

```
3|Application | {"service":"phable.api.backend.js","file":"app.js","level":"info","message":"86ee16lbxs00 GET / 200 16 - 0.305 ms\n","timestamp":"2020-11-17T12:27:36.803Z"}
4|ADMIN_PORTAL | {"service":"phable.api.backend.js","file":"app-admin.js","level":"info","message":"kpxosju7igg00 GET / 200 1016 - 3.412 ms\n","timestamp":"2020-11-17T12:27:36.811Z"}
5|DOCTOR_PORTAL | {"service":"phable.api.backend.js","file":"app-doctor.js","level":"info","message":"coh4nfwtxvk00 GET / 200 739 - 0.931 ms\n","timestamp":"2020-11-17T12:27:36.812Z"}
4|ADMIN_PORTAL | {"service":"phable.api.backend.js","file":"app-admin.js","level":"info","message":"87i043i2rh800 GET / 200 1016 - 0.714 ms\n","timestamp":"2020-11-17T12:27:36.923Z"}
3|Application | {"service":"phable.api.backend.js","file":"app.js","level":"info","message":"4r48gg14ns000 GET / 200 16 - 2.114 ms\n","timestamp":"2020-11-17T12:27:39.810Z"}
5|DOCTOR_PORTAL | {"service":"phable.api.backend.js","file":"app-doctor.js","level":"info","message":"7wj2oh14g9k00 GET / 200 739 - 13.200 ms\n","timestamp":"2020-11-17T12:27:39.810Z"}
4|ADMIN_PORTAL | {"service":"phable.api.backend.js","file":"app-admin.js","level":"info","message":"cr5t6rub3xx00 GET / 200 1016 - 16.381 ms\n","timestamp":"2020-11-17T12:27:39.819Z"}
5|DOCTOR_PORTAL | {"service":"phable.api.backend.js","file":"app-doctor.js","level":"info","message":"4ey7y1jigcu00 GET / 200 739 - 0.793 ms\n","timestamp":"2020-11-17T12:27:42.798Z"}
3|Application | {"service":"phable.api.backend.js","file":"app.js","level":"info","message":"ehq1jyp9ltk00 GET / 200 16 - 0.340 ms\n","timestamp":"2020-11-17T12:27:42.810Z"}
4|ADMIN_PORTAL | {"service":"phable.api.backend.js","file":"app-admin.js","level":"info","message":"kwth18y4gzk00 GET / 200 1016 - 7.625 ms\n","timestamp":"2020-11-17T12:27:42.812Z"}
3|Application | {"service":"phable.api.backend.js","file":"app.js","level":"info","message":"gpab3ysaoz400 GET / 200 16 - 0.304 ms\n","timestamp":"2020-11-17T12:27:45.795Z"}
4|ADMIN_PORTAL | {"service":"phable.api.backend.js","file":"app-admin.js","level":"info","message":"z885zi4c3w0 GET / 200 1016 - 0.767 ms\n","timestamp":"2020-11-17T12:27:45.795Z"}
```

# ELK — Elastic, Logstash & Kibana



Rama Krishnan

Follow

Nov 24, 2020 · 7 min read

If you're looking to setup the **ELK** stack opting Elastic's cloud service, and visualise your logs in **Kibana**, you're at the right place! In addition to that, you can also monitor a lot of API metrics from a performance standpoint using Elastic's solution - **APM** (Application Performance Monitoring).

## Table of Contents

- [Prerequisites](#)
- [Creating ELK Deployment](#)
- [APM](#)
- [FileBeat](#)
- [Logstash](#)
- [Kibana](#)

This article will explain how we will ship NodeJS service (run using **pm2**) logs to Logstash, see API metrics in APM, and visualize logs in Kibana dashboards.

## Prerequisites:

A logger that outputs logs in JSON format!!

- A logger for Node that has capability of writing every single LOG line as a JSON. We have gone with **winston** to achieve the same.
- We will include **timestamps** to denote at what time each line was printed, along with the JSON. Other meta attributes like **file**, **requestId** can help us trace logs even better.c

## Sample log JSON

```
{
  "service": "phable.api.backend.js",
  "file": "ReportsService.js",
  "level": "debug",
  "message": {
    "reportsCount": 6723
  },
  "timestamp": "2020-11-13T08:22:35.279Z"
}
```

## Crontab commands for pm2 logs

If you run your services on server instances that reboot every day, or have a scheduled downtime to minimise cost, here are the crontabs for you!!

```
#@reboot cd path_to_service/ && sudo pm2 start index.js --log-date-  
format '' --name "ServiceName"
```

### *Note: Common pm2 logging mistakes*

- If you had pm2 services running before this change, make sure you delete and start them again (**not just restart**). Do a `pm2 delete "ServiceName"` without fail, for the `log-date-format` change to get reflected the next time you start!
- We need to make sure — **log-date-format** attribute is empty or removed. As that makes the log output a string with timestamp prefix; instead of a JSON, which is not what we need.
- Also, `colorize: true` option in winston logger adds some prefixes and suffixes like `^[[32m { JSON log data } [[36m` to indicate to the terminal that this particular line has to be green/red. Choose `colorize: false` in production winston config.

Now that pm2 logs are set up in the desired format, we will just have to work on shipping them to Logstash. Let's ship it!





Ship it before you wreck it!

## Creating ELK Deployment

If logging, monitoring and observing API metrics are your only objectives, I'd suggest to choose **Elastic Observability**. Choose the number of instances for elastic (2+ if you're keen on not losing data).

*Once you're done creating the deployment, elastic gives you username and password, keep it saved. When you want to add data to Kibana, you will need to use these credentials.*






## Elastic Observability

Consolidate your logs, metrics,  
application traces, and system availability  
with purpose-built UIs.

✓ Selected

After your deployment is created. Copy the **cloud Id and cloud auth key**. You will need this in the Filebeat setup explained later below.

### Applications ?

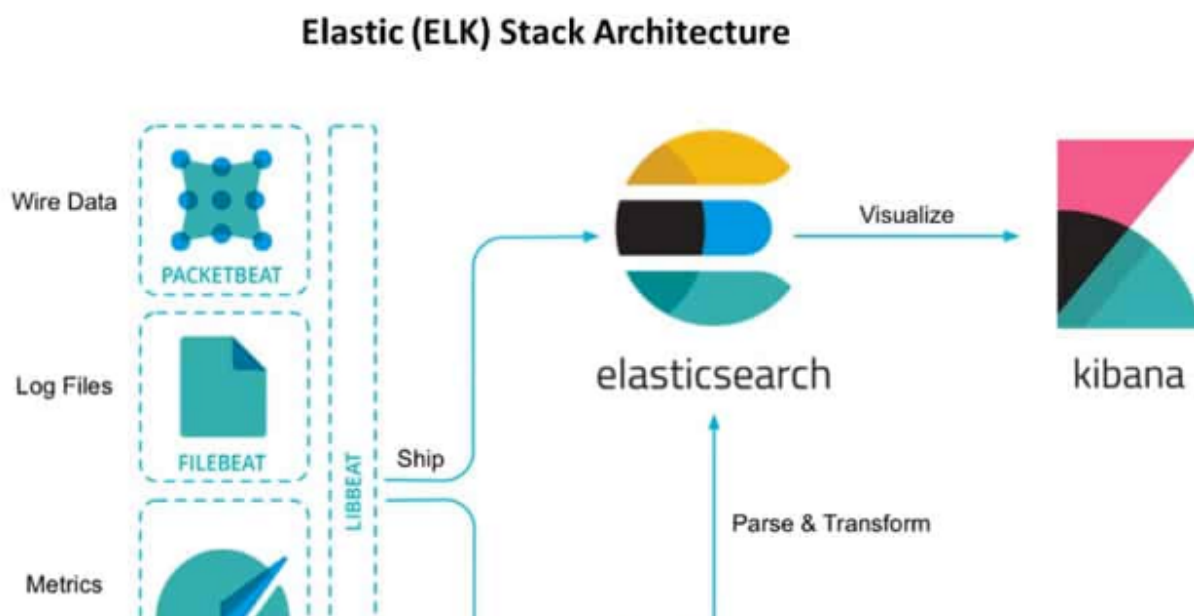
-  Elasticsearch / [Copy endpoint](#)
-  Kibana / [Launch](#) / [Copy endpoint](#)
-  APM / [Launch](#) / [Copy endpoint](#)

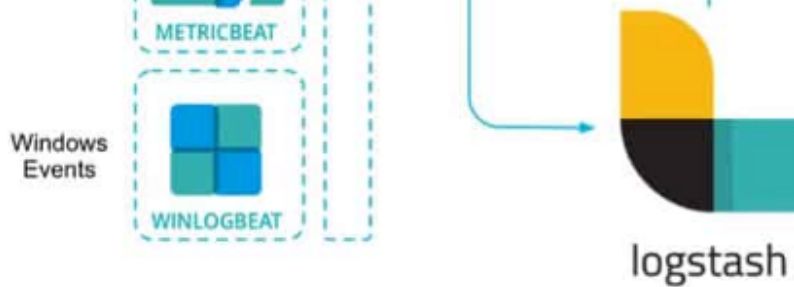
### Cloud ID ?

```
elastic-obs  
deployment:  
Y30GQ0YWVmY  
40TVmNTYwZj
```

Save the cloud id that's shown here for **filebeat.yml** configuration later.

Now that the deployment is created, let's see how the architecture looks like and works... As shown in the picture below, log files is the use case we'll be focusing on and we'll see how we can have your logs shipped using FILEBEAT and visualize it in dashboards in Kibana!





## Tools/Solutions used

### APM (Application Performance Monitoring)

Elastic APM is an application performance monitoring system built on the Elastic Stack. It allows you to monitor software services and applications in real-time, by collecting detailed performance information on response time for incoming requests, database queries, calls to caches, external HTTP requests, and more. This makes it easy to pinpoint and fix performance problems quickly.

Elastic APM also automatically collects unhandled errors and exceptions. Errors are grouped based primarily on the stack trace, so you can identify new errors as they appear and keep an eye on how many times specific errors happen.

Can give us an overview of which call is taking how much time. Can think of it as a node middleware that's been added to monitor api calls and push metrics to this solution from Elastic.

### Snippet added in Node service to plug this tool in

Reference: <https://www.elastic.co/guide/en/apm/agent/nodejs/master/configuring-the-agent.html>

```
// app.js OR index.js -- entry point
// add config values for APM to .env variables

const {
  apmEnv, apmServiceName,
  apmSecretToken, apmServerUrl
} = process.env

if (apmEnv && apmEnv !== 'dev') {
  var apm = require('elastic-apm-node').start({
    serviceName: apmServiceName,
    secretToken: apmSecretToken,
    serverUrl: apmServerUrl
  })
}
```

## Action items that can be created from this solution

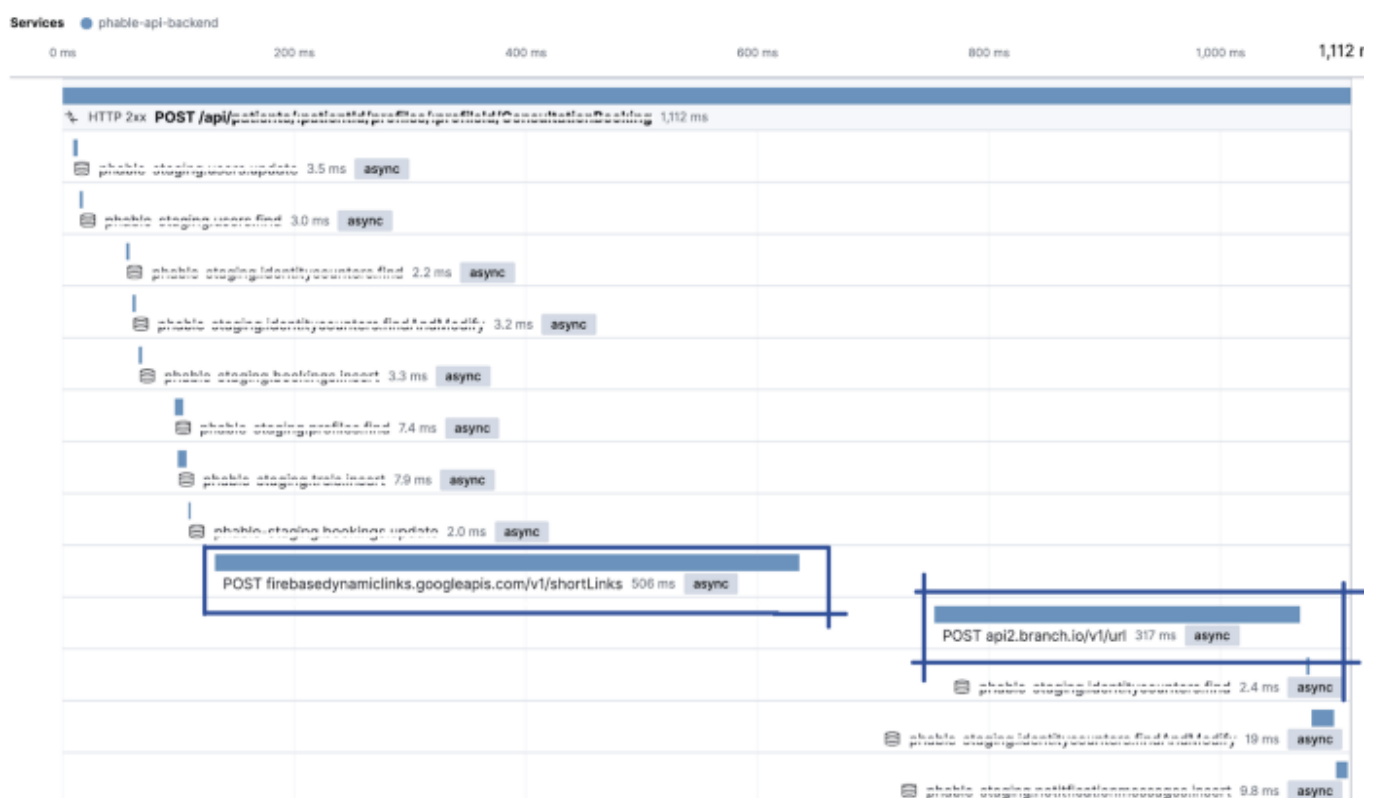
- We can sort the average time taken for every call in the backend. And pick each one up, and optimize wherever there is scope to improve!

**Eg:** The following data clearly shows that /notify endpoint requires a revisit. We can find more problems once we have this in prod, these is performance metrics from staging

Transactions				
Name	Avg. duration ↓	95th percentile	Trans. per minute	Impact ⓘ
GET /api/notify	76,935 ms	77,070 ms	0.0 tpm	
GET /api/notify/1	15,414 ms	15,663 ms	0.0 tpm	
GET /api/notify/2	15,398 ms	15,663 ms	0.0 tpm	
POST /api/notify	994 ms	991 ms	0.0 tpm	
GET /api/notify/3	888 ms	1,335 ms	0.0 tpm	
GET /api/notify/4	875 ms	1,514 ms	0.0 tpm	
GET /api/notify/5	789 ms	1,096 ms	0.0 tpm	
POST /api/notify/6	735 ms	1,169 ms	0.0 tpm	

- Find time taken in worst case scenarios. Some APIs you write may seem performant for the sample data you're requesting. But APM provides the 95th percentile time taken value for every API. That way we can find which calls are slow in production
- Find which method or function block inside the APIs implementation is taking the longest to execute. That level of clarity is provided for each and every API call by APM.

**Eg:** The following data shows Firebase dynamic link generation took the longest amount of processing time in one of our api calls!



## FileBeat

It can be looked at like a **log watcher**. It's a service/executable developed by elastic, which comes with its own config file. You need to mention the path of the directory that it needs to monitor for logs. `\home\ubuntu\.pm2\logs\*.log` in our case.

### Ship to Elasticsearch or Logstash. Visualize in Kibana.

Filebeat is part of the Elastic Stack, meaning it works seamlessly with Logstash, Elasticsearch, and Kibana. Whether you want to transform or enrich your logs and files with Logstash, fiddle with some analytics in Elasticsearch, or build and share dashboards in Kibana, Filebeat makes it easy to ship your data to where it matters most.

### Filebeat.yml

```
- type: log
# Change to true to enable this input configuration.
enabled: true
# Paths that should be crawled and fetched.
- /home/ubuntu/.pm2/logs/*.log

#elastic cloud config

cloud.id: "phable_staging:xxxxxxxxxxxxxxxxxxxx"
cloud.auth: "elastic:xxxxxxxxxxxxxx"

#cloud id can be found in the your deployment's landing page
#use the pwd that was generated on creating a deployment here
"elastic:xxxxxxxxxx"
```

Installation guide:

<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation-configuration.html>

## Logstash

*Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.*

source: [www.elastic.co](http://www.elastic.co)



I don't think I could have described it any better, you can think of it as the port where your logs reach after you ship 'em! ;)

Now that filebeat is set up, we need it to ship logs to logstash — as the name suggests — a stash for logs. All we need to do is enable the logstash module in filebeat. It automatically ships every single line written to pm2 log files to logstash.

## Kibana

Whatever logs go into Logstash can be visualized in Kibana, with kibana you can create customized dashboards. With pm2 logs coming in as jsons in this format. You can add filters to sub fields in the JSON too, eg: search by **message.body contains 'exception'**

### Sample JSONs coming to Kibana — A Morgan log

```
{
  "service": "phable.api.backend.js",
  "file": "app.js",
  "level": "info",
  "message": "fy7aihcu3vk00 POST /api/apiName 200 100-971.558 ms",
  "timestamp": "2020-11-12T12:52:53.029Z"
}
```

Kibana sample view...

Time ▾	message
Nov 17, 2020 @ 18:16:38.671	{"service": "phable.api.backend.js", "file": "app-doctor.js", "level": "info", "message": "48bzaz57k0400 GET / 200 739 - 0.714 ms\n", "timestamp": "2020-11-17T12:46:38.671Z"}
Nov 17, 2020 @ 18:16:37.672	{"service": "phable.api.backend.js", "file": "app.js", "level": "info", "message": "iqijn0tajjk00 GET / 200 16 - 0.635 ms\n", "timestamp": "2020-11-17T12:46:37.672Z"}
Nov 17, 2020 @ 18:16:35.674	{"service": "phable.api.backend.js", "file": "app-admin.js", "level": "info", "message": "gvkqmd6p6vs00 GET / 200 1016 - 6.232 ms\n", "timestamp": "2020-11-17T12:46:35.674Z"}
Nov 17, 2020 @ 18:16:35.671	{"service": "phable.api.backend.js", "file": "app-doctor.js", "level": "info", "message": "8dgv50q9rgo00 GET / 200 739 - 4.526 ms\n", "timestamp": "2020-11-17T12:46:35.671Z"}
Nov 17, 2020 @ 18:16:34.671	{"service": "phable.api.backend.js", "file": "app.js", "level": "info", "message": "aqfun59n4yw00 GET / 200 16 - 0.437 ms\n", "timestamp": "2020-11-17T12:46:34.671Z"}

Once we load the data in the dashboard, and after we get hold of KQL {Kibana Querying Language : <https://www.elastic.co/guide/en/kibana/7.9/kuery-query.html> }

- we can filter the logs with queries like message contains “exception” and filter out occurrences of errors/exceptions
- Build reports and visualize them, say you logged the JSON here..

```
{
  "reportTypeA" : 20,
  "reportTypeB" : 72,
```



```
"reportTypeC" : 19  
}
```

once every day in the logs' JSON output. Then you can use Kibana and visualize it by building reports like this too...

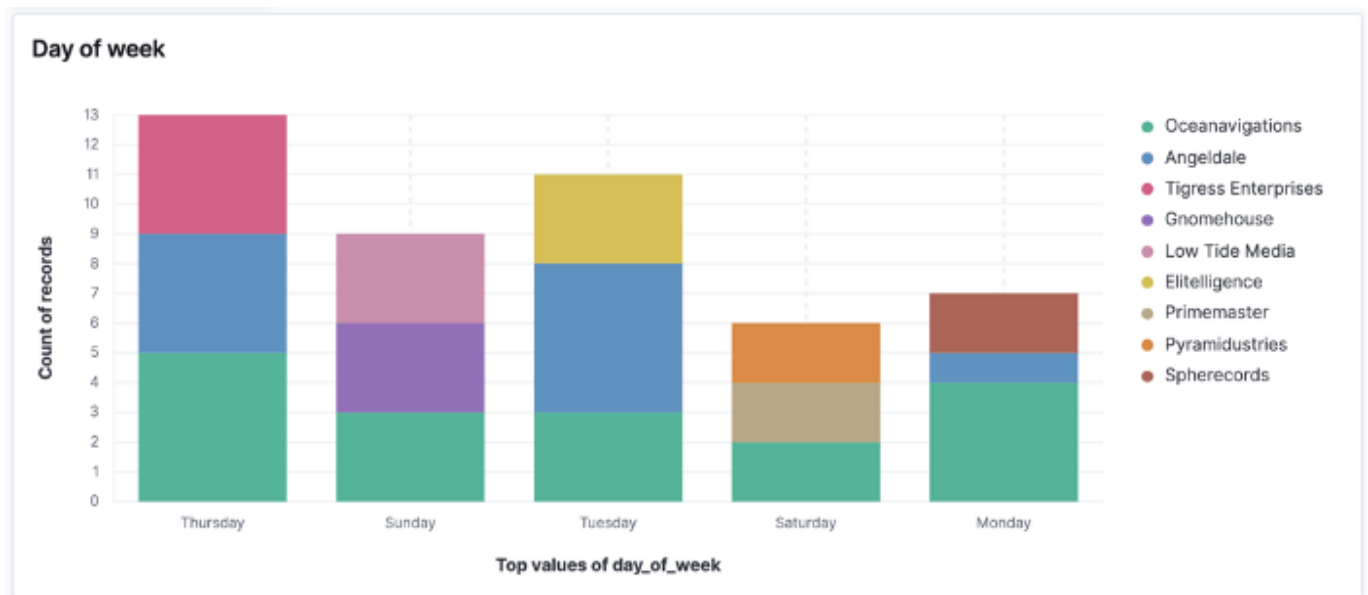


Image courtesy: [www.elastic.co](http://www.elastic.co)

Hope this article helped you understand ELK setup using Elastic's cloud solution. We at **Phable** will be running some cost-benefit analysis to see if this is a sustainable solution, or switch to in-house elastic server deployments after monitoring how the solution works for a couple months. Hopefully the cost benefit analysis becomes another fruitful article in the future.

## WHY LOGGING IS SO IMPORTANT

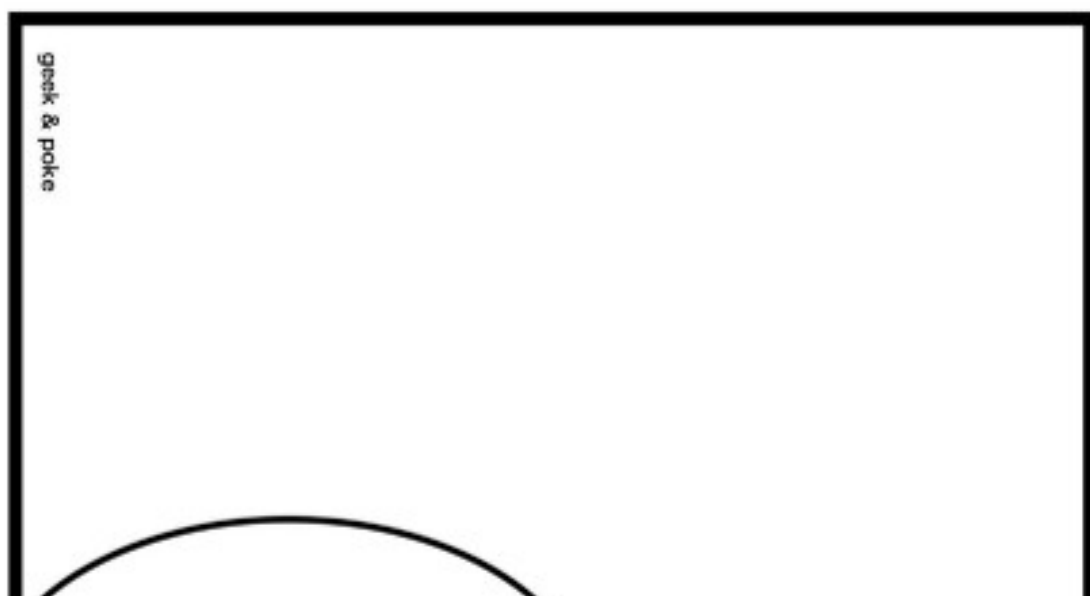




Image courtesy: <https://in.pinterest.com/pin/743094007243620223> Geek and Poke

Until then, happy logging!

Elastic Stack

Elk

Logging

Pm2

Health Technology

 Medium

[About](#) [Help](#) [Legal](#)

Get the Medium app

 Download on the  
App Store

 GET IT ON  
Google Play