

ELK Stack

Improving the Computing Clusters at DFCTI
Through Log Analysis

Robert Poenaru

Research Assistant @ IFIN-HH

robert.poenaru@protonmail.ch

Outline

ELK Stack – Improving the Computing Clusters at DFCTI Through Log Analysis

Robert Poenaru
DFCTI
IFIN-HH
Magurele, Romania
robert.poenaru@protonmail.ch

Dragos Ciobanu-Zabet
DFCTI
IFIN-HH
Magurele, Romania
zdragos@nipne.ro

- Aim
- Motivation
- ELK stack
- Setting up the stack
- Ingesting & parsing of logs
- Currently working environment
- Future work

Abstract—The full stack logging service provided by Elastic™ has become a powerful tool within the high-performance computing community due to its ease of use, lightweight impact on the machines, performance speeds, and scalability. In the current work, we attempt to deploy such a stack on a server inside our department, which will be used for ingesting, parsing, and analyzing logs coming from multiple clusters. By analyzing the overall performance of each machine that is under continuous monitoring, we can provide immediate support in case of any issues that might occur, and more importantly, we can improve the computing power of our clusters through optimizations in terms of system management, networking, and other specific features.

Keywords— *Elasticsearch, Kibana, Logstash, pipelines, logs, metrics, clusters, compute nodes, Kubernetes.*

resource: log shipping, log ingesting, log parsing, log storing, and finally analysis – ELK stack [3].

We attempt to build and configure a full Elasticsearch logging service (ELK stack) that will be used within the department for analyzing logs from a multitude of computing clusters. In this way, the team will be able to check the status of the machines that run simulations (or, in other words, compute jobs) and see if any issues require an immediate fix. Not only that but by having an insight on the resource performances with time, one could pinpoint certain patterns preventing issues or even deploy optimizations throughout the system. Analyzing the overall performance for the compute nodes which run large simulations could also be useful in helping the research teams that develop the actual simulations: through allocating the available

About me

- Research Assistant @ Department of Theoretical Physics (IFIN-HH)
 - Part-time R.A. @ Department of Computational Physics and Information Technology (IFIN-HH)
- Ph.D. Student @ Faculty of Physics (University of Bucharest)
 - Theoretical Physics
 - Studying the structure of deformed nuclei
 - Developing (phenomenological) models that aim to describe the effects involved in nuclear motion



@basavyr

Aim

- Monitor and analyse logs, generated by:

- Servers
- Compute clusters
- Apps & services

*Developed, deployed, and running
inside our department*

- Implement the log monitoring process using the tools provided by Elastic.
- The developed stack must:
 - Support multiple log sources (different formats)
 - Be scalable enough for the entire infrastructure within DFCTI

Motivation

- Research centres around the world are in a continuous expansion in terms of the compute power. **HPC community is growing faster than ever.**
 - Our department is also working & planning on a *big upgrade* (compute resources)
- Analysis of the logs could provide information for:
 - Compute nodes performance
 - Issues with running simulations
 - System malfunction
 - Unusual network behavior

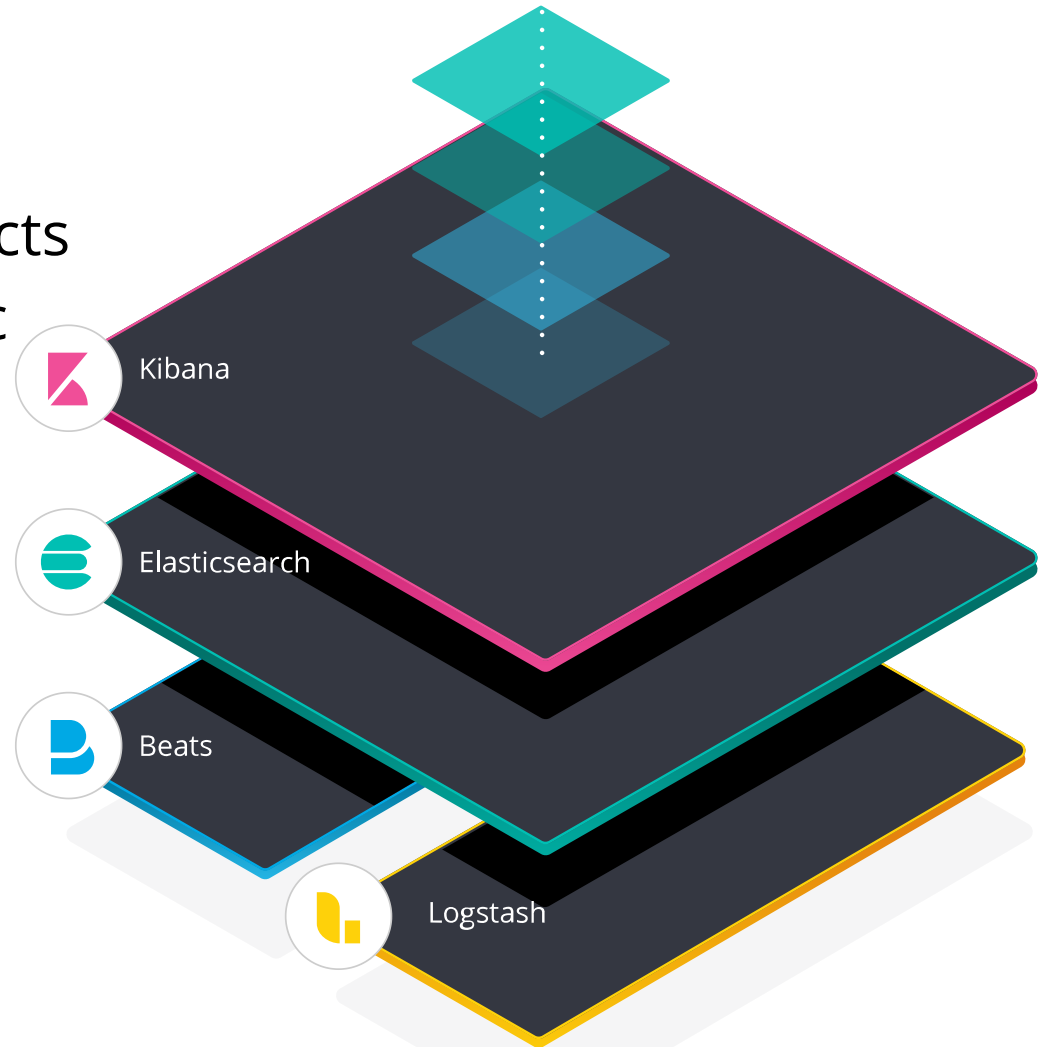
Scientific community



HPC Community

The ELK stack

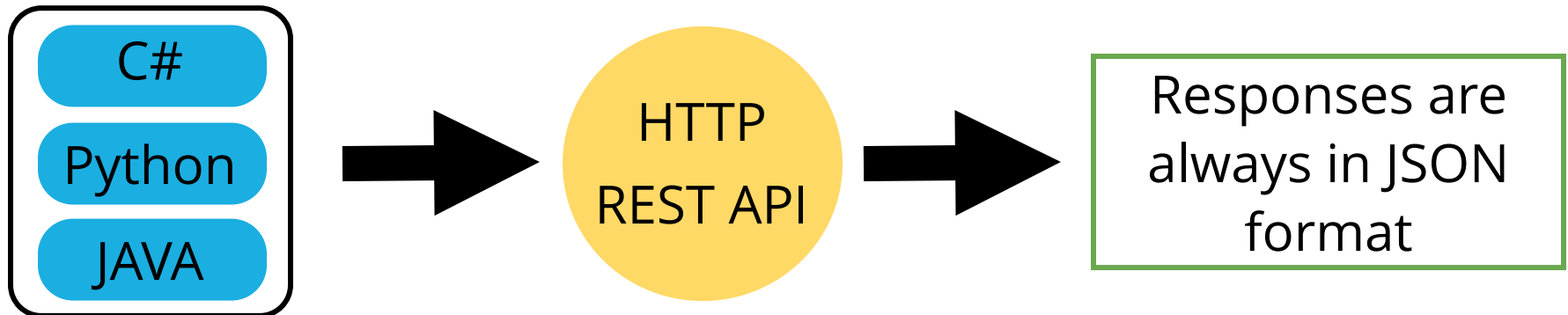
- Consists of three products developed by the Elastic company
 - Elasticsearch
 - Logstash
 - Kibana
- Open-source
- Strong community



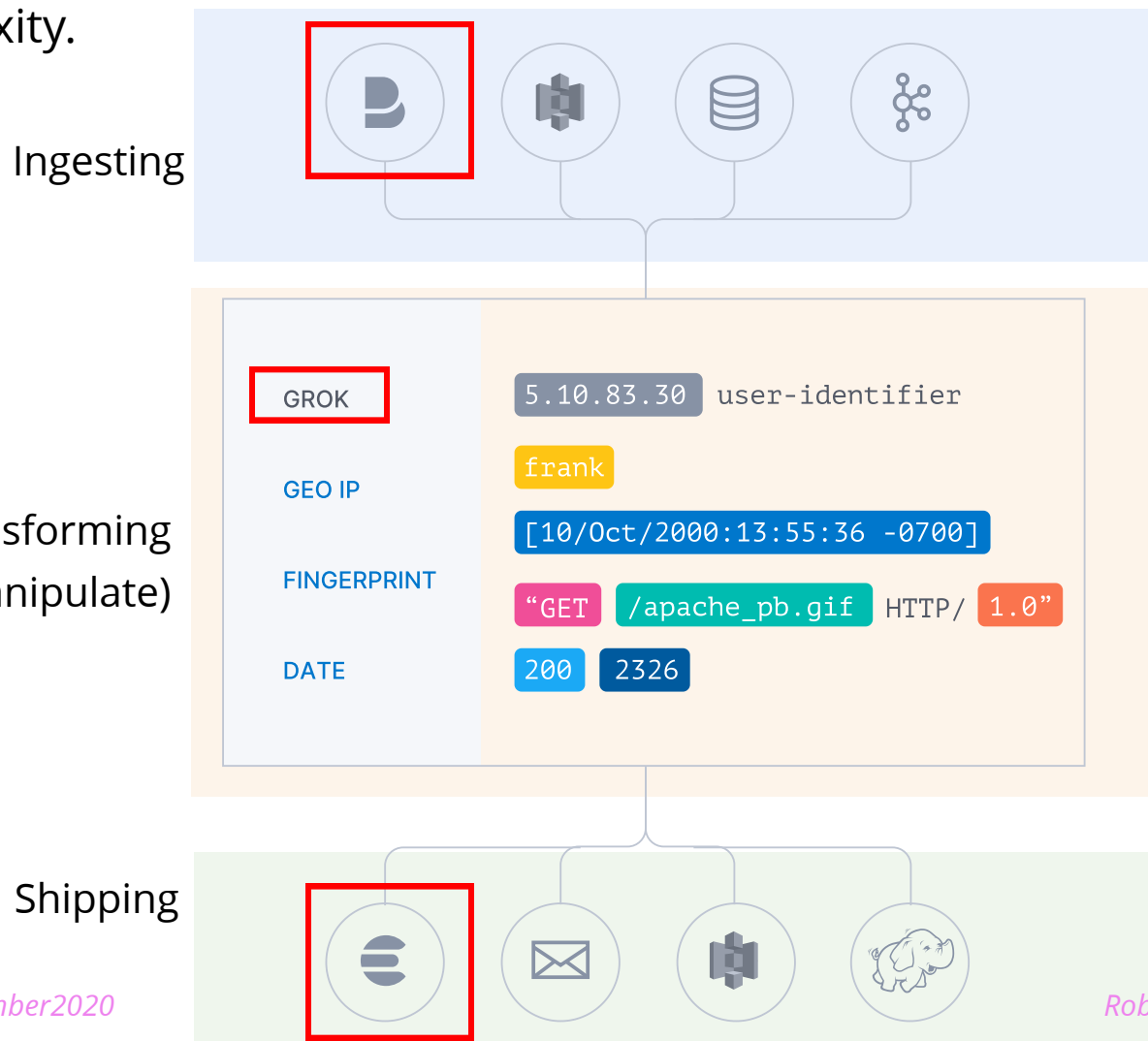
<https://www.elastic.co/what-is/elk-stack>

Main features:

- Highly scalable full-text search and analytics engine
- Store, search, and analyze big volumes of data quickly and in **near real-time**
- It provides a distributed, *multitenant*-capable engine with an HTTP web interface and schema-free JSON documents
- *API driven* ➡ Every action can be performed with a RESTful API using JSON over HTTP.



- An open-source, server-side data processing pipeline
- Dynamically *ingests*, *transforms*, and *ships* data regardless of format or complexity.



Logstash works through a configuration pipeline which has to be written by the user.

- **Input**


- Get data into logstash. Examples: files, syslogs, beats.

- **Filters**

- Intermediary processing devices in the Logstash pipeline. Examples: grok, mutate, drop, GeoIP

- **Output**

- Logstash has a variety of outputs that let you route data where you want. Examples: elasticsearch, files, graphite



```
1 input {  
2   ...  
3 }  
4  
5 filter {  
6   ...  
7 }  
8  
9 output {  
10  ...  
11 }
```

Structure of a Logstash config file

Logstash works through a configuration pipeline which has to be written by the user.

- **Input**


- Get data into logstash. Examples: files, syslogs, beats.

- **Filters**

- Intermediary processing devices in the Logstash pipeline. Examples: grok, mutate, drop, GeoIP

- **Output**

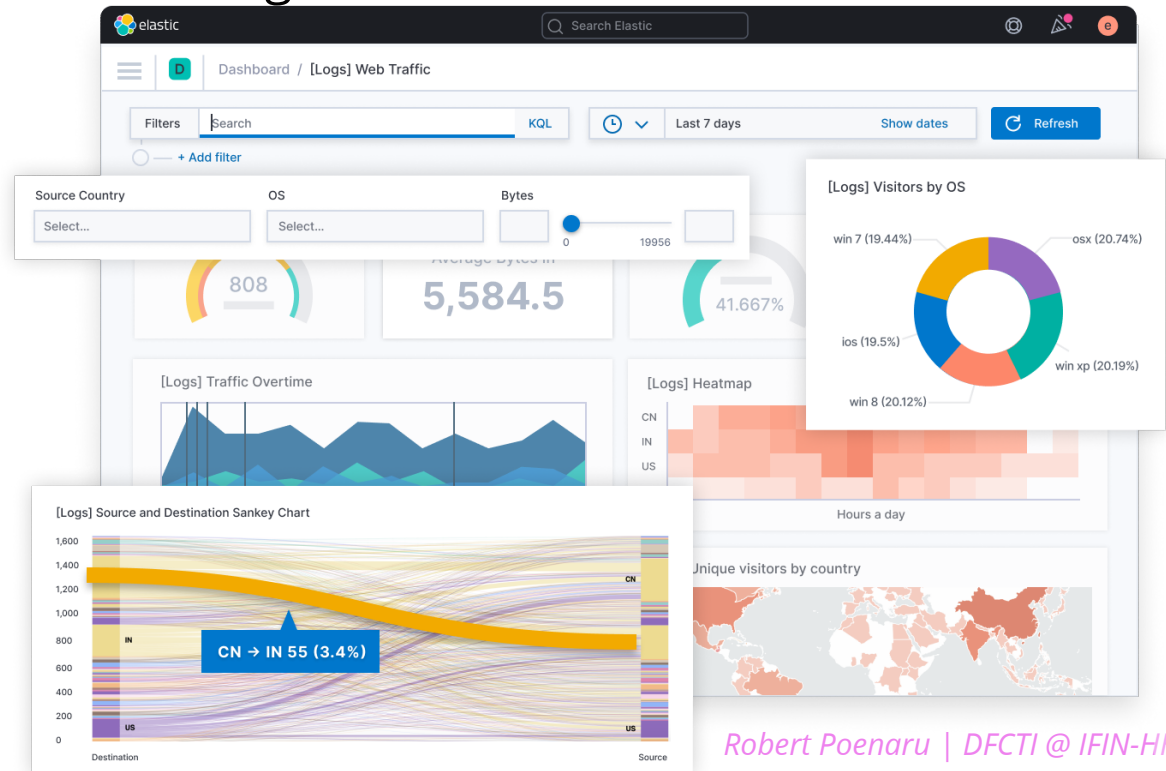
- Logstash has a variety of outputs that let you route data where you want. Examples: elasticsearch, files, graphite



```
1 input {  
2   ...  
3 }  
4  
5 filter {  
6   ...  
7 }  
8  
9 output {  
10  ...  
11 }
```

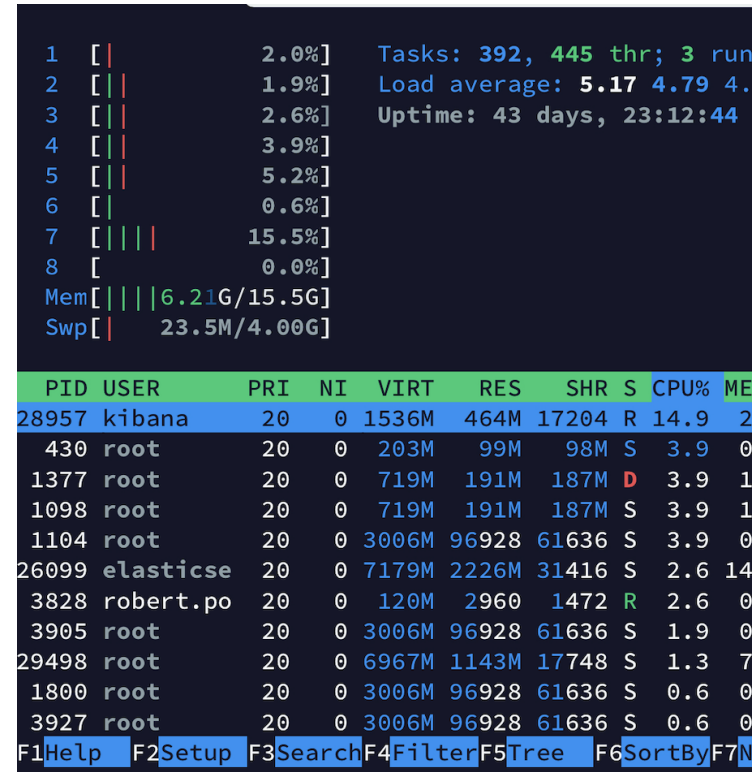
Structure of a Logstash config file

- Acting as the data visualization within the stack
- It is a web-based user interface:
 - Allows navigation through the Elasticsearch data
 - Large volumes of data can be visualized as charts, diagrams, histograms, tables, and so on.
 - The Kibana UI is accessible through the browser



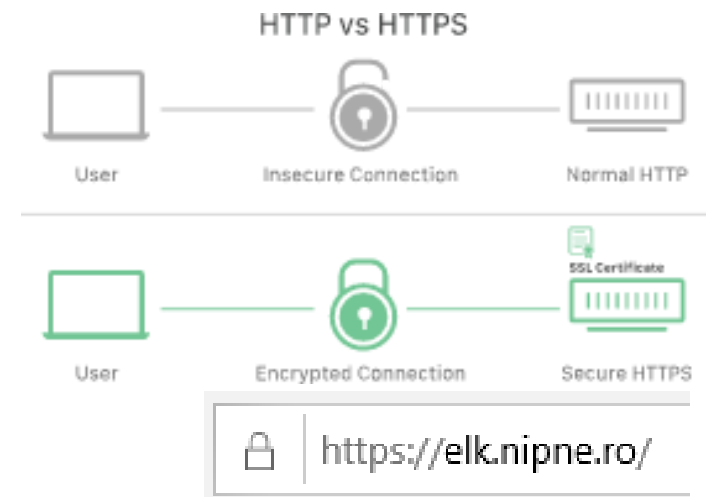
Testing environment

- Before going into main production, the stack was tested on a server within our department
 - Access to a VM running CentOS 7
 - 8 cores CPU (no multi-threading)
 - 16GB of RAM



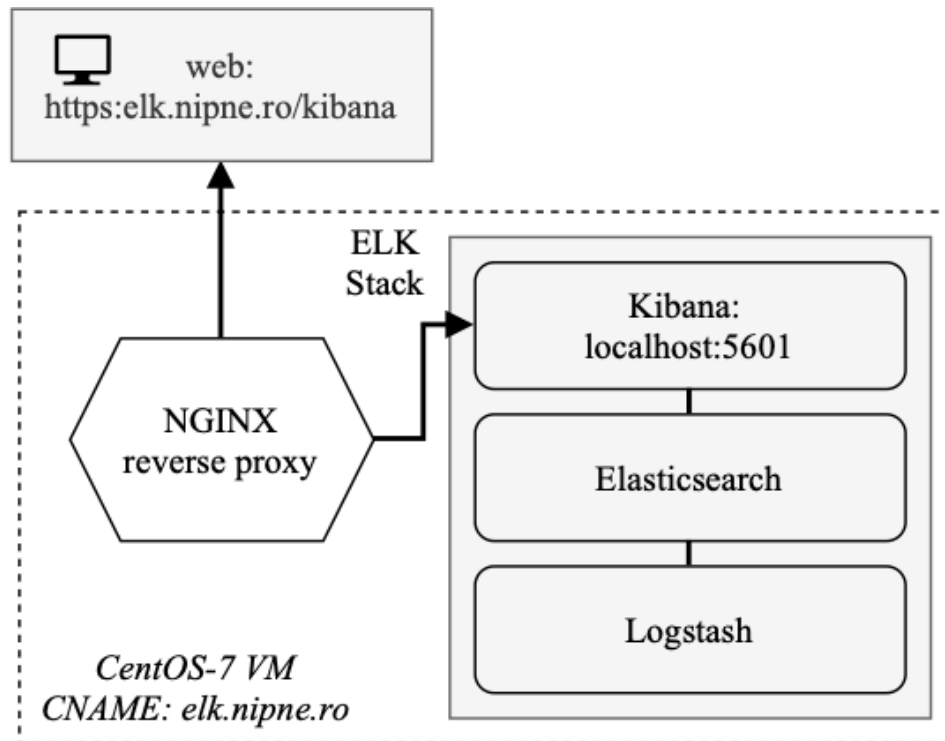
VM -> Deploy the stack

Workflow steps



Install the necessary tools:

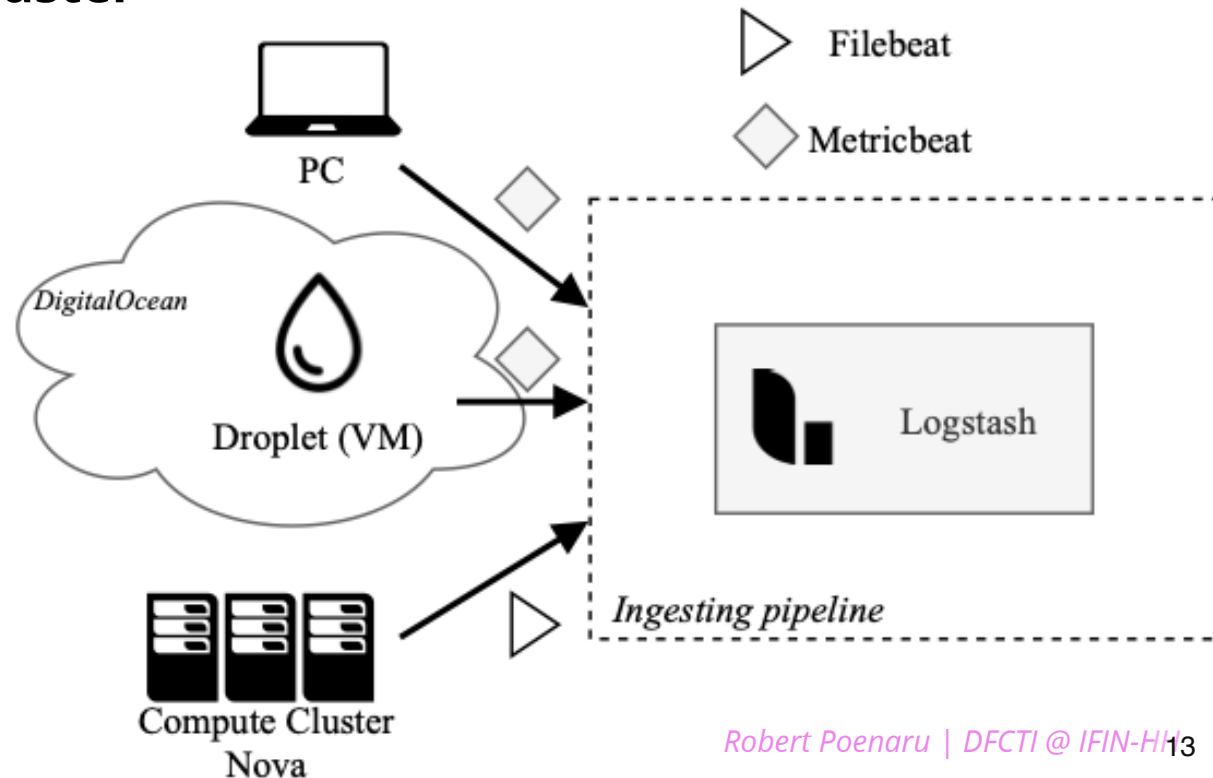
1. ES + Logstash + Kibana
2. Configure VM's firewall to accept external connections (via port 5044)
3. Install & configure NGINX as a web-server and *reverse-proxy tool*
4. Configure C.A. for the machine and implement HTTPS



ELK stack -> log sources

- A personal computer
 - sending system metrics to the server
- A VM deployed on DigitalOcean's cloud platform
 - Sending system & network metrics
- A **nova-compute-node cluster**
 - Unique log format

*Delivering logs from
external sources->
Through **beats***



ELK stack -> log sources

Nova-compute nodes

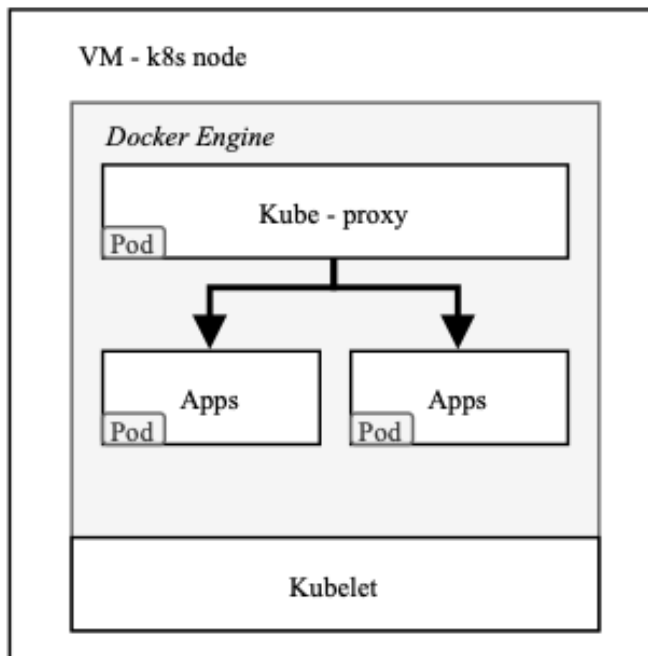
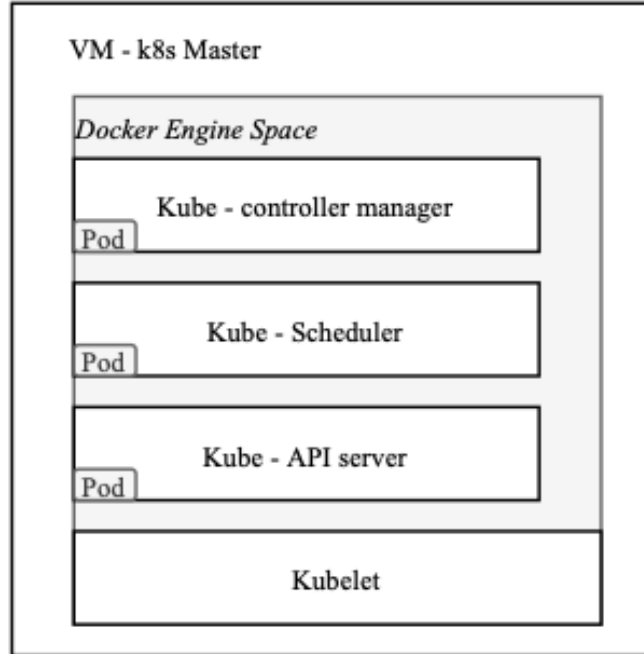
- Currently under development at DFCTI
- Using Open-Stack to create and manage **virtualized** resources as a compute grid to the scientific community within our department
 - Provide a way to set-up virtualized compute resources (built on top of physical resources)
 - Scale the resources with the compute power demand, e.g.:
 - Solid-state physics simulations
 - EM radiation interaction with matter & strong laser fields
- Deployed on the barebone metal through Open-Stack, k8s and Dockerized containers.

OpenStack

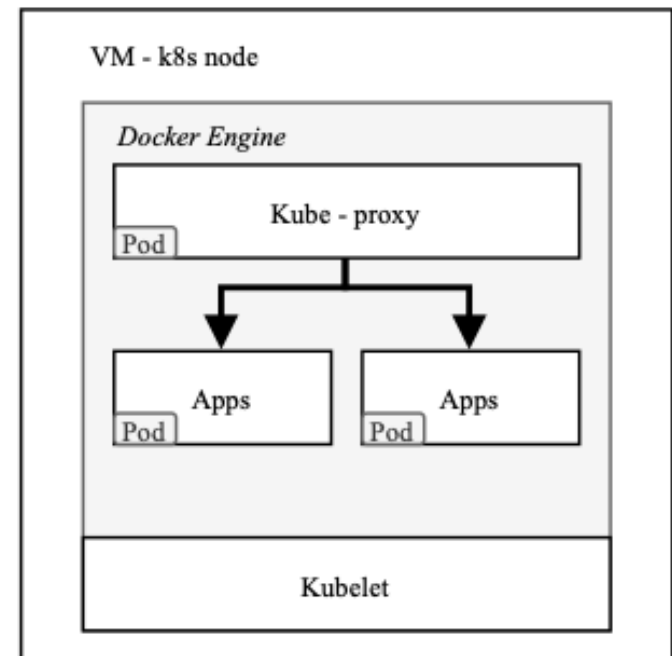
Keystone

Cinder

*Overview of the
nova compute grid*



...






...

Beats




- A free and open platform for single-purpose data shippers. They send data from hundreds or thousands of machines and systems to Logstash or Elasticsearch.

information which is sent to Logstash by a default metricbeat pipeline

<https://www.elastic.co/beats/>

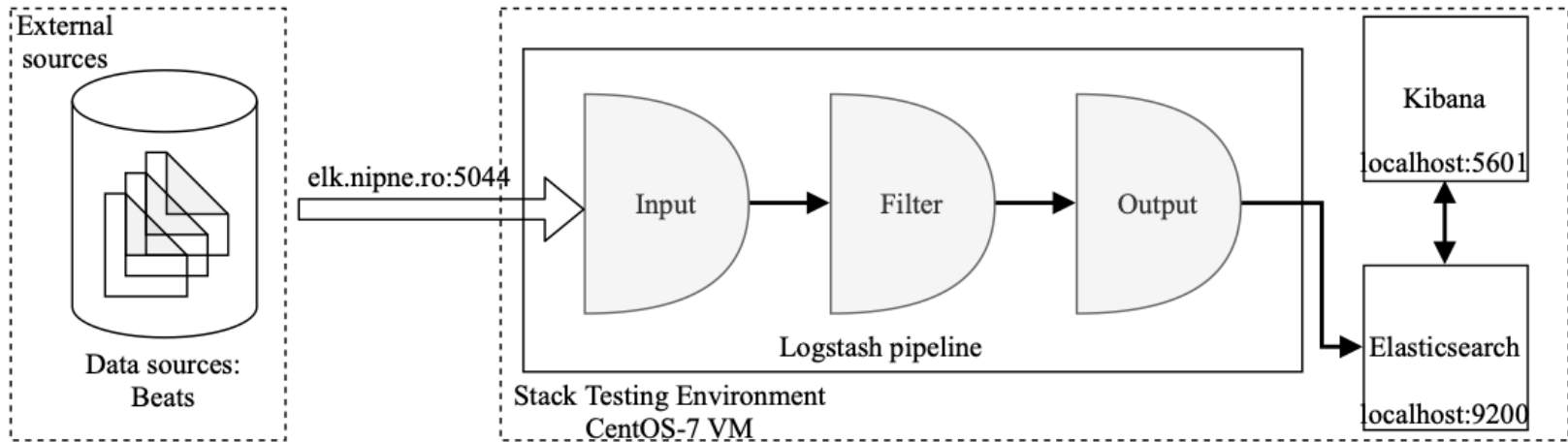
**Filebeat**
Lightweight shipper
for logs and other data**Metricbeat**
Lightweight shipper
for metric data**Packetbeat**
Lightweight shipper for
network data

→ **We used:** →

**Auditbeat**
Lightweight shipper
for audit data**Heartbeat**
Lightweight shipper
for uptime monitoring**Functionbeat**
Serverless shipper for
cloud data

```
t host.architecture
t host.hostname
t host.id
t host.ip
t host.mac
t host.name
t host.os.build
t host.os.family
t host.os.kernel
t host.os.name
t host.os.platform
t host.os.version
```

Logstash - configuration pipeline



- Metricbeat doesn't require that transformation (since the logs are pre-formatted & standardized)
- Logs coming from our compute nodes **do require filtering steps**

A basic configuration pipeline for a Logstash instance

```
input {
  beats {
    port => "5044"
  }
}
filter {
}
output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
  }
}
```

Logstash - filtering custom logs

- Requires additional configuration within the **filter**.
- Usage of *plug-in* mechanisms that are available in the Logstash package

```
2020-10-01 17:54:17.631 4831 WARNING nova.pci.utils [req-fef1267e-f155-417a-b5aa-73e1fa7eb7f1 - - - -] No net
device was found for VF 0000:3b:01.5: PciDeviceNotFoundById: PCI device 0000:3b:01.5 not found
2020-10-01 18:36:01.835 4831 INFO nova.compute.resource_tracker [req-fef1267e-f155-417a-b5aa-73e1fa7eb7f1 - - - -]
Final resource view: name=dual-c phys_ram=128453MB used_ram=41472MB phys_disk=66967GB used_disk=200GB total_vcpus=64
used_vcpus=20 pci_stats=[]
2020-10-02 10:28:05.952 4831 WARNING nova.pci.utils [req-fef1267e-f155-417a-b5aa-73e1fa7eb7f1 - - - -] No net
device was found for VF 0000:3b:01.6: PciDeviceNotFoundById: PCI device 0000:3b:01.6 not found
2020-10-02 10:28:07.238 4831 INFO nova.compute.resource_tracker [req-fef1267e-f155-417a-b5aa-73e1fa7eb7f1 - - - -]
Final resource view: name=dual-c phys_ram=128453MB used_ram=41472MB phys_disk=66967GB used_disk=200GB total_vcpus=64
used_vcpus=20 pci_stats=[]
```

```
1 2020-06-17 05:25:56,2020-06-17
2 05:27:14,b2793dec81d24496bbad171c74dfc965,0dc56f246ec14c92ba85a0239a5862d9,provider-
instance,fba88b4f-66ba-4fdb-ba1d-a5e17f576e12,bchs65,1,deleted,64
3 2020-06-17 06:33:35,2020-06-17
4 06:45:30,b2793dec81d24496bbad171c74dfc965,0dc56f246ec14c92ba85a0239a5862d9,k8s-cluster-
qt1owr3ddyny-master-0,5cec469f-fb65-4cb8-aefd-f4234e926ab0,bchs65,2,deleted,2048
5 2020-06-17 06:47:47,2020-06-17
6 07:29:19,b2793dec81d24496bbad171c74dfc965,0dc56f246ec14c92ba85a0239a5862d9,k8s-cluster-
vlwovohpq7fl-master-0,e3cb26e4-91dd-49c6-b1a5-c0a7c9d6e6c6,bchs65,2,deleted,2048
7 2020-06-17 06:48:30,2020-06-17
8 07:28:58,b2793dec81d24496bbad171c74dfc965,0dc56f246ec14c92ba85a0239a5862d9,k8s-cluster-
vlwovohpq7fl-minion-0,eb20b53c-af89-4094-b400-058f043cb138,bchs65,2,deleted,2048
```

*Example of a log-files
generated by the nova-
compute nodes*

using plug-in tools within the Filter{}

- using **grok** to parse arbitrary text and structure it => making it queryable.
- The syntax for a grok pattern is **%{SYNTAX:SEMANTIC}**
- **ruby** plug-in for complex manipulating (make new fields)
- **aggregate** plug-in (gathering multiple events into a single field)

```
1 ruby {
2     init => "require 'time'"
3     code => '
4 t1=event.get("deleted_at")
5 t2=event.get("created_at")
6 duration = ""
7 unless (t2.empty? || t2.nil?) && (t1.empty? || t1.nil?)
8 duration=(Time.parse(t1).to_i-Time.parse(t2).to_i).abs
9 end
10 event.set("VM_duration",duration)
11 '
12 }
```

```
1 grok{
2 match=>{
3     "message"=>
4 ["%{TIMESTAMP_ISO8601:created_at},%{TIMESTAMP_ISO8601:deleted_at},%{DATA:user_id},%
   {DATA:project_id},%{DATA:display_name},%{DATA:uuid},%{DATA:launched_on},%{INT:v_cpus},%
   {DATA:VM_state},%{INT:VM_memory_MB}" ]}
5 }
```

Visualizing the logs



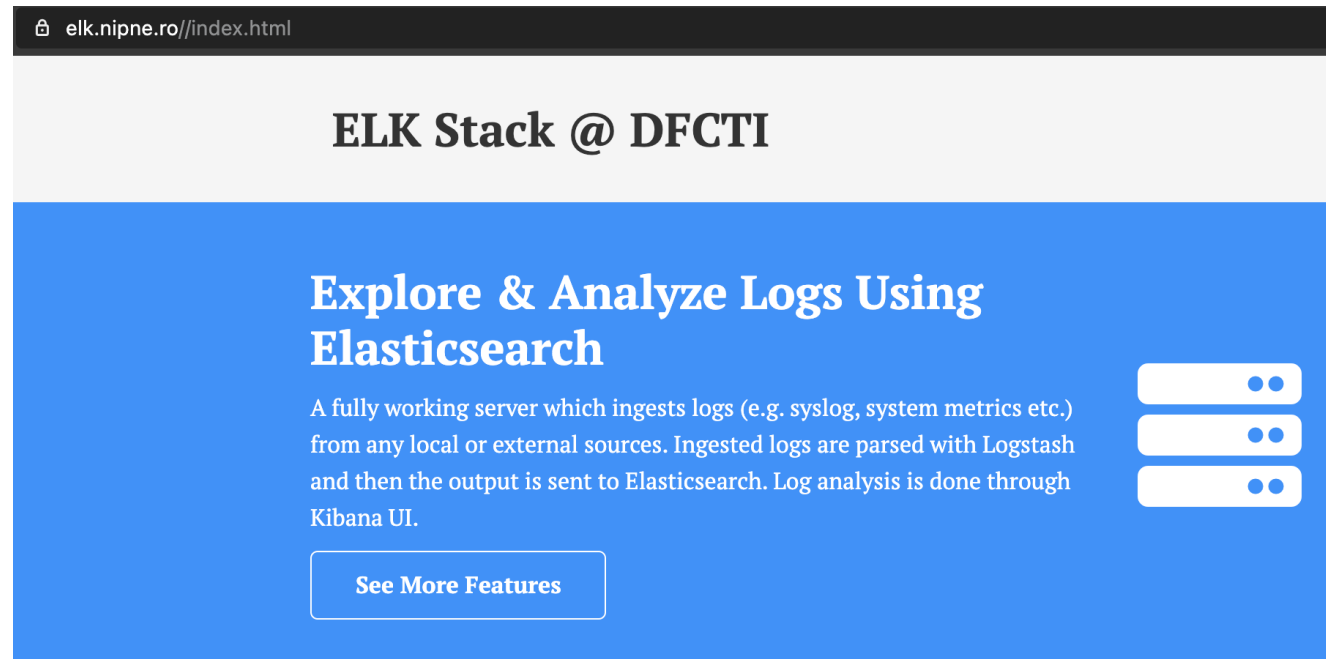
1-> contains the available fields within the Elasticsearch index

2-> contains each event as it is coming

3-> real-time evolution with the incoming events (counts per unit time).

elk - platform @ DFCTI

- The entire workflow is publicly available on a dedicated website under active development.
 - Access the Kibana UI and analyze the incoming data
 - Read our documentation
- Configuration processes for all services are available on GitHub.
 - Logstash pipelines
 - Setup files for **filebeat**



Outlook

- The developed ELK stack was successfully configured for ingesting, parsing, and storing logs from multiple sources (e.g. PC, external VMs, complex compute clusters)
 - Creating additional information from parsed logs was also possible through the usage of logstash plug-ins.
- ELK stack seems stable enough for a release within the department
- Analysis of the incoming logs might help for:
 - indicate network problems
 - identify *unresponsive* compute nodes
 - malicious behavior within the network
 - see the performance of the running applications (e.g. how do the scientific simulations scale up in terms of parallelization)