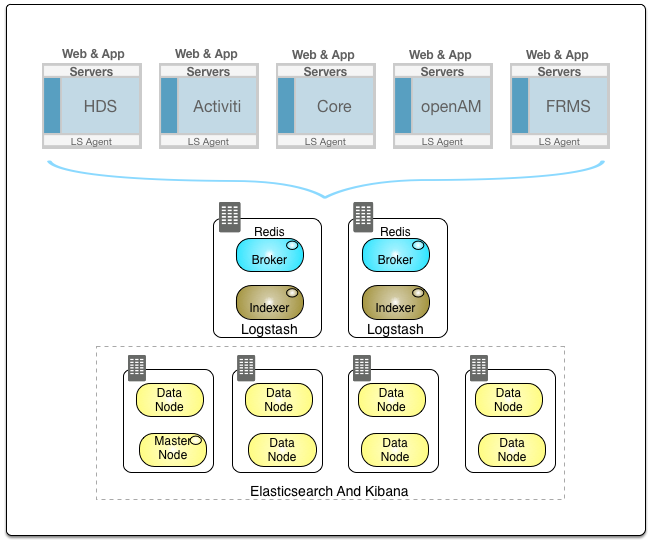


Log Management Component Overview

This section describes the overall interaction between components such as: client applications, Epsilon components, and backend services, in this case logstash and elastic search overview. This is how we will be launching in a beta phase.



To scale, only if we get to a limit of 60K per second. Replace Redis with kafka, replay the last stored data from disc.



Future Phase: We will introduce Hadoop and flume into log management solution for archival of data (more than a one month).

Solution Design

Harmony Implementation Approach

This section describes the key elements of the Log Management solution. For each component, an overview of its role and integration with the complete solution is provided. Additional details regarding a component may be referenced in a supporting document.

**Implementation is Phased approach**

This section is to manage the growth of log data on logstash and elastic search

|  |  |
| --- | --- |
| Component | Log details |
| OpenAM Servers | Only Access Logs |
| Front end Servers | Only Access Logs |
| Back end Servers | Only Access Logs |
| Apache Servers | All Logs |
| JBOSS | All Logs |
| API Servers | Only Access Logs |

Application Overview

The detailed configuration of a developed shipper, forwarder, indexer and elastic search application will be captured in the. conf files. That code lives with the source code (GIT) log management project as it is directly related to development effort of scaling this solution. Additionally, abstracted information related to a solution’s architecture is present in the above below cases. Below is a list of all Epsilon Filters for reference:

|  |  |
| --- | --- |
| Filters | Supported |
| Grok |  |
| Date |  |
| Mutate |  |
| csv |  |
| Geoip: (Maxmind) |  |
| Kv: key value pair |  |
| Output | Supported |
| Storage: | **Hadoop** |
| Notification | **Nagios, Emails** |
| Relay: | **TCP, Redis, Syslog** |
| Metrics | **Ganglia** |

Debbugging Scenario for Engineers (TBD)

This section details the implementation use for L3 and Engineering Support.

|  |  |
| --- | --- |
| Use Case Name | Debugging Scenario for L3 and Engineering Team |
| Actors | L3 and Engineering Teams |
| Pre-Conditions | Access to Elastic Search and Kibana |
| Success End Condition | Able to connect end to end scenario of the client from a request and response perspective |
| Failed End Condition | Missing one of the log agents or logs from the server |
| Post-Conditions | Able to visualize and search for relevant fields. |
| Basic Flow | 1. Engineers log into ES to see if we have any relevant activity. 2. Search for particular 200 status for error analysis 3. Customers coming from IP analysis with Geo support 4. DDOS attach 5. OpenAM dashboard (login’s and log out’s) 6. Metrics flow to see how campaigns are doing |
| Extension Points | Build analytics (Similar like google analytics) |

TargetServers for collecting logs

Logs have been identified. Because of reasons of security, we will be directly getting this information from Engineering Services portal.

Archieve Policy

Archiving data according to relevant data retention policies, including provisions for the appropriate level of data protection – for example, off-site storage.

Expectations for data

Provide any expectations for logs that need to be stored for longer duration of time.

Collection

This section provides an overview of Harmony Log collection details. It will describe the the measures and explain what is represented by the data and how it is collected. Application component creates logs in different ways, some log through syslog, others log directly to files.

Transport

Provide an overview of how data is collected by lumberjack, syslog and shipper into ELK. (TBD)

Storage

**How long should it be stored:**

|  |  |  |  |
| --- | --- | --- | --- |
| Type of Storage | Epsilon Environment | Data Volume | Epsilon Host |
| Real Time, Troubleshooting | prod | ELK - 15 Days | Epsilon.example.internal:8443 |
| Long-Term, Archival purpose | prod | ELK Hadoop instance – 3 Months | Epsilon.example.internal:8443 |

**Production data volume:** Allows you to scale-out horizontally.

**Access to the logs:** Elastic serarch will be used to support raw data formats.

Analysis

Kibana will be used for analysis

Alerting

**Email and Notify:** Respective teams will be sent and email or text messages in the below format. We will also send these messages into Nagio’s.

**Exception Monitoring**: Log patterns with errors that indicate problems. Aggregate repetitve exceptions

**Performance Monitoring**: Most of the data captured as part of metrics will be routed to collectD for visualization purpose on Graphana.

**Calculated Metrics**: TBD (Riemann) is another option for us to send calculated metrics.

Metric Graphs

Graphana along with CollectD will be used to visualize metrics data. We will also use CollectD for stats on log management hosts.

Implementation Steps

http://github01.epsilon.com/ctorres/Log-Management/tree/poc

* Install Logstash
  + Build a single central LogStash server
  + Move logstash into /opt/logstash
  + Configuration /etc/logstash
  + Logstash output /var/log/logstash
* Installing broker (it’s a buffer between agent and server)
  + Download stable version,
  + Install (./install\_server.sh) redis-server from utils
* Installing Shipper & Lumberjack steps
  + Need to complie the code using GO. We will check this code into GIT.
  + Configuring central server to receive events
    - Configuration Steps:

<https://www.elastic.co/downloads/logstash>

* + Index and make the data available to search.
  + Install LogStash on a remote agent
  + Configure LogStash to send some selected log events from our
    - Remote agent to our central server.
* Install Sheild for security (out of scope Phase 1)
  + bin/plugin -i elasticsearch/license/latest  
    bin/plugin -i elasticsearch/shield/latest
  + Adding admin as a user: bin/shield/esusers useradd es\_admin -r admin
  + Test 1: Curl –XGET ‘http://localhost:9200/
  + Test 2: curl -u es\_admin -XGET 'http://localhost:9200/'
  + Adding User:
* Install ElasticSearch
  + Download the latest tar and untar (version 1.5.0)
  + Running ES: bin/elasticsearch –f
  + Test: curl localhost:9200
  + Starting 2nd node: bin/elasticsearch f -Des.node.name=harmony\_2
  + Web front end for cluster: Install Head Plugin: bin/plugin -install mobz/elasticsearch-head
  + Test Mappings: curl localhost:9200/documents/\_mapping
  + Test Cluster: curl -XGET <http://localhost:9200/_cluster/health>
  + Test Health: curl -XGET "http://localhost:9201/\_cluster/health"
  + Test Get Data: curl -XGET <http://localhost:9200/documents/blog/one>
  + Test Search: curl -XGET "http://localhost:9200/documents/blog/\_search?q=\_id:one"
* Install the LogStash Kibana agent to act as a web console.
  + To use a local client node to load balance Kibana requests:

Use : [LINUX 64-BIT](https://www.elastic.co/thank-you?url=https://download.elastic.co/kibana/kibana/kibana-4.0.2-linux-x64.tar.gz)

* + Install Elasticsearch on the same machine as Kibana.
  + Configure the node as a client node. In elasticsearch.yml, set both node.data and node.master to false:
  + You want this node to be neither master nor data node, but # to act as a "search load balancer" (fetching data from nodes, # aggregating results, etc.) # node.master: false node.data: false
  + Configure the client node to join your Elasticsearch cluster. In elasticsearch.yml, set the cluster.name to the name of your cluster.
  + cluster.name: "my\_cluster"
  + Make sure Kibana is configured to point to your local client node. In kibana.yml, the elasticsearch\_url should be set to localhost:9200.
  + The Elasticsearch instance to use for all your queries. elasticsearch\_url: "http://localhost:9200"
* Install ElasticSearch enable Hadoop

Key Team Members

|  |  |
| --- | --- |
| Name | Role |
| Engineering Team | Individual engineering teams will help us with filters |
| Scalability Team | Dev, log management solution, POC, Architecture, Implementation guide and project lead |
| DevOps | Implementation in stage and production, this team will also help us with infrastrucute needs |
| Support (L1, L2, L3) | Verify and test the implementation |

Schedule

|  |  |
| --- | --- |
| Server Name | Implementation Date (Prod only) |
| Procure Infrastrucute | End of Month (May), new servers will be added with below week schedule |
| Pipeline | Jun First week Start |
| CDS | Jun second week Start |
| External and Internal Apache | Jun third week Start |
| OpenAM | July First week Start |
| Core | July Second week Start |
| Reporting | July Third week Start |
| FRMS | July Fourth week Start |
| MDM | Aug First week Start |
| HDE | TBD |

Environment Details For Setup

Platform Environment Specification

In the case of on-premise Epsilon installations, the infrastructure architecture is captured in the Environment Specification document. Refer to the Supporting Documents section of the Appendix for the location of the Environment Specification document.

For solutions using the Epsilon platform hosted in Epsilon’s cloud, this document is absent since the cloud infrastructure is solely managed and operated by Epsilon.

**Change Defaults**

TBD

Epsilon Environment Mappings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type of Storage | Environment | Epsilon Environment | External Host | Epsilon Host |
| long-term, archival purpose | customer-non-prod | dev | dev.api.example.com | Epsilon-router.example.internal:8443 |
| QA | customer-non-prod | qa | qa.api.example.com | Epsilon-router.example.internal:8443 |
| Prod | customer-prod | prod | api.example.com | Epsilon-router.example.internal:8443 |

Epsilon Virtual Host Definitions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Environment | Virtual Host Name | Host Alias | Port | SSL (Y or N) |
| Dev | default | dev.api.example.com | 8443 | Y |
| QA | default | qa.api.example.com | 8443 | Y |
| Prod | default | api.example.com | 8443 | Y |

High Availability and Failover

Lewisville Datacenter Scenario for Acceptance / Stage

The development environment will be setup as a single data center as disaster recovery is not required in this environment. Also, only specific components will be setup for high availability as identified by component to save on the hardware requirements. In a Lewisville datacenter setup, there will be multiple instances of Forwarder agent, Indexer, Search, Storage and web UI for visualization. As long as there is one instance of this combination running, live log traffic will not be impacted. If one of the Server goes down then server will fail over and teams still can do the current process. If the web UI goes down then Developers cannot use the visualization. As such these two components are considered non critical for log monitoring.

Multi Datacenter– Lewisville Production

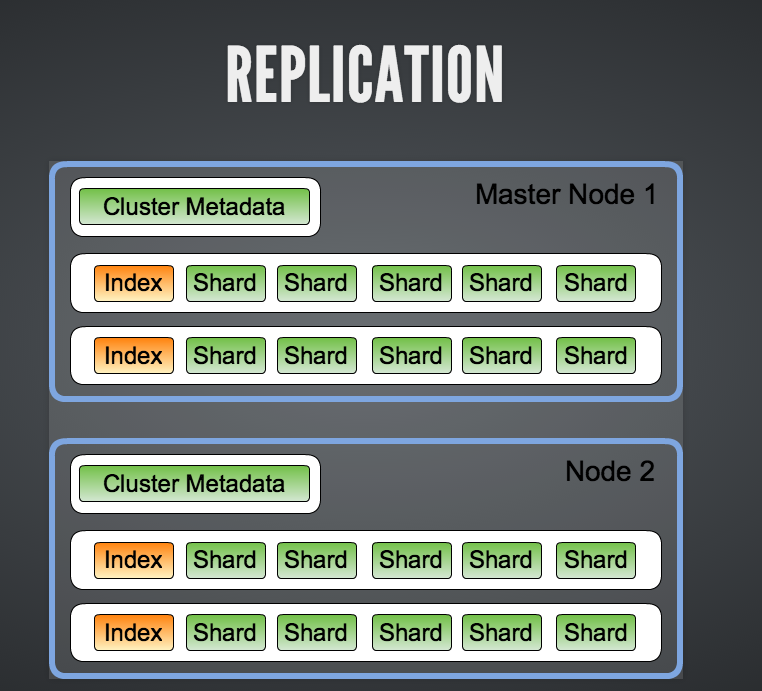
Harmony production environment will be setup within two data center to enable high availability and recovery from failures when the primary data center fails due to a disaster. The two data centers will work in an active / active configuration. In case of a failure due to a disaster, the DNS switch will route all traffic to the second data center and we will start collecting logs in this data center. The following diagrams will indicate how failover DCs will work with respect to log management.

Diagram is yet to be updated with the latest log management solution



Size of your heap – 32G is max. 700K per second need 50 nodes

Fields grow, size grows. Use bulk loading.



**Hardware Requirements:**

* All servers with logs will have agent installed either Shipper / Forwarder to central Broker server.
* Broker / Indexer server has Redis running. For fail over, we need 2 nodes to start. This is a CPU bound machine. They will have quite a few grok filters that ask for a lot of CPU power. These are all filters that skim the logs for necessary inputs. Two VM’s with 2x24x100 (local) - Broker & Indexer ( Redis & Logstash)
* 4 ES nodes, each having 2x24x400 (NFS) - ElasticSearch & Kibana. This is memory bound machine it does high size of field cache. CPU should be ok, need to keep a check on the cluster.
* Due to RAM limitation, we should keep logs for a week. Each day logs are about 14GB in compressed form (text files). Indices created for this 14GB of compressed text file is around 200GB (This can vary drastically depending on the mapping you've set for your index).
* Current setup indexes around 8k messages per second in peak hours. But, if there is some issue we need to restart ES, this increases the Redis list length. After that, we can see this setup indexing around 18k messages per second.

Go-Live Check List

This is yet to be identified (TBD)

Additional Information (AS NEEDED)

Monitoring Log Management instance <https://github.com/royrusso/elasticsearch-HQ> or <http://bigdesk.org/>

**Appendix**

Key Concepts and Terminology

|  |  |
| --- | --- |
| Term/Concept | Description |
| Elastic Search | Json Document oriented search engine |
| Elastic Search Nomenclature | An index is a table, a document is a table row and a field is a table column, schema’s is called Mapping. Indexes are stored in Lucene instances called "shards". (Primary and Replica Shards). Indexing in Elasticsearch corresponds to create and update in CRUD |
| Elasticsearch client nodes | Nodes are essentially smart load balancers that are part of the cluster |
| Shard | A shard in Elasticsearch refers to a Lucene index. Elasticsearch by default uses five shards for each Elasticsearch index. Document is stored in one shard, then replica to other. |
| Facets | Provides aggregated statistics of a query |
| Logstash Forwarder | this small tool, developed in Go, allows to securely ship compressed logs data with minimal resources usage, using the Lumberjack protocol. |
| Redis | LogStash sends events to and receive events from multiple Redis instances in a failover configuration.  *It's important to note that this is a failover rather than true high availability*. |
| Broker Load balancing | Events are not "round robin'ed or load balanced between Redis instances. LogStash will try to connect to a Redis instance and send events. If that send succeeds then it will continue to send events to that Redis instance. If the send fails then LogStash will select the next Redis instance and try to send to it instead. |

Supporting Documents

|  |  |  |
| --- | --- | --- |
| Document Title | Description | Location |
| Infrastructure Architecture | This document captures the Epsilon platform infrastructure, including all environment specifications. | [Provide location of the document] |
| Filters – Config |  |  |
| Deployment Plan |  |  |
|  |  |  |