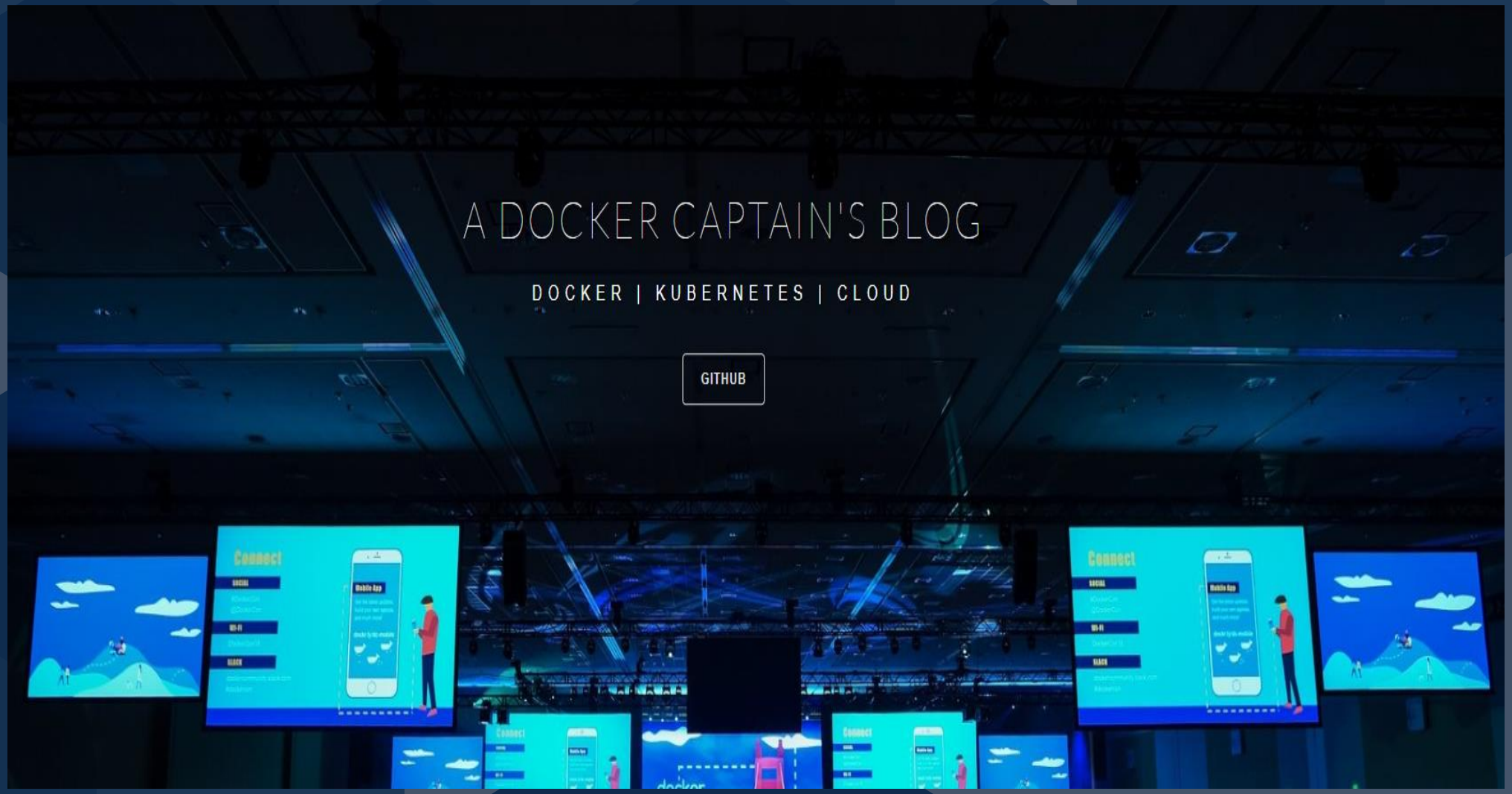
OpenUSM – A Modern Approach to Server Management, Log Analytics & Machine Learning using Docker & Redfish

Docker Bangalore Meetup

Ajeet Singh Raina
Twitter: @ajeetsraina
GitHub: ajeetraina

- Principal Development Engineer at DellEMC
- 1st half of my career was in CGI & VMware
- 2nd half of my career has been in System Integration Testing
- Docker Captain (since 2016)
- Docker Bangalore Meetup Organizer (8000+ Registered Users)
- DockerLabs Incubator ~ 700+ Slack Members

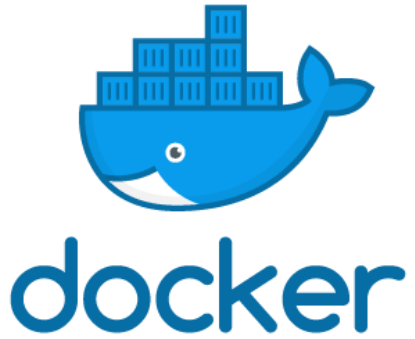
$curl www.collabnix.com



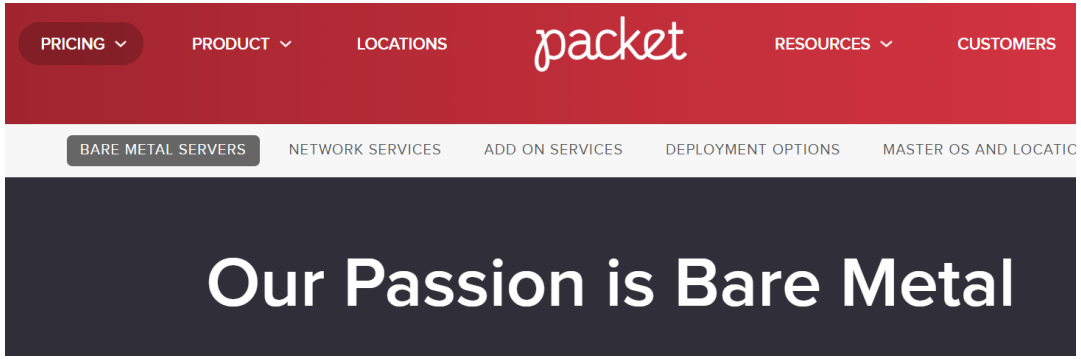
dockerlabs.collabnix.com


packet


openusm


docker

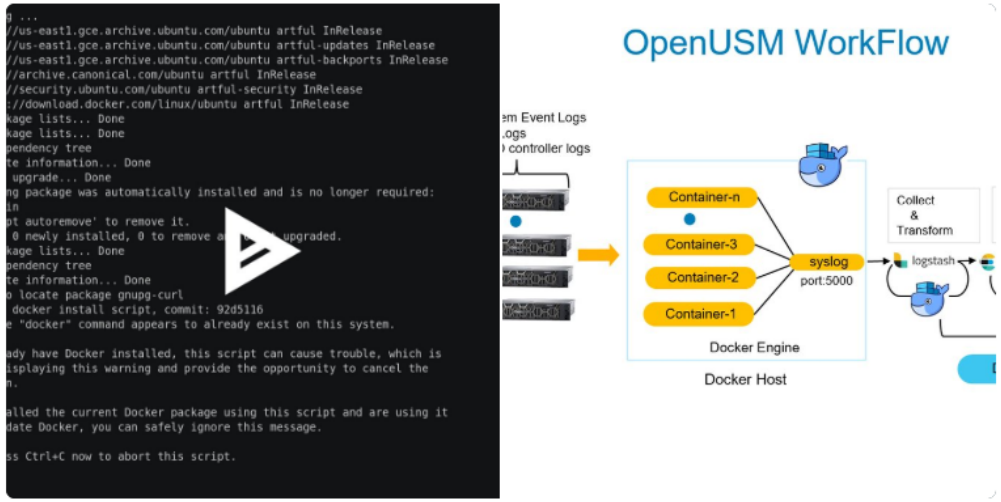

Stack



Packet
@packethost

Following

This ain't for everyone, but for those of us with a lot of Dell servers it's the cat's meow: check out OpenUSM, which uses Docker, Redfish & an ELK stack for server management & logs analytics baremet.al/2weRFiL



9:35 AM - 26 Aug 2018

7 Retweets **24** Likes

openusm

Agenda

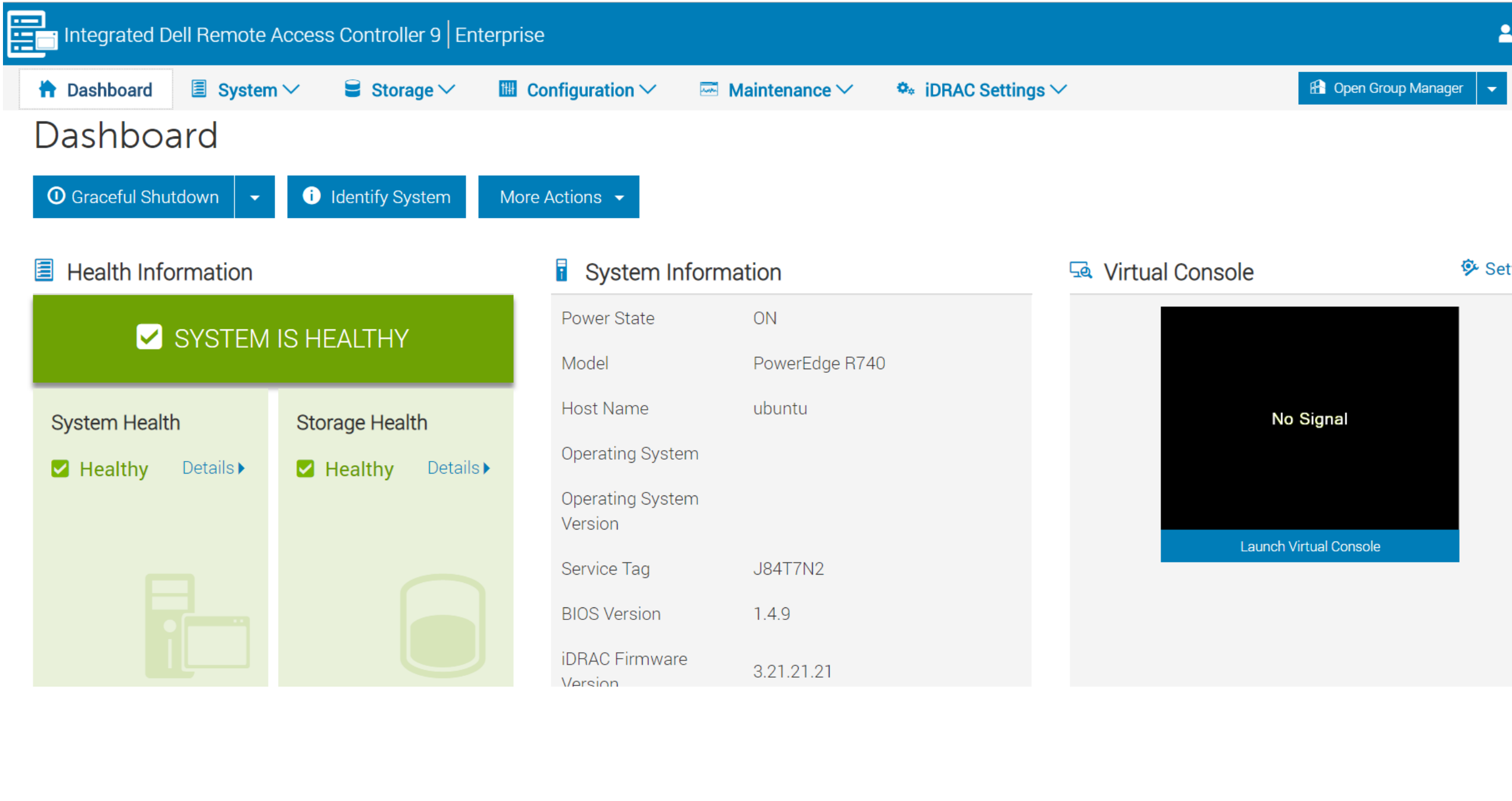
- Out-Of-Band System Management & Redfish
- Introducing OpenUSM
- Why OpenUSM?
- Technology Stack
- Value Proposition
- Logs Analytics & Recommendation Tool
- Demo

What is Out-Of-Band Management?

- Server Management independent of the operating system
- Provided by an embedded chip, has its own Ethernet port, usually connected to a separate management network
- Many names: Dell iDRAC, HP iLOM, Lenovo IMM, BMC

Management Capabilities includes:

- Device Inventory
- Hardware Failure Detection
- BIOS Configuration
- Firmware Inventory


Integrated Dell Remote Access Controller 9 | Enterprise
Dashboard
System
Storage
Configuration
Maintenance
iDRAC Settings
Open Group Manager
Dashboard
Graceful Shutdown
Identify System
More Actions
Health Information
SYSTEM IS HEALTHY
System Health
Healthy
Details
Storage Health
Healthy
Details
System Information
Power State
ON
Model
PowerEdge R740
Host Name
ubuntu
Operating System
Operating System Version
Service Tag
J84T7N2
BIOS Version
1.4.9
iDRAC Firmware Version
3.21.21.21
Virtual Console
Set
No Signal
Launch Virtual Console

Integrated Dell Remote Access Controller 9 | Enterprise

Dashboard | System | Storage | Configuration | Maintenance | iDRAC Settings

Maintenance

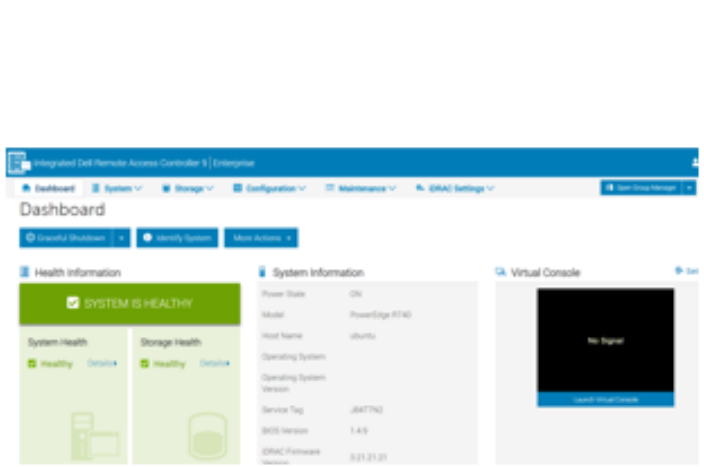
Lifecycle Log | Job Queue | System Update | System Event Log | Troubleshooting | Diagnostics | SupportAssist

System Event Log

Instructions: The System Event Log contains information about the managed system. To sort the log by column, click a column header.

Severity	Description
✓	The process of installing an operating system or hypervisor is successfully completed.
✓	The process of installing an operating system or hypervisor is started and is in progress.
✓	The chassis is closed while the power is off.
✗	The chassis is open while the power is off.
✓	Log cleared.

A Simple OOB Management



Challenges

- IPMI is very old protocol(15 years old)
- Non-API friendly
- Scripting against IPMI is done by line-by-line bash scripts
- Non-Scale-Out Architecture
- Insecure
- Non-Human Readable Information(low level interfaces & byte-oriented messages)

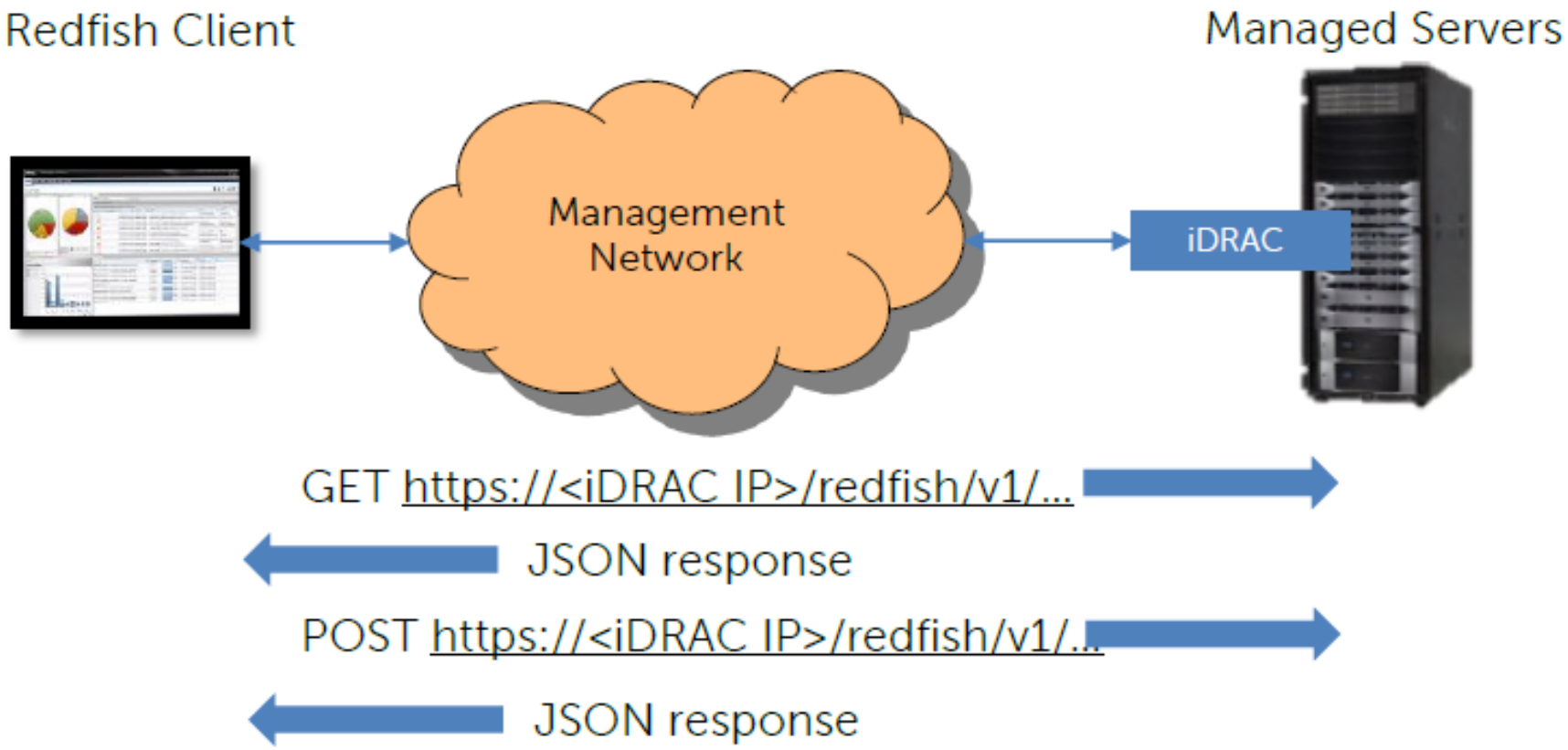
Enter… Redfish

- ✓ Open Source, Open Industry standard specification published by DMTF for Hardware Management.
- ✓ Provides a RESTful API used to obtain information about servers or control them through an OOB controller.
- ✓ Built on a modern too-chain which includes HTTPs and JSON.
- ✓ Easier to use and more secure than legacy specifications such as IPMI and WSMAN.
- ✓ A Redfish request is sent as an URI, so a client could be any application on a sever, workstation or mobile device.

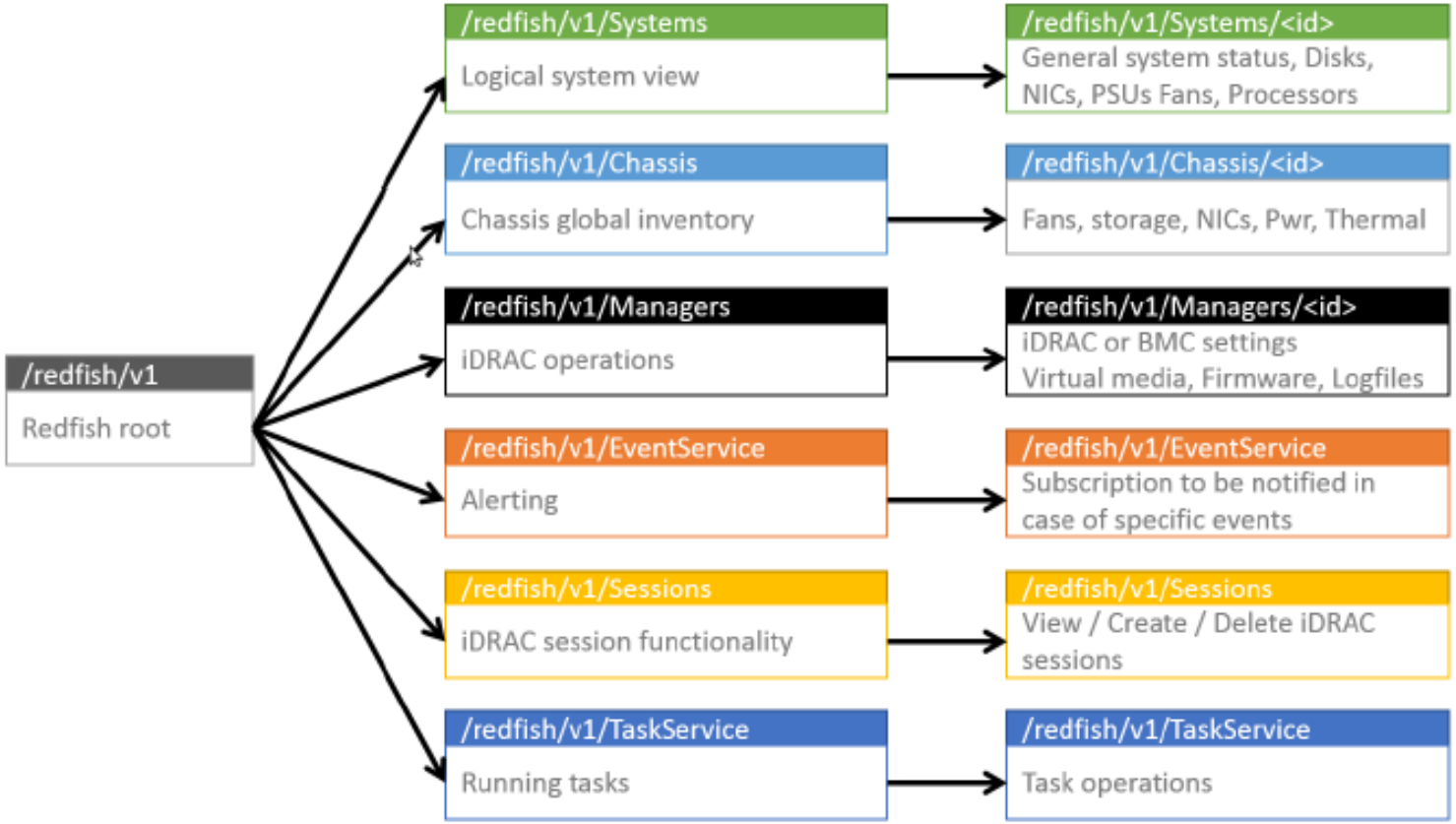
What can we do with Redfish?

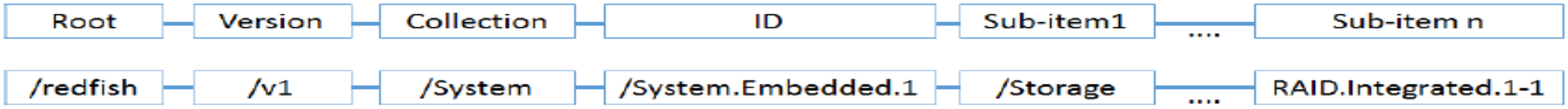
- ✓ Retrieve server Health status
- ✓ Retrieve Hardware and Firmware Inventory
- ✓ Power Up, Power Down, Warm Boot, Cold Boot
- ✓ Change BIOS Settings.
- ✓ Configure OOB controller (i.e. users, network settings)
- ✓ Configure Hardware RAID
- ✓ Firmware Updates

Redfish Operational Model



Redfish API tree structure





Examples

1. Checking system health

```
[root@localhost ~]# curl -s https://10.94.214.168/redfish/v1/Systems/System.Embedded.1 -k \
> -u root:calvin | python -m json.tool | jq .Status
{
  "Health": "OK",
  "HealthRollUp": "OK",
  "State": "Enabled"
}
```

2. Checking the RAID controller

```
[root@localhost ~]# curl -s -k -u root:calvin \
> https://10.94.214.168/redfish/v1/Systems/System.Embedded.1/Storage/Controllers/RAID.Embedded.1-1 \
> | python -m json.tool | jq .Name
"PERC S130 Controller"
```

3. Getting power metrics for last hour

```
[root@localhost ~]# curl -k -s -u root:calvin \
> https://10.94.214.168/redfish/v1/Chassis/System.Embedded.1/Power/PowerControl \
> | python -m json.tool | jq .PowerMetrics
{
  "AverageConsumedWatts": 184,
  "IntervalInMin": 60,
  "MaxConsumedWatts": 186,
  "MinConsumedWatts": 184
}
```

4. Checking and setting session timeout

```
[root@localhost ~]# python3 Redfishtool/redfishtool.py -r 10.94.214.168 -u root -p calvin -S Always \
> SessionService | python -m json.tool | jq .SessionTimeout
9999
```

```
[root@localhost ~]# python3 Redfishtool/redfishtool.py -r 10.94.214.168 -u root -p calvin -S Always \
> SessionService setSessionTimeout 10000
{
    "SessionTimeout": 10000
}
```

A Typical Swiss Army Knife(Analogy)

A Multi-tool Device

Open Universal Systems Manager

System Monitoring

Firmware Update

Sensor Analytics

Machine Learning

BIOS Token Change

Insight Log Analytics
(LC, SEL, Event Logs)

openusm

What is OpenUSM?

- OpenUSM is a suite of open source tools & scripts which purely uses Redfish API to perform Server Management tasks, Monitoring & Insight Log Analytics.
- It follows “Container Per Server” (CPS) model.
- It is an out-of-band system management solution purely based on Redfish API Interface.
- 100% container-based solution which heavily uses Docker & Docker Compose for building Microservices for Monitoring & Logging Analytics.
- It is a platform agnostic solution(can be run from laptop, server or cloud) and works on any of Linux or Windows platform with Docker Engine running on top of it

OpenUSM’s Value Proposition

- Easy Deployment Model(Docker containers just make it simple)
- Integrates well with near real-time search analytics tools like ELK stack
- Simplifies Sensor Log Analytics & visualization using Grafana tool
- Can scale both vertically & horizontally
- It can be built and customized by anyone based on the needs and holds a plug-and-play components and functionalities.

OpenUSM Technology Overview

Monitoring

SYSTEM MANAGEMENT

Insight Log Analytics

Sensors GPU

Prometheus

Grafana

Alert Manager

Universal Systems Manager

CLI

GUI

Django

Golang

Redfish API

Python

Requests

Docker Compose

Commodity supported Docker Engine

On-Premise

Cloud

Sensors LC SEL

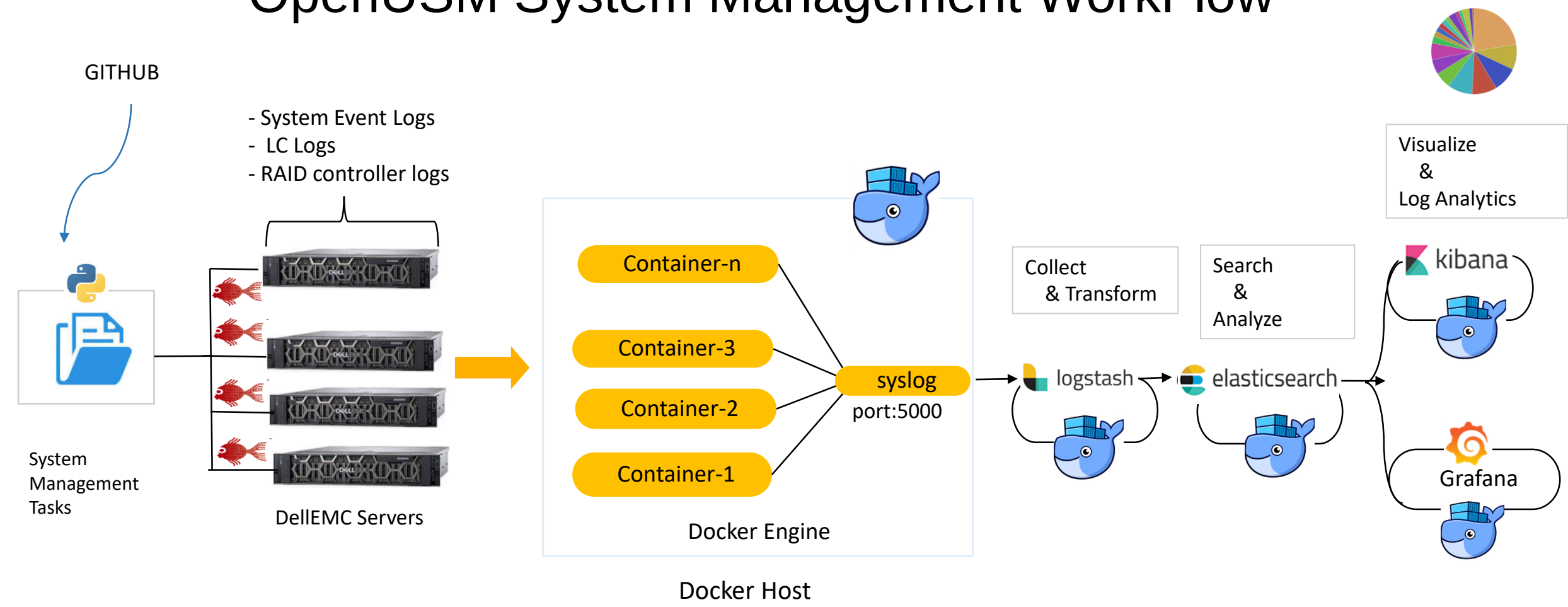
logstash

elasticsearch

kibana

openusm

OpenUSM System Management WorkFlow



openusm

Overview of ELK

ELK Stack is a combination of 3 open source tools which forms a log management platform, that helps in deep searching, analyzing & visualizing the log generated from one or different machines

Search & Analytics

NoSQL Database

Based on Apache Lucene

Uses Index to search which makes it easier

Data pipeline tool

Centralizes the data processing

Collects, parses & analyzes large variety of structured/unstructured data

Provides plugins to connect to various types of input sources & platforms

Visualization tool

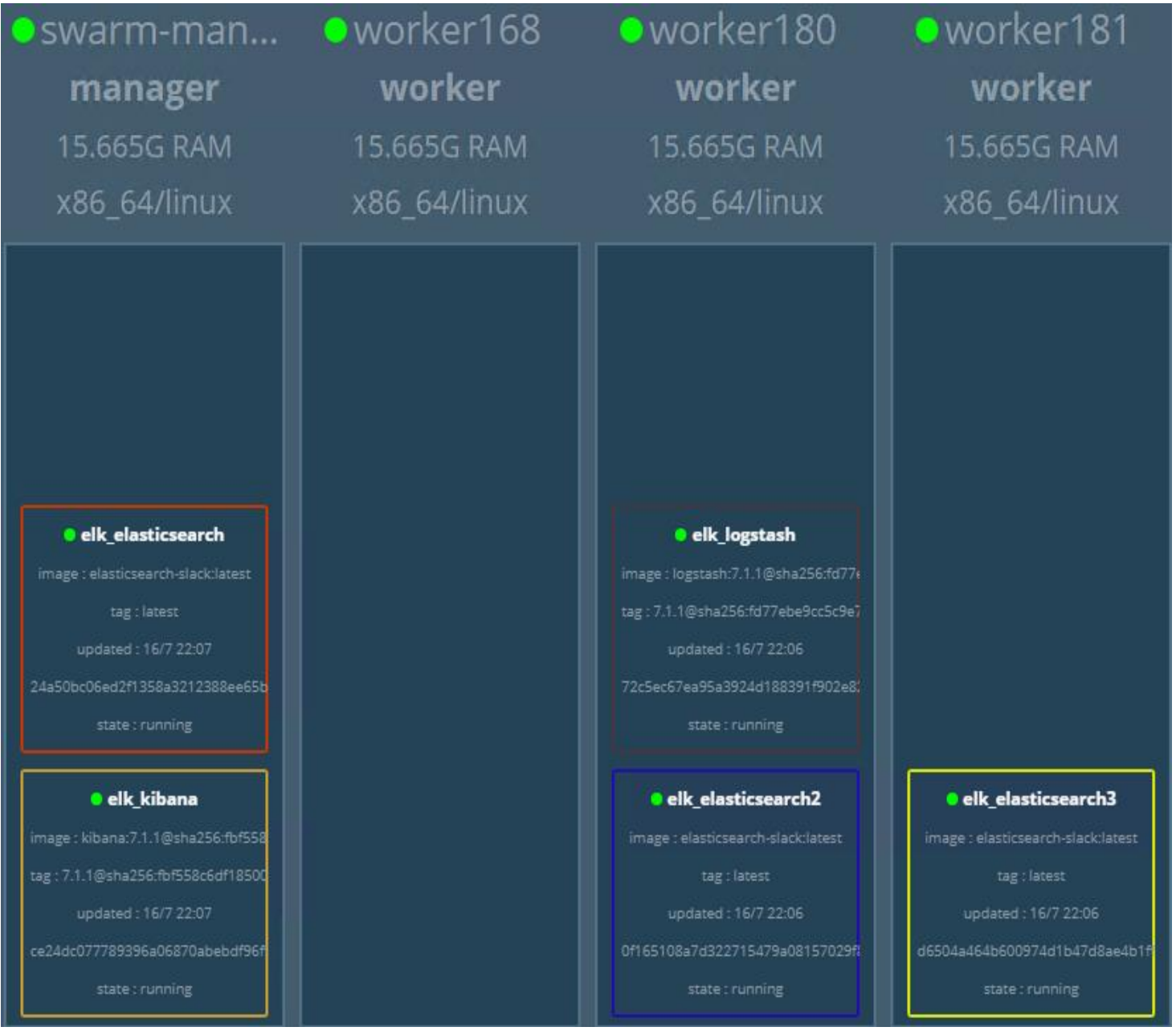
Provides real time analysis, summarization, charting & debugging

Provides user friendly interface

Allows sharing of snapshots of the logs searched for

Why we chose ELK?

- ELK is known for its speed and full-text capability
- High Performance indexing over 150GB/hour on all our modern hardware
- It requires smaller RAM – only 1 MB heap
- A Powerful, accurate and efficient search algorithm
- Easy to setup with a single Docker Compose file
- Powerful query types: phrase queries, wildcard queries, range queries and more fielded searching (e.g. title, author, contents)
- Integrated well with Prometheus which is popular for real-time monitoring



Running Elastic Stack on Docker Swarm

openusm

Its Demo Time

Pushing Hardware Logs to ELK Stack

Insight HW Log Analytics (LC, SEL & Sensor Logs)

Insight Log Metrics (Sensor Logs) via Grafana

Its Demo Time

Pushing Hardware Logs to ELK Stack

Insight HW Log Analytics (LC, SEL & Sensor Logs)

Insight Log Metrics (Sensor Logs) via Grafana

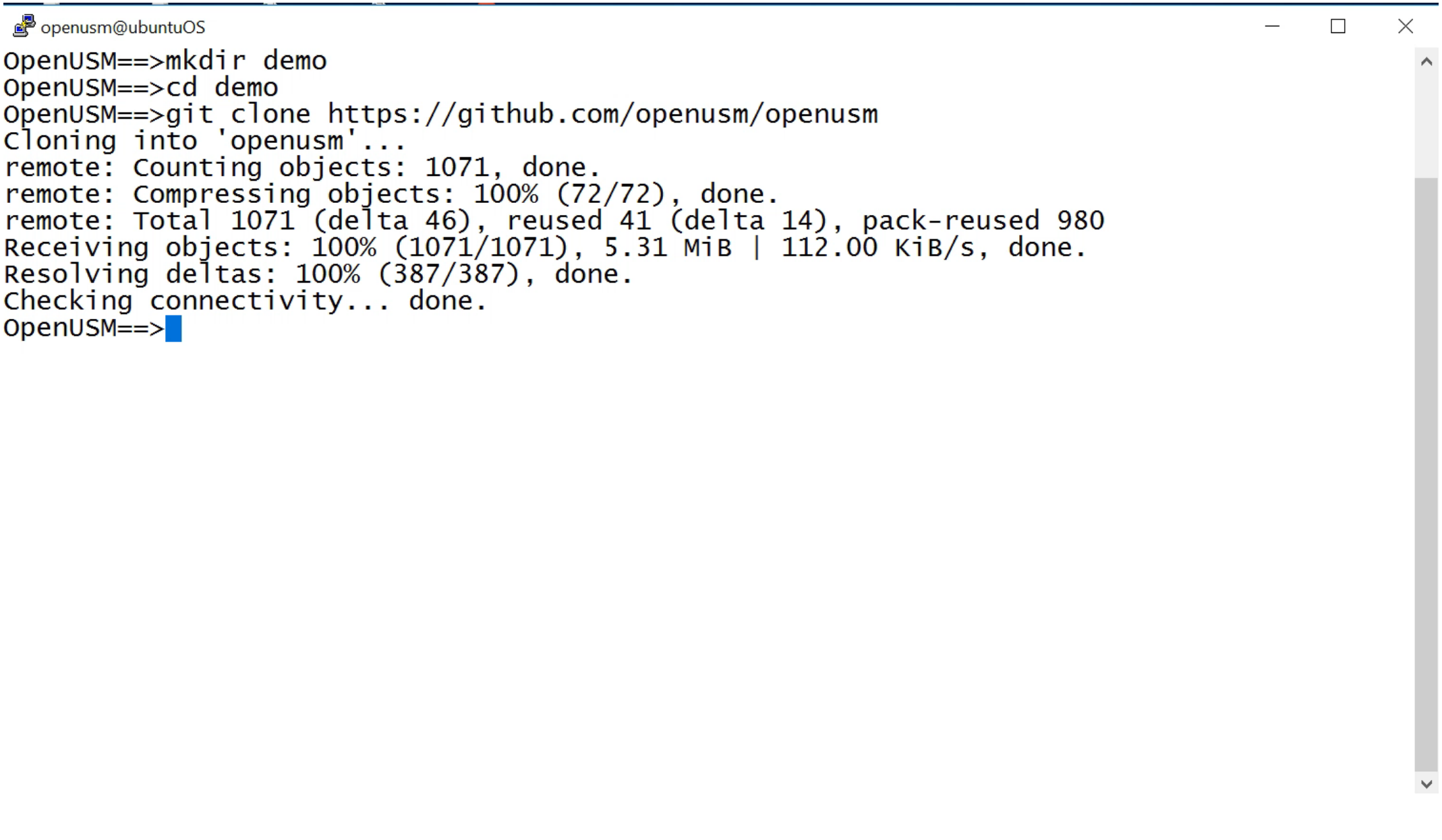

openusm@ubuntuOS: ~
OpenUSM==>

Its Demo Time

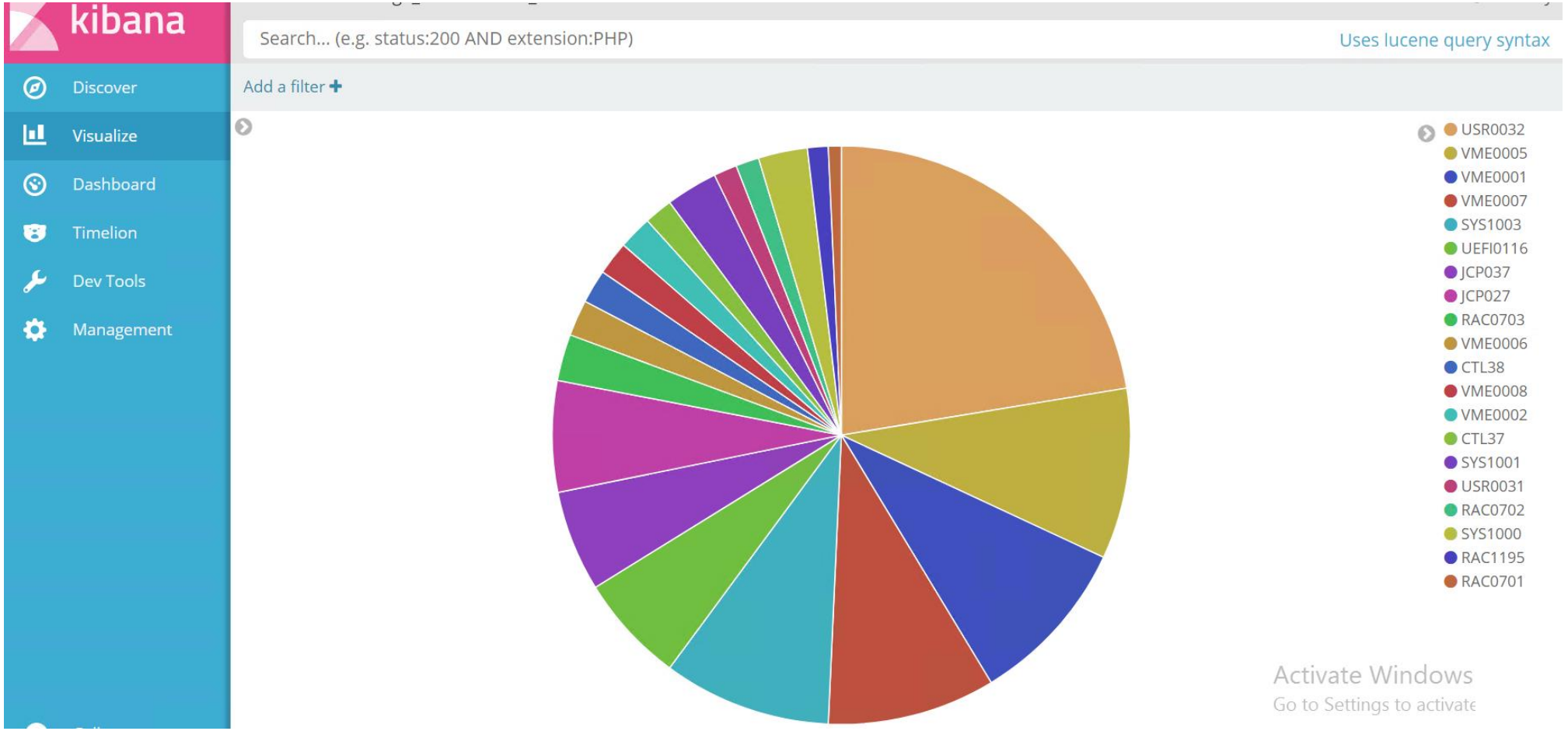
Pushing Hardware Logs to ELK Stack

Insight HW Log Analytics (LC, SEL & Sensor Logs)

Insight Log Metrics (Sensor Logs) via Grafana


openusm@ubuntuOS
OpenUSM==>mkdir demo
OpenUSM==>cd demo
OpenUSM==>git clone https://github.com/openusm/openusm
Cloning into 'openusm'...
remote: Counting objects: 1071, done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 1071 (delta 46), reused 41 (delta 14), pack-reused 980
Receiving objects: 100% (1071/1071), 5.31 MiB | 112.00 KiB/s, done.
Resolving deltas: 100% (387/387), done.
Checking connectivity... done.
OpenUSM==>

Visualizing LC Logs Message ID for the last 1 year (using Kibana)



Its Demo Time

Pushing Hardware Logs to ELK Stack

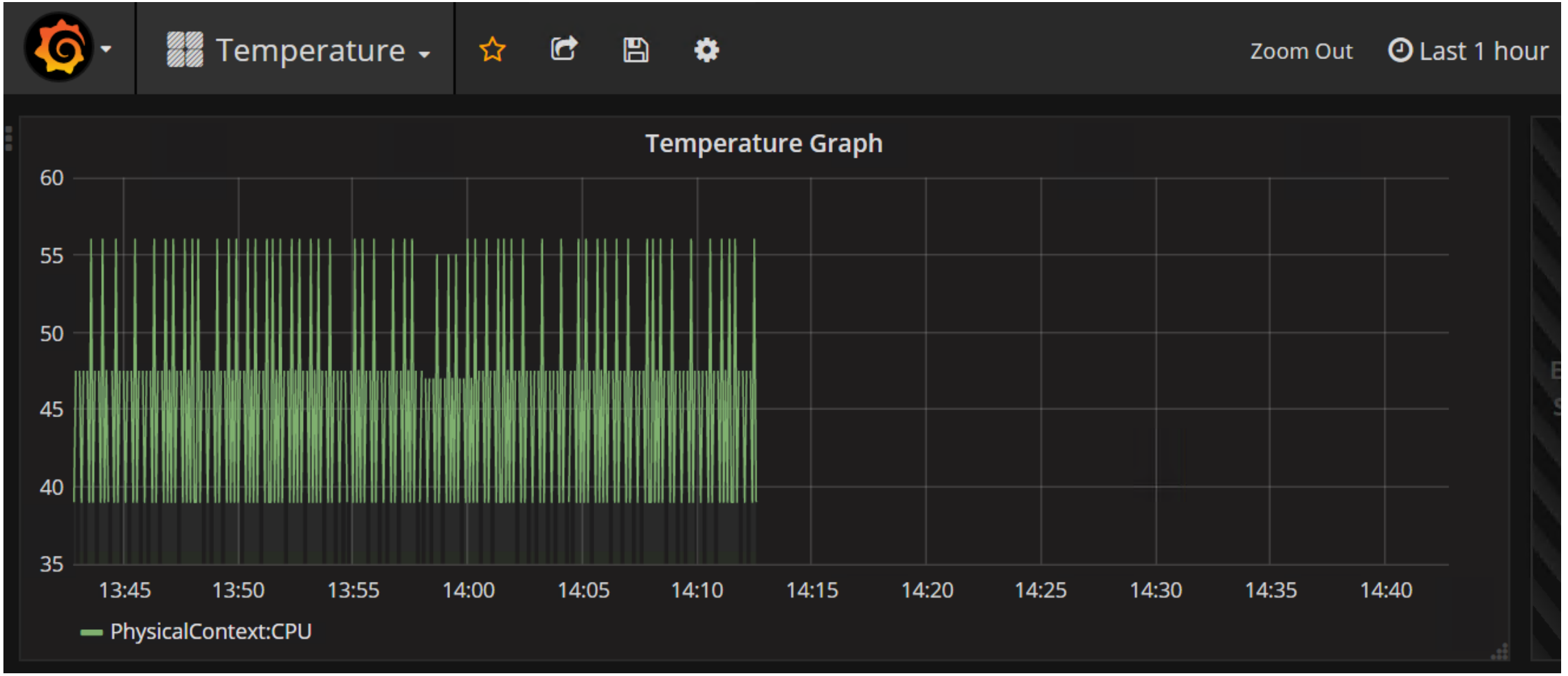
Insight HW Log Analytics (LC, SEL & Sensor Logs)

Insight Log Metrics (Sensor Logs) via Grafana

openusm@ubuntuOS

```
OpenUSM==>pwd
/root/demo/openusm/logging
OpenUSM==>
```

Visualizing Sensors Logs for last 1 hour (using Grafana)

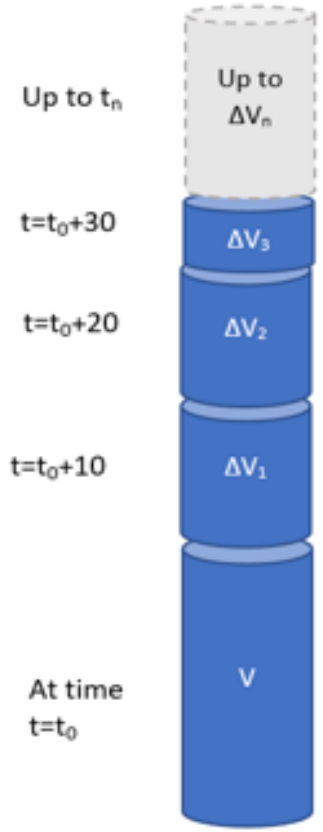


Developing a Recommendation Tool for Optimizing Operational Workloads for Systems

An Approach

- Tracking machines using their OOB Management IP Addresses.
- Collecting logs in every 10 minutes and pushing them to Elasticsearch database.
- Logs being segregated based on the Message IDs or Error Codes which help to analyze the current condition and working of the system under consideration.
- Generating alerts incase of undesirable situations.
- Automated Workflow

Log Accumulation & Analysis



Logs get accumulated in Elasticsearch every 10 minutes. For every server, we have a separate dedicated index which stores its lifecycle logs.

Naming of the index is done as such:
For a server with IDRAC IP as 100.98.26.xx, index would be named as index100.98.26.xx

When the script runs for the first time, Volume V logs get collected in the index.
And then in every 10 minutes, we get some ΔV_n logs over the old logs.

Thus, our indices keep building up.

Challenges

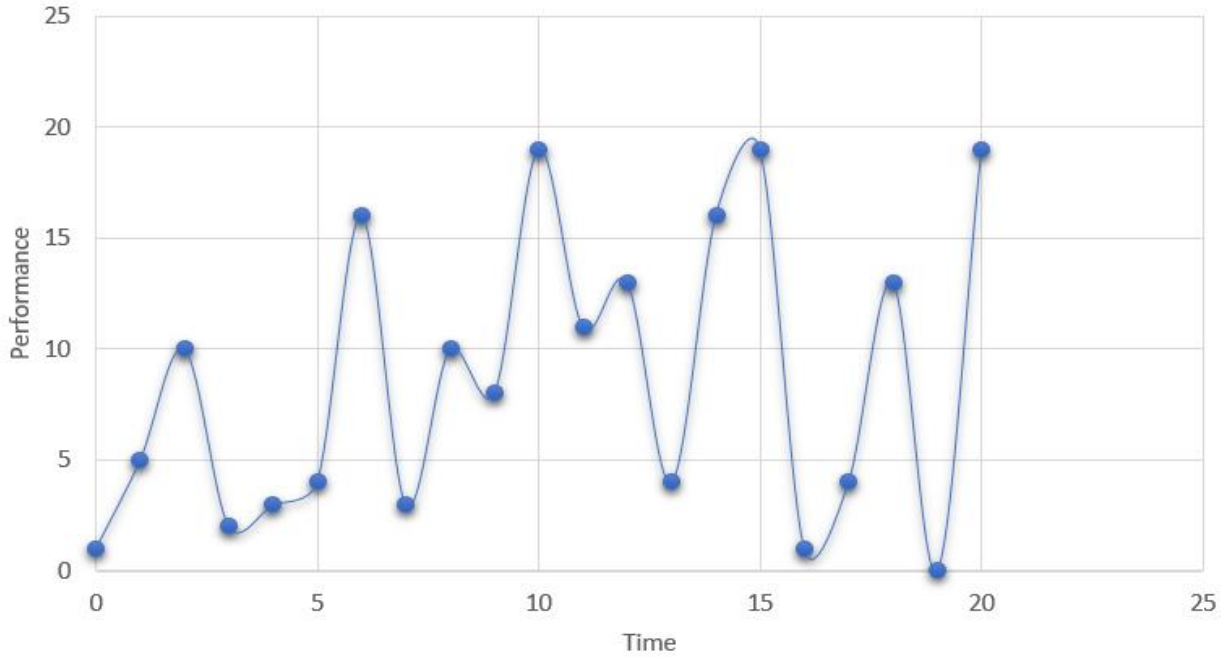
- Getting bunch & bunch of logs to conduct analysis.
- Simulation of an overutilized system.
- Bulk rejections from Elasticsearch Master Node.
- Finding a suitable alerting mechanism.

Challenges

- **Getting bunch & bunch of logs to conduct analysis.**
- Simulation of an overutilized system.
- Bulk rejections from Elasticsearch Master Node.
- Finding a suitable alerting mechanism.

Solution

- We thought of stressing the system so that it starts producing logs.
- **Idea:** Simulate such a scenario in which we get variations in CPU and Memory usage and Power Consumption.



Challenges

- Getting bunch & bunch of logs to conduct analysis.
- **Simulation of an overutilized system.**
- Bulk rejections from Elasticsearch Master Node.
- Finding a suitable alerting mechanism.

Simulating Overutilization of System Hardware

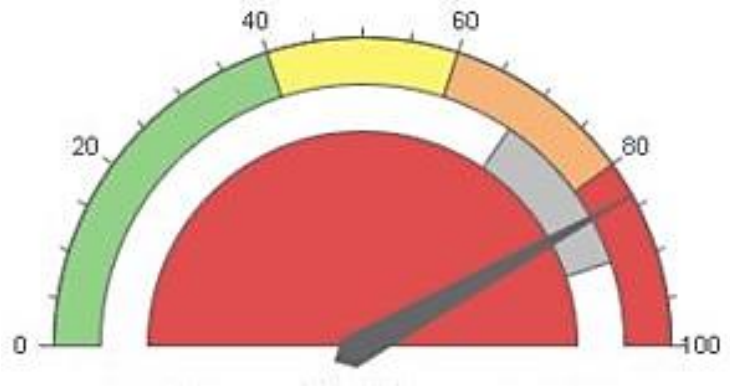
- Any server can undergo stress or a downtime due to numerous reasons. Some of these are stated as follows:

- Overloaded processor.
- Application demand higher than available resources.
- Inadequate power supplies.
- Thermal issues

Every issue which can lead to a system downtime generates logs which get collected in IDRAC logs under Life Cycle Logs or System Event Logs.

Tools identified for Stressing the system:

- Linpack
- HPL
- Intel Power Thermal Utility



Challenges

- Getting bunch & bunch of logs to conduct analysis.
- Simulation of overutilized system using an open source tool (HPL Tool).
- **Bulk rejections from Elasticsearch Master Node.**
- Finding a suitable alerting mechanism.

Solution

- Deployed a three-node Elasticsearch cluster over Docker Swarm.
- Made a dedicated Elasticsearch master for handling the requests.
- Increased timeout period of Elasticsearch connection.
- Increased number of retries.

Challenges

- Getting bunch & bunch of logs to conduct analysis.
- Simulation of overutilized system using an open source tool (HPL Tool).
- Bulk rejections from Elasticsearch Master Node.
- **Finding a suitable alerting mechanism.**

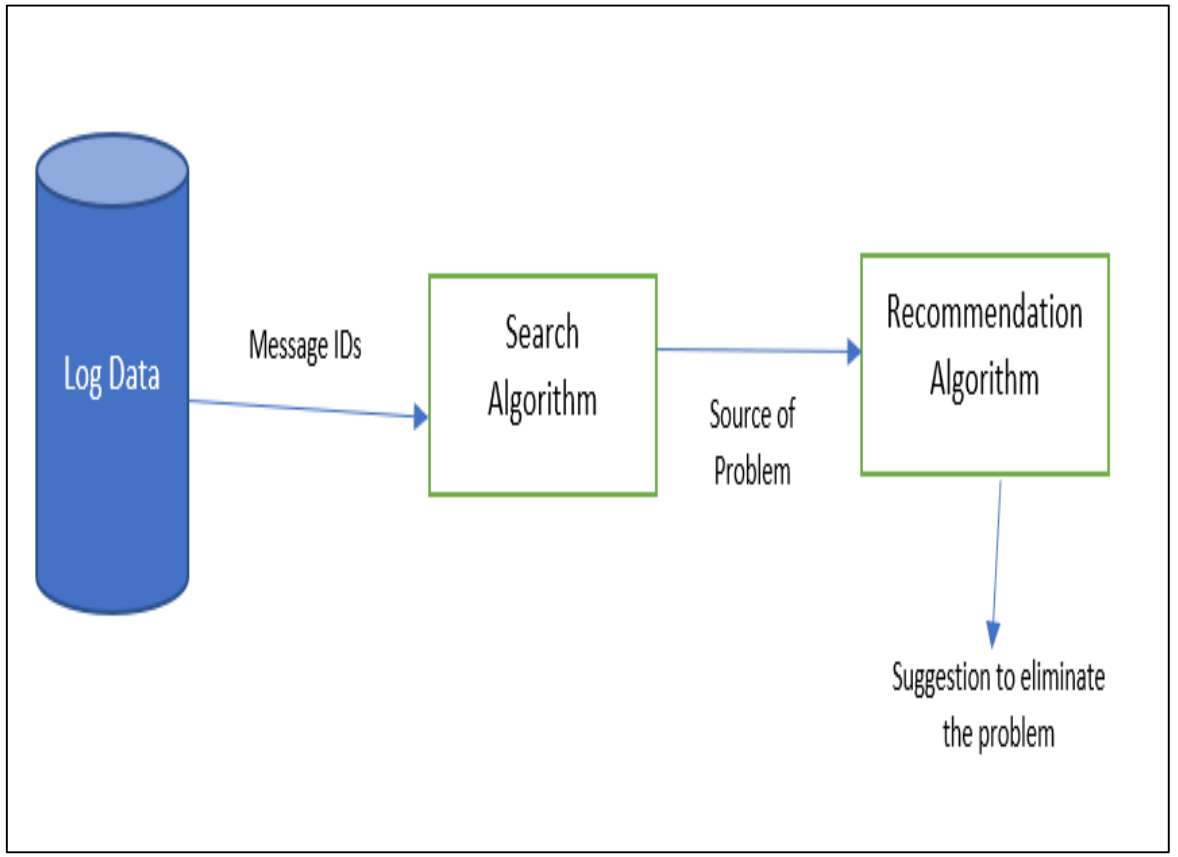
Solution

- Integrating the developed workflow with Prometheus and used its inbuilt mechanism of alerting via Slack.
  - Time delays in alerting.
  - Complex to set time-based monitors.

- Configured ELK Watchers and integrated them with Slack API.

Analysis Workflow

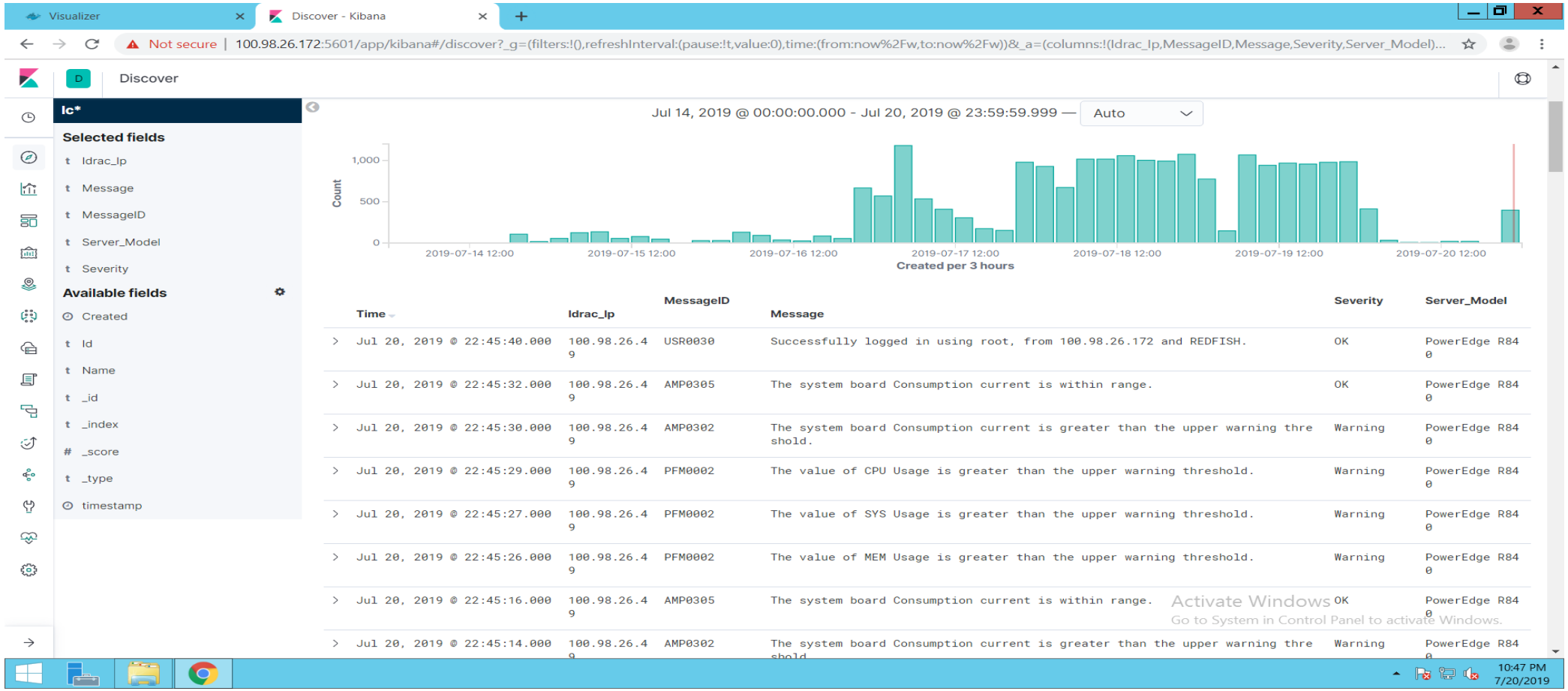
- The Search Algorithm would be fed with the newly generated logs and it would find the Message ID which is frequently generated.
- A priority scheme will be used to segregate the Message IDs and finding the source of problem.
- Most frequently occurring log message with the highest priority will be treated as the main cause of problem.
- This can be fed to our recommendation system which can suggest what steps to take next to resolve the error and prevent the system from crashing.



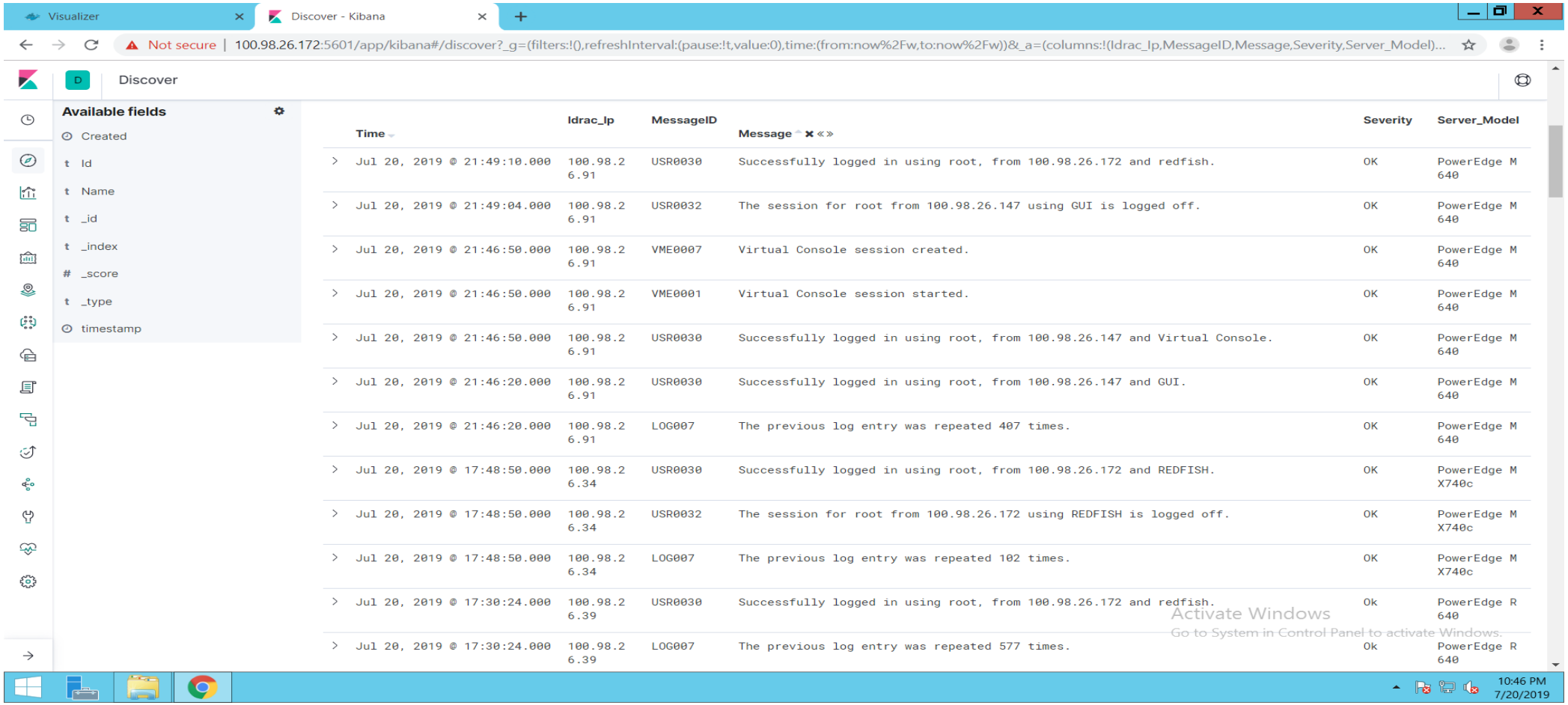
```
root@swarm-manager:~/server-management# python search_script.py

For Server:  100.98.26.49


Error Codes      Occurrence        Percentage
------------------------------------------------

PFM0004          3                   30.0 %
AMP0302          2                   20.0 %
AMP0305          2                   20.0 %
USR0030          1                   10.0 %
PFM0002          2                   20.0 %

Error Types      Occurrence
-------------------------

AMP              4
PFM              5
USR              1
root@swarm-manager:~/server-management#
```

Search Algorithm Working

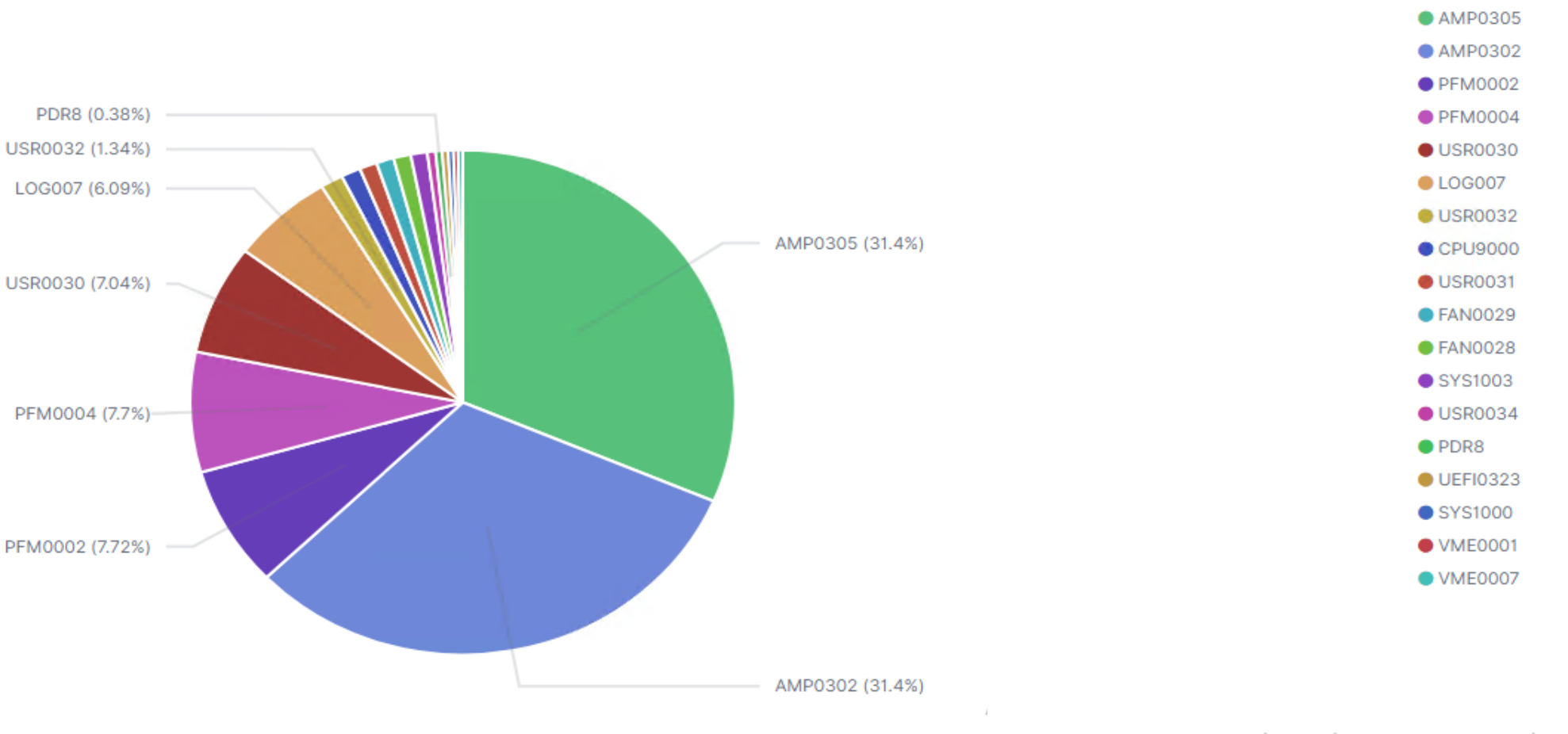
Results – Periodic Log Accumulation



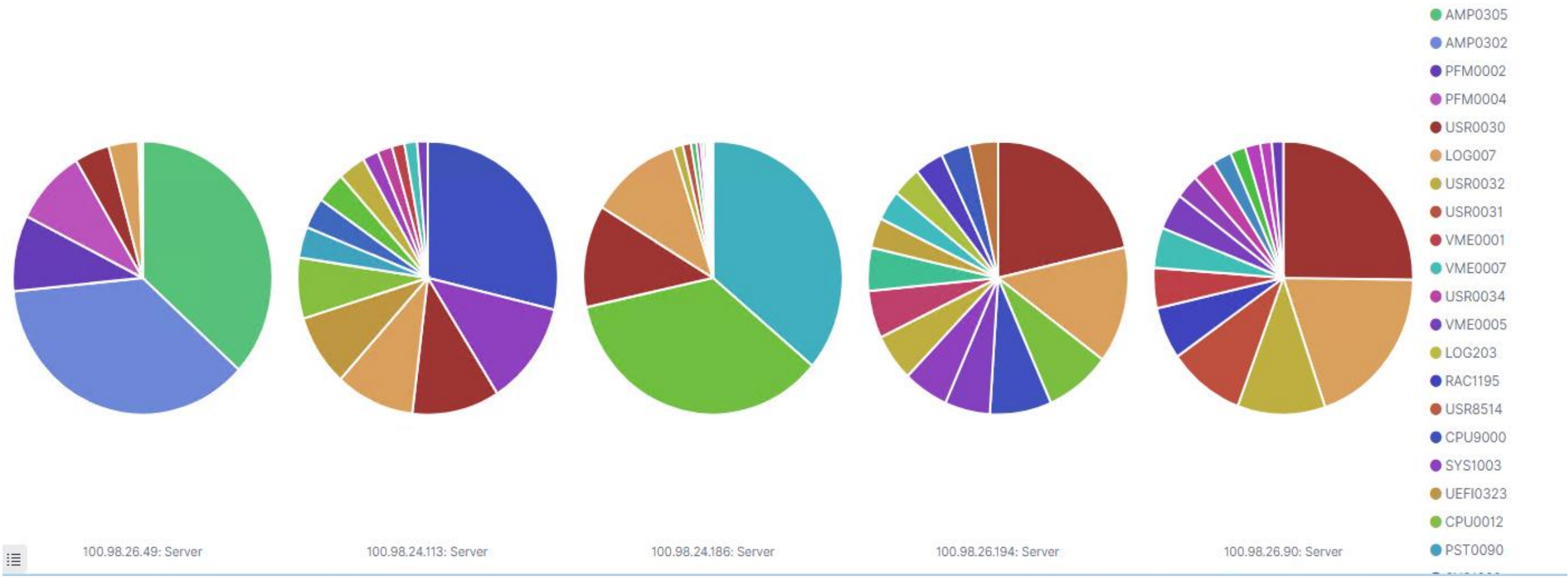
Results – Periodic Log Accumulation

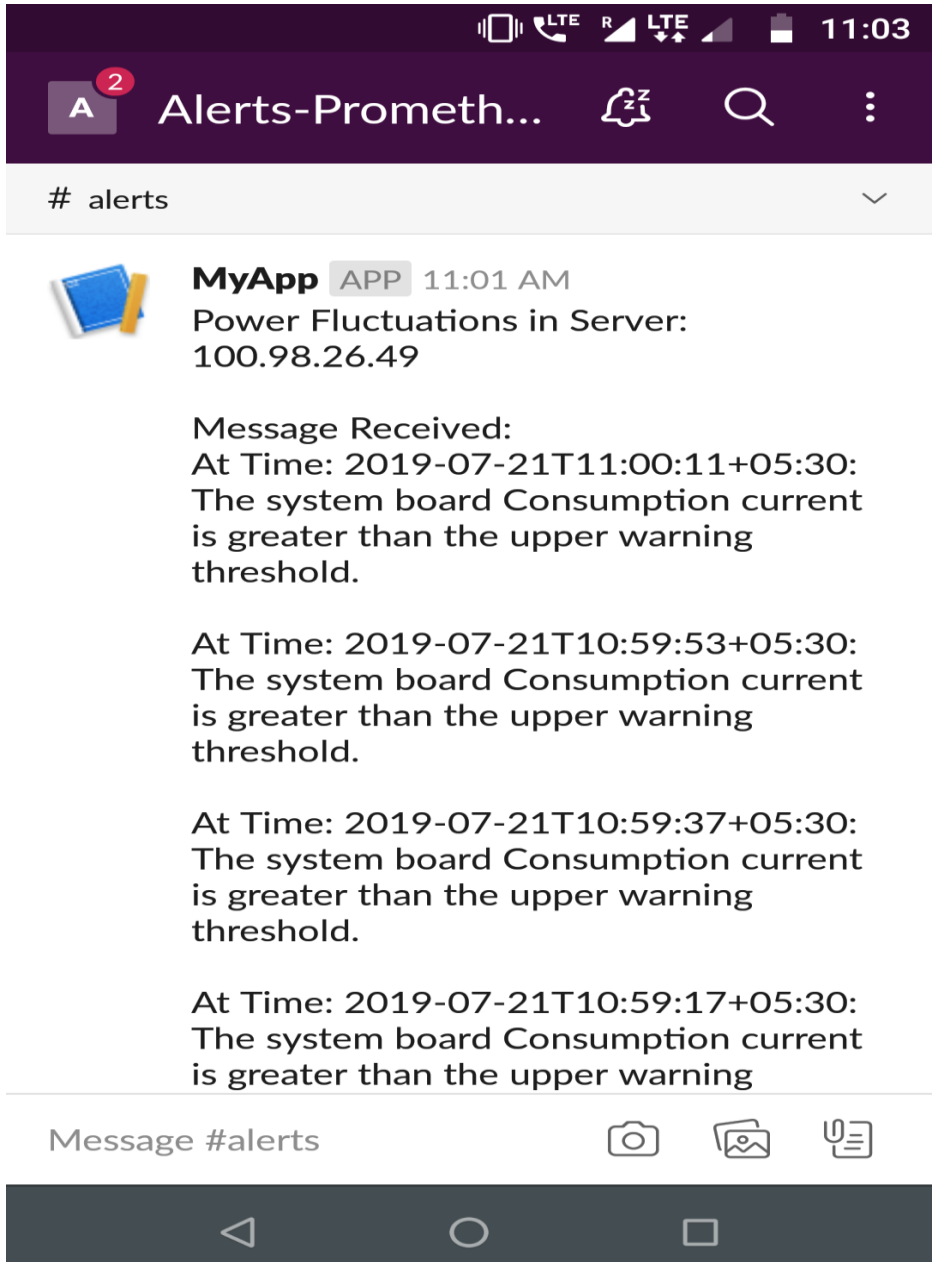


Results – Kibana Visualizations

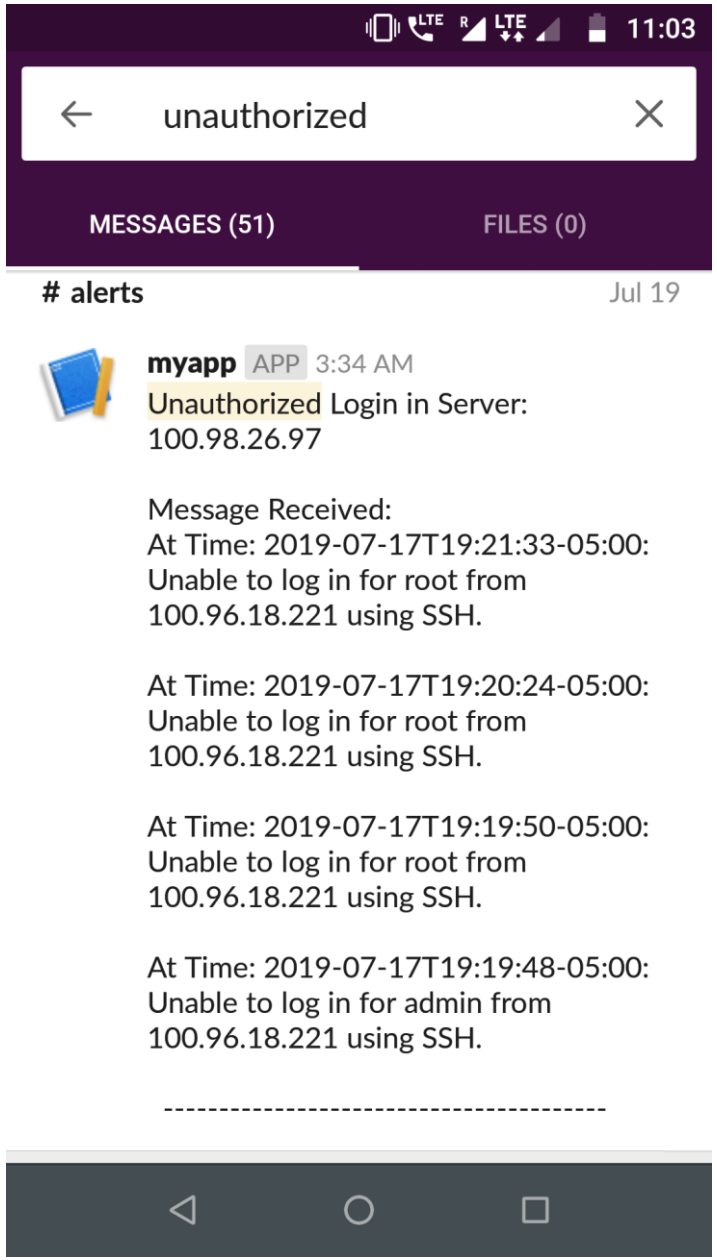


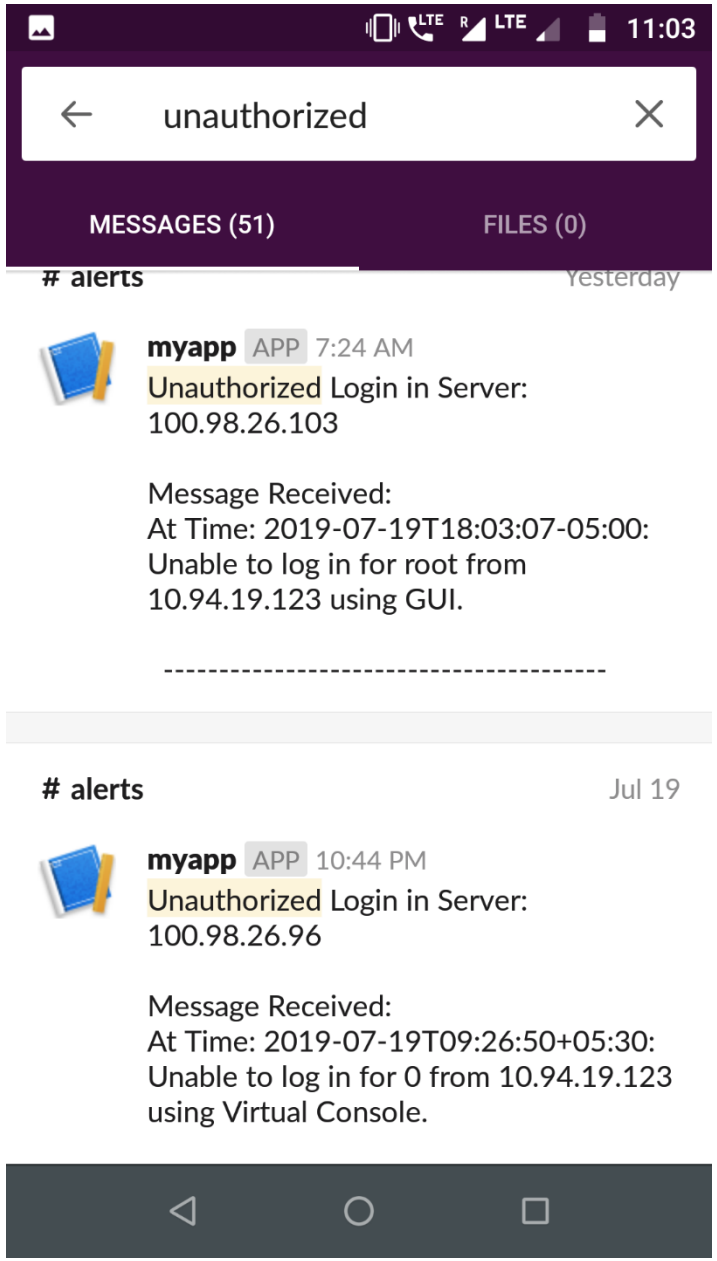
Results – Kibana Visualizations





Results – Slack Alerts





Results – Slack Alerts

Reference

https://github.com/collabnix/openusm

Questions?

Thank You