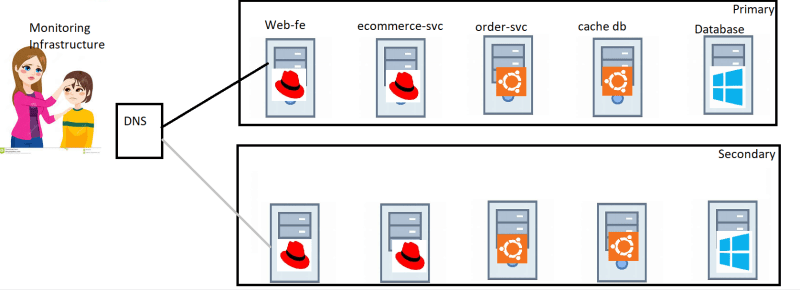
JULY 5, 2022

DevOps Classroomnotes 05/Jul/2022

**Monitoring & log analysis**

* Main aim of devops pipeline is reduce time to market & increase business value.
* Lets consider a sample ecommerce application  
  
* We need monitoring and log analysis
* There is
  + Server Monitoring
  + Application Monitoring
  + Log Analysis
* There are several tools.
* Server Monitoring
  + Nagios
  + Zabbix
* Application Monitoring
  + App Dynamics
  + New Relic
* Log Analysis
  + Splunk
* There is an open source productl line which was earlier referred as ELK & now as Elastic Stack.

**Share this:**

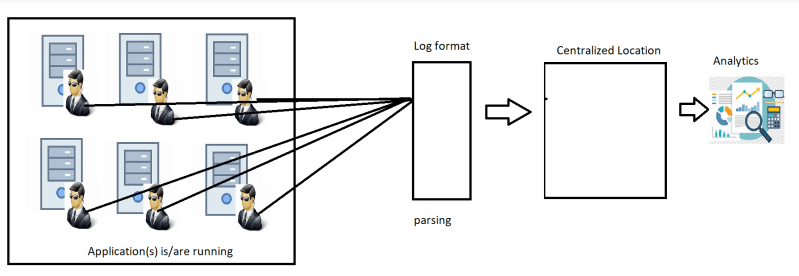
JULY 6, 2022

DevOps Classroomnotes 06/Jul/2022

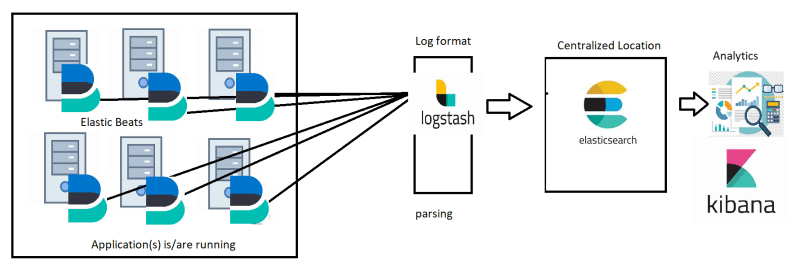
**Decentralized Log and Centralized Log Analysis**

* To demonstrate this i will be installing
  + Apache on linux
  + IIS on Windows Server
  + Some application on linux Server
* I will use this applications & then lets understand the logs.
* Each application/operating system logs the data somewhere, but
  + They have different formats
  + Logs are text records which are not queryable.
* Logs are located in the respective servers, searching for error message by login into each server will be a cumbersome activity this is decentralized logging.
* If we export the logs from all the servers into some common/central location this is referred as centralized log server.
* If we can make logs queryable it will save much more time.
* If we can build charts then it would be much easier to troubleshoot.

**Generalized Architecture for Centralized Logging**



**Elastic Stack Architecture**



**Share this:**

JULY 7, 2022

DevOps Classroomnotes 07/Jul/2022

**Elastic Stack**

**Elastic Search**

* Basic Purpose: To store the logs in the centralized location
* What is it good at? Searching text and resistent to node failures
* Installing Elastic Search:
  + Lets create a free vm (ec2 instance) (Azure=> Standard\_B1s/t2.micro)
* [Refer Here](https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html) for the official documentation

**Windows Package Manager = Chocolatey**

* [Refer Here](https://chocolatey.org/install) to install choco
* We need
  + Git Bash choco install git -y
  + Visual Studio Code choco install vscode -y

**Mac Package Manager => homebrew**

* [Refer Here](https://brew.sh/) to install homebrew on mac
* We need
  + git brew instal git
  + visual studo code brew install --cask visual-studio-code

**Creating Linux Machines**

* Azure: [Refer Here](https://directdevops.blog/2022/07/07/devops-classroomnotes-07-jul-2022/youtube.com/watch?v=P9X-4Z-NeGg&list=PLuVH8Jaq3mLuqXuGs6aeqxhuvCYSzB1kT&index=2)
* AWS: [Refer Here](https://www.youtube.com/watch?v=me2s3mTNwGo&list=PLuVH8Jaq3mLszrC7lv68a0VcrDripW-HK&index=2)

**Share this:**

JULY 8, 2022

DevOps Classroomnotes 08/Jul/2022

**Installing Elastic Search**

**Scenario 1: Installing Elastic Search on ubuntu 20.04**

* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html) for the official documentation.
* We would be installing from package manager [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html#deb-repo)
* [Refer Here](https://www.elastic.co/support/matrix) for support matrix of elastic search, kibana, logstash and beat
* On Ubuntu

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

sudo apt-get install apt-transport-https

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list

sudo apt-get update && sudo apt-get install elasticsearch

* The installation gives us following information

--------------------------- Security autoconfiguration information ------------------------------

Authentication and authorization are enabled.

TLS for the transport and HTTP layers is enabled and configured.

The generated password for the elastic built-in superuser is : LJGveyix-ey-+bsvdGvC

If this node should join an existing cluster, you can reconfigure this with

'/usr/share/elasticsearch/bin/elasticsearch-reconfigure-node --enrollment-token <token-here>'

after creating an enrollment token on your existing cluster.

You can complete the following actions at any time:

Reset the password of the elastic built-in superuser with

'/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic'.

Generate an enrollment token for Kibana instances with

'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana'.

Generate an enrollment token for Elasticsearch nodes with

'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s node'.

-------------------------------------------------------------------------------------------------

### NOT starting on installation, please execute the following statements to configure elasticsearch service to start automatically using systemd

sudo systemctl daemon-reload

sudo systemctl enable elasticsearch.service

### You can start elasticsearch service by executing

sudo systemctl start elasticsearch.service

* The major locations:
  + /usr/share/elasticsearch => installation of files is done
  + /etc/elasticsearch => configuration of elastic search
* Elastic search installation configurations are by default configured to be accesible from the same machine (loopback)
* Now change the network host with private ip and discovery.seed\_hosts with private ip as discussed in the classroom in the file /etc/elasticsearch/elasticsearch.yml. Then execute the below commands
* Elastic search works on port 9200 by default
* To view the logs of elastic search sudo journalctl --unit elasticsearch
* Elastic search 8+ is secure by default access by https://<publicip&gt;:9200 and enter credentials

**Share this:**

JULY 9, 2022

DevOps Classroomnotes 09/Jul/2022

**Installing Elastic Search**

* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html#deb-repo)
* After sucessful installation try accessing https://<public-ip&gt;:9200

**Elastic Search Details**

* It exposes functionailty over REST API (API)
* To use the API we need to compose/create JSON
* To Configure elastic search we will be using yaml
* [Refer Here](https://www.youtube.com/watch?v=ggOmHlnhPaM&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=15) for json and yaml tutorial

JULY 11, 2022

DevOps Classroomnotes 11/Jul/2022

**Kibana**

* Kibana is the UI of the Elastic Stack
* Lets use Kibana to interact with Elastic Search
* Installing kibana: [Refer Here](https://www.elastic.co/guide/en/kibana/current/deb.html)
* After installing kibana, the issue which we have is we don’t have users which are non-super users.
* In the kibana configuration, we cannot use super user credentials
* Some HTTP Methods
  + GET
  + PUT/POST
  + DELETE
* HTTP Status
  + 1xx => Information
  + 2xx => Success
  + 3xx => redirection
  + 4xx => client side errors
  + 5xx => Server side errors
* Exercise: Interact with elastic search using curl statements and [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html) for the rest api of elastic search

**Share this:**

JULY 14, 2022

DevOps Classroomnotes 14/Jul/2022

**Elastic Search Concepts**

* Elastic Search is Distributed Store for documents in json format which is horizontally scalable & it is effecient in searching text
* Shards and Replicas
* In Elastic Search, When we insert documents an Inverted Index is built [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/documents-indices.html)
* From the above concepts:
  + How Elastic search works even when server(s) in cluster is down?
    - Shards and Replicas
  + Why Elastic Serarch is good at searching text?
    - Inverted Index

**Exercise:**

* Try Building inverted index for the following data

Until one has loved an animal, a part of ones soul remains unawakened. â€“Anatole France

Animals are such agreeable friendsâ€”they ask no questions; they pass no criticisms. â€“George Eliot

An animals eyes have the power to speak a great language.â€“Martin Buber

The better I get to know men, the more I find myself loving dogs. â€“Charles de Gaulle

Dogs are our link to paradise. They don't know evil or jealousy or discontent. â€“Milan Kundera

* What would be the ideal number of shards and replicas when i have 5 servers and assume the cluster to work even if two nodes are down

**Share this:**

JULY 15, 2022

DevOps Classroomnotes 15/Jul/2022

**Index and Document**

* Index and Document
* The process of inserting data/documents is called as indexing
* Lets refer the elastic search rest api’s [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html)
* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/indices.html) for index apis
* Lets create curl request to create the index

ESPASS='nplUwdODeH4xtPfp=CT2'

SERVER\_URL='https://20.198.70.188:9200'

# Checking if the index exists

curl -I -u elastic:$ESPASS "${SERVER\_URL}/sample" --insecure

# we got 404 response => index not found

\* Lets create one more index with  
\* name: sample2  
\* number of shards 1  
\* number of replicas 1

**Exercise:**

* Create an index with yourname
* Check if the index exists or not
* delete the index
* The commands used are

curl -I -u elastic:$ESPASS "https://20.198.70.188:9200/khajaibrahim?pretty" --insecure

curl -X DELETE -u elastic:$ESPASS "https://20.198.70.188:9200/khajaibrahim?pretty" --insecure

curl -I -u elastic:$ESPASS "https://20.198.70.188:9200/khajaibrahim?pretty" --insecure

* Now let me insert a document about Devops

{

"name": "DevOps",

"tools": ["Git", "Ansible", "Jenkins", "Docker", "K8s", "Terraform", "SRE"]

}

* The curl request is

curl -X POST -u elastic:$ESPASS "https://20.198.70.188:9200/khajaibrahim/\_doc?routing=kimchy&pretty" -H 'Content-Type: application/json' --insecure -d'

{

"name": "DevOps",

"tools": ["Git", "Ansible", "Jenkins", "Docker", "K8s", "Terraform", "SRE"]

}

'

**Share this:**

JULY 17, 2022

DevOps Classroomnotes 17/Jul/2022

**Configure Kibana**

* When we configured kibana, the error was cannot login with superuser account (elastic)
* Note: We had configuration issues which will be resolved in next session
* For avoiding installation and configuration at this moment, lets create an account in elastic cloud which is free for 14 days. [Refer Here](https://cloud.elastic.co/registration?baymax=rtp&elektra=en-cloud-page&storm=nav&rogue=mp-b)
* For Converting text into multiple fields we need to parse the text via regular expressions (GROK)
* Any Log should tell about
  + Log level: INFO, ERROR, DEBUG, TRACE
  + When it was created => Datetime
  + From which component => Source
  + message

**Share this:**

JULY 19, 2022

DevOps Classroomnotes 19/Jul/2022

**LogStash**

* Extract , Transform and Load (ETL):
  + Extract i.e. read the logs from source
  + Transform the log records into queryable fields
  + load i.e. store/push/index into elastic search
* Install Logstash: [Refer Here](https://www.elastic.co/guide/en/logstash/current/installing-logstash.html)
* Logstash performs extract, transform and load with the help of plugins
  + Extract => input plugins [Refer Here](https://www.elastic.co/guide/en/logstash/current/input-plugins.html)
  + Transform => filter plugins [Refer Here](https://www.elastic.co/guide/en/logstash/current/filter-plugins.html)
  + Load => Output plugins [Refer Here](https://www.elastic.co/guide/en/logstash/current/output-plugins.html)

**Play with Logstash**

* Logstash is located generally in /usr/share/logstash/bin
* Lets search for the options sudo /usr/share/logstash/bin/logstash --help
* [Refer Here](https://www.elastic.co/guide/en/logstash/current/running-logstash-command-line.html) for the command line options
* To the logstash we need to provide pipeline as input

input

{

<plugin-name>

{

<option-1> => <value-1>

...

<option-n> => <value-n>

}

}

filter

{

<plugin-name>

{

<option-1> => <value-1>

...

<option-n> => <value-n>

}

}

output

{

<plugin-name>

{

<option-1> => <value-1>

...

<option-n> => <value-n>

}

}

* filter is optional
* Activity 1: Lets create a pipeline which reads from command line (termianl) and shows the output on the terminal. Save this in ~/logstash-pipelines/hello-wrold.conf

input

{

stdin

{

}

}

output

{

stdout

{

}

}

* Now try to run logstash with the following command sudo ./logstash -f ~/logstash-pipelines/hello-wrold.conf
* Now lets give some input
* Activity 2: Now lets try to create a logstash pipeline which reads from stdin and stores in a file and also shows the output in stdout. save it as activity2.conf

input

{

stdin {}

}

output

{

stdout {}

file

{

path => '/tmp/activity2'

create\_if\_deleted => true

flush\_interval => 0

}

}

* Now run logstash sudo ./logstash -f ~/logstash-pipelines/activity2.conf
* Activity 3: Create a logstash pipeline which reads the inputs from a file /tmp/messages and writes the output to stdout

input

{

file

{

path => '/tmp/messages'

start\_position => 'beginning'

}

}

output

{

stdout

{

}

}

* Next Steps:
  + Lets read logs from some applications

**Share this:**

JULY 20, 2022

DevOps Classroomnotes 20/Jul/2022

**Log Parsing with Logstash**

**Filter Plugins**

* Lets read the input from stdin and display the output to the stdout and i want to add one field called as purpose with value learning

input

{

stdin {}

}

filter

{

mutate {

add\_field => {

"purpose" => "learning"

}

}

}

output

{

stdout {}

}

* Lets search for a filter plugin which can add fields . [Refer Here](https://www.elastic.co/guide/en/logstash/current/filter-plugins.html) for all the standard filter plugins and [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html) for mutate filter
* Now start the logstash with the above pipeline
* Activity 5: Split the message with commas

input

{

stdin {}

}

filter

{

mutate {

split => {

"message" => ","

}

}

}

output

{

stdout {}

}

* Activity 6: Convert the message into upper case and then split the message with ,

input

{

stdin {}

}

filter

{

mutate {

uppercase => [ "message" ]

}

mutate {

split => {

"message" => ","

}

}

}

output

{

stdout {}

}

**Grok filter plugin**

* [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html) for the official documentation
* Logstash is shipped with the grok patterns [Refer Here](https://github.com/logstash-plugins/logstash-patterns-core/tree/main/patterns)
* For testing grok patterns we can use [Refer Here](http://grokdebug.herokuapp.com/)
* Ensure you go through GROK Basics [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html#_grok_basics)
* By using Grok Patterns we had parsed 55.3.244.1 GET /index.html 15824 0.043 this into multiple fields by using expression %{IP:clientip}%{SPACE}(?<method>\w+)%{SPACE}%{UNIXPATH:path}%{SPACE}%{NUMBER:size}%{SPACE}%{NUMBER:time} and the result was

{

"clientip": [

[

"55.3.244.1"

]

],

"IPV6": [

[

null

]

],

"IPV4": [

[

"55.3.244.1"

]

],

"SPACE": [

[

" ",

" ",

" ",

" "

]

],

"method": [

[

"GET"

]

],

"path": [

[

"/index.html"

]

],

"size": [

[

"15824"

]

],

"BASE10NUM": [

[

"15824",

"0.043"

]

],

"time": [

[

"0.043"

]

]

}

**Share this:**

JULY 21, 2022

DevOps Classroomnotes 21/Jul/2022

**Grok Patterns Contd**

**Regular Expressions**

* Characters
  + Escape character: \
  + Any character: .
  + Digit: \d
  + Not a digit: \D
  + Word character: \w
  + Not a word character: \W
  + Whitespace: \s
  + Not whitespace: \S
  + Word boundary: \b
  + Not a word boundary: \B
  + Beginning of a string: ^
  + End of a string: $
* Groupings
  + Matches characters in brackets: [ ]
  + Matches characters not in brackets: [^ ]
  + Either or: |
  + Capturing group: ( )
* Quantifiers
  + 0 or more: \*
  + 1 or more: +
  + 0 or 1: ?
  + An exact number of characters: { }
  + Range of number of characters: {Minimum, Maximum}
* refer to the classroom video for examples

**Share this:**

JULY 22, 2022

DevOps Classroomnotes 22/Jul/2022

**Standard Logs and their grok patterns**

**Apache Log:**

* Sample log: [Refer Here](https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs) for the whole file

83.149.9.216 - - [17/May/2015:10:05:03 +0000] "GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1" 200 203023 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_9\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"

* For apache log format [Refer Here](https://httpd.apache.org/docs/2.4/logs.html)

83.149.9.216 - - [17/May/2015:10:05:03 +0000] "GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1" 200 203023 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_9\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"

83.149.9.216 => host

17/May/2015:10:05:03 +0000 => timestamp

METHOD => GET

/presentations/logstash-monitorama-2013/images/kibana-search.png => PATH

HTTP/1.1 => http version

200 => Status

203023 => size

Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_9\_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36 => client info (User Agent)

http://semicomplete.com/presentations/logstash-monitorama-2013/" => Referer

* Now lets build a pipeline which reads logs from stdin and shows all the necessary fields on the stdout and filter to create fields using GROK

input

{

stdin {}

}

filter

{

grok

{

match => {

"message" => "%{COMBINEDAPACHELOG}"

}

}

}

output

{

stdout {}

}

\* Note: [Refer Here](https://github.com/logpai/loghub) for the logs of various applications

**SSH Log**

* Sample log

Dec 10 06:55:46 LabSZ sshd[24200]: reverse mapping checking getaddrinfo for ns.marryaldkfaczcz.com [173.234.31.186] failed - POSSIBLE BREAK-IN ATTEMPT!

* [Refer Here](https://github.com/logpai/loghub/blob/master/OpenSSH/SSH_2k.log) for the log file

%{MONTH:month}%{SPACE}%{MONTHDAY:day}%{SPACE}%{TIME:time}%{SPACE}%{WORD:host}%{SPACE}%{WORD}(?:\[)%{NUMBER:pid}(?:...)%{GREEDYDATA:message}

**Share this:**

JULY 23, 2022

DevOps Classroomnotes 23/Jul/2022

**Log Monitoring with Elastic Stack**

**Scenario: Monitoring Apache Logs**

* Basic Structure for Log Monitoring using Elastic Stack
* Lets setup the apache server

sudo apt update

sudo apt install apache2 -y

* Once the apache server is setup to generate the traffic we have created the small script

#!/bin/bash

while true

do

SERVER\_IP\_ADDRESS='54.244.44.152'

curl "http://${SERVER\_IP\_ADDRESS}"

sleep 1s

curl "http://${SERVER\_IP\_ADDRESS}"

sleep 1s

curl "http://${SERVER\_IP\_ADDRESS}/test.html"

sleep 1s

done

* Setting up logstash: [Refer Here](https://www.elastic.co/guide/en/logstash/current/installing-logstash.html)
* Now we need to create the logstash pipeline to recieve the logs from beats and send it to the elastic search and also transfom the apache logs
* The untested pipeline for transforming apache logs is as shown below

input

{

beats

{

port => 5044

}

}

filter

{

grok

{

match => { "message" => "%{COMBINEDAPACHELOG}" }

}

}

output

{

elasticsearch

{

cloud\_auth => "elastic:vu2fraeXw6bLb7DuDF6w0U3A"

hosts => ["https://qt-elastic.es.us-central1.gcp.cloud.es.io"]

index => "apache-%{+yyyy.MM.dd}"

}

}

* Now save this as apache.conf and save it in /etc/logstash/conf.d/ and then enable and start logstash

sudo systemctl enable logstash

sudo systemctl start logstash

* Now we need to setup beats which reads the logs written to file /var/logs/apache2/access.log
* Elastic stack has different beats
* For our scenario, we use file beat
* [Refer Here](https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation-configuration.html) for the installation
* Installation using APT [Refer Here](https://www.elastic.co/guide/en/beats/filebeat/current/setup-repositories.html)
* [Refer Here](https://www.elastic.co/guide/en/beats/filebeat/current/configuring-howto-filebeat.html) for configuring file beats
* Create a Dataview in kibana for the apache-\*
* Lets search for the logs with status code 200 and 404 and save the query
* [Refer Here](https://github.com/asquarezone/ElasticStackZone/commit/775a563ac52d42d3e1785d0bae040c19bb0ddb91) for the changeset containing the polling script and apache config for the logstash.

**Share this:**