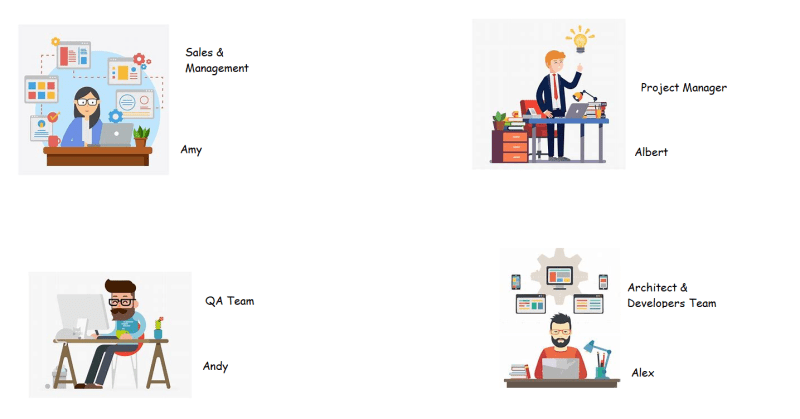
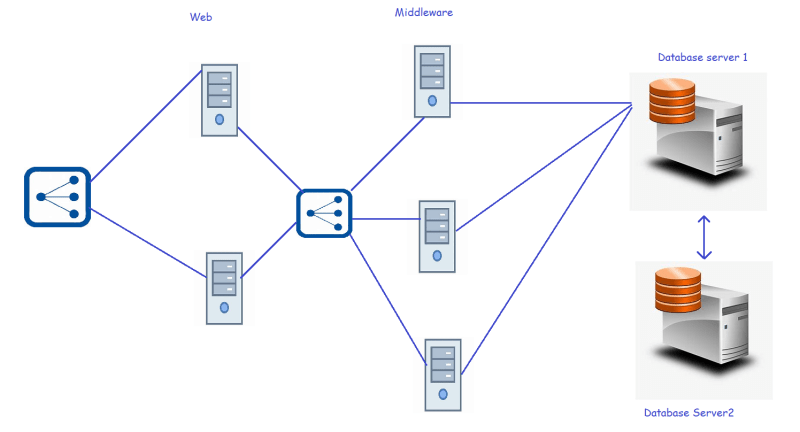
SEPTEMBER 14, 2021

DevOps Classroom Series – 14/Sept/2021

**Story of an Organization**

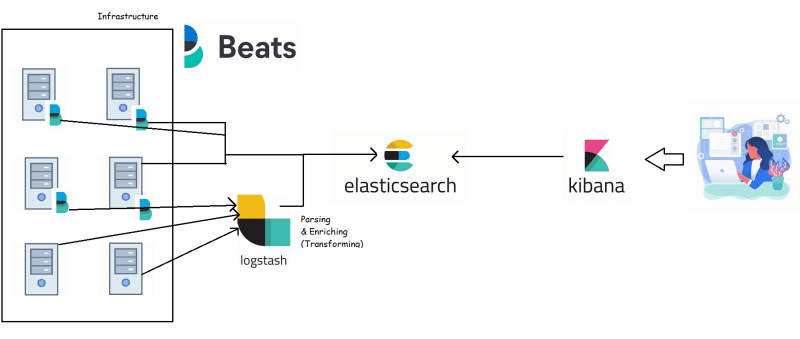
* LearningThoughts (a ficticious company) is having an application lt-hrms which is a human resource management system and is used by multiple organizations
* For this Learning Thoughts is hosting the application lt-hrms and they have the following team 
* Customers of LT are 
* The architecture of the application is as follows 
* Learning Thoughts updates a new release in every two weeks
* It is often observed that the following issues occur randomly
  + functionality stops working
  + Some server disks get filled up and there will no space left so servers donot respond
  + In Some server CPU utilization is above 95% all the time and users experience slow/unresponsive behavior
  + And many more…
* During Peak times, some users are facing request timed out errors etc.
* Hardware/Network failures happen randomly.
* Now We need to find a solution
  + to stop as many failures as possible from occuring
  + In the case of failures to resolve as early as possible
* We need to have monitoring in place to
  + monitor systems (Whether they are up or not)
  + monitoring health of your application
  + Monitor system resources
    - CPU
    - Memory
    - Storage
    - Network
* Applications generally create logs which donot have any standard approach. Reading text is tricky and creating meaningful information from text is quite difficult, so in majority of the case, we would use humans to find issues by going through logs
* So learning thoughts have decided that they would use a monitoring system which can not only read metrics but also parse log files and also helps in finding error patterns in logs.
* We would try to understand on resolving/identifying/trouble shooting failures with Elastic Stack (Which can do monitoring, APM, Log parsing, Alerting…)

**Share this:**

SEPTEMBER 15, 2021

DevOps Classroom Series – 15/Sept/2021

**Elastic Stack – Introduction**

* Elastic Stack is a rich ecosystem of components serving as full search adn analytics stack. The main components of Elastic Stack are
  + Elastic Search
  + Logstash
  + Kibana
  + Beats
  + X-Pack
* Elasticsearch is at the heart of the Elastic Stack providing storage, search and analytics capabilities.
* Kibana is referred as user interface for Elastic Stack with great visualization capabilities
* Logstash and Beats help the data into Elastic stack
* X-Pack provides features including alterting, security, graph & machine learning to make Elastic Stack production ready 

**Elastic Search**

* Elastic Search intro by official documentation is as follows "Elastic search is the distributed search and analytics engine at the heart of the Elastic Stack. Elastic search is where the indexing, search, and analysis magic happens. Elastic search provides near real-time search and analytics for all types of data. Whether you have structured or unstructured text, numerical data, or geospatial data, Elastic search can efficiently store and index it in a way that supports fast searches. You can go far beyond simple data retrieval and aggregate information to discover trends and patterns in your data. And as your data and query volume grows, the distributed nature of Elastic search enables your deployment to grow seamlessly right along with it."
* Elastic stack is built on the radically different technology ‘Apache Lucene’
* Key Benefits of Elastic Search
  + Schemaless, document-oriented
  + Searching
  + Analytics
  + Rich client library support and the REST API
  + Easy to operate and Easy to Scale
  + Near real time
  + Lightning-fast
  + Fault-tolerant
* Exercise: [Refer Here](https://www.youtube.com/watch?v=ggOmHlnhPaM&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=15) to this video to understand JSON and YAML

**Schema less and document oriented**

* Elastic search does not impose a strict structure on your data; you can store any Json documents.
* These JSON documents are first-class citizens of Elastic search as opposed to rows and columns in a relational database

**Share this:**

SEPTEMBER 16, 2021

DevOps Classroom Series – 16/Sept/2021

**Schemaless and document-oriented**

* Elastic search doesnot impose a strict structure on your data, you can store any json documents.
* JSON documents are first class citizens of Elastic search (as opposed to rows and columns in the Relational database).
* A document in Elastic Search is roughly equivalent to record in relational database table
* Traditional databases require a schema to defined beforehand
* JSON documents naturally support dynamic data

//first record

{

"name": "John Doe",

"email": "johndoe@gmail.com",

"address": {

"city": "New York"

}

}

//second record

{

"email": "jane\_doe@gmail.com",

"age": 45,

"address": {

"country": "USA",

"zipcode": "10010"

}

}

**Searching capabilities**

* The core strength of Elastic Search lies in its text-processing capabilities.
* Elastic search is great at searching especially full-text searches
* Full text search means searching through all the terms of the documents available in database, This requires entire content of all documents to be parsed and stored before hand.
* When you want to perform a search similar to Google search on your own data, Elastic search is your best bet, you can index emails, text documents, pdf files, web pages
* At a high level, Elastic search breaks up text data into terms and makes every term searchable by building lucene indexes

**Analytics**

* Elastic search supports wide variety of aggregations for analytics

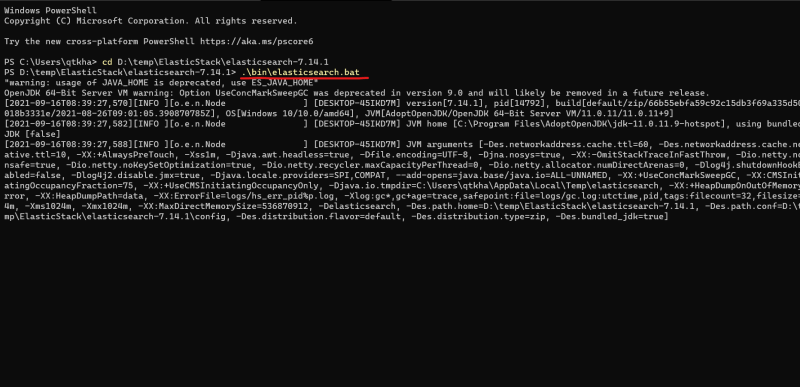
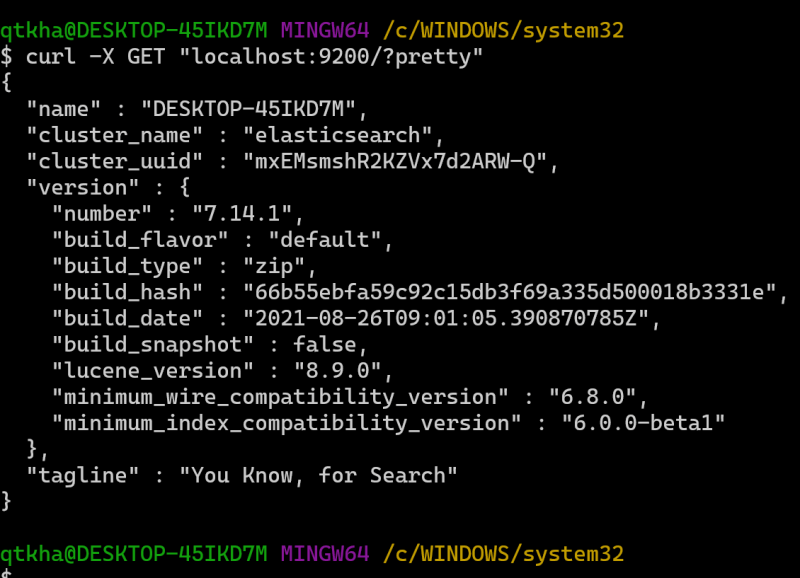
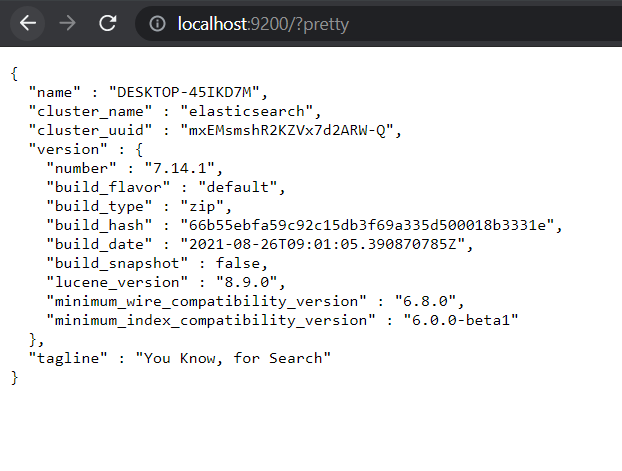
**Rich Client Library Support and REST API**

* Elastic search has a very rich client library support to access from languages like JAVA, C#, python, JavaScript, Ruby & more
* Elastic search has a very rich REST API (Representational state transfer) which works on HTTP protocol.

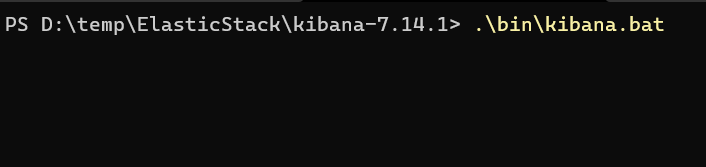
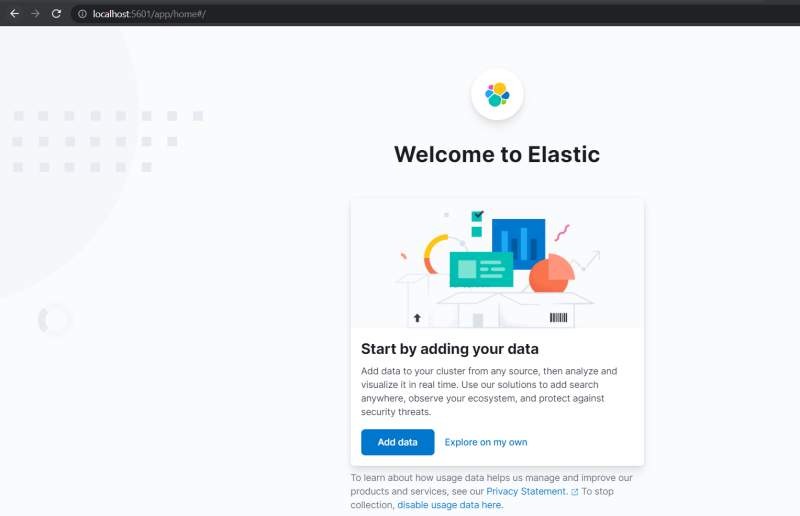
**Easy to Operate and easy to Scale**

* Elastic search can run on single node and easily scale out to hunderds of nodes
* Unlike most traditional database that only allow vertical scaling, Elastic search can be scaled horizontally
* The client application doesn’t need to change wheter it is running against a single node or a hundred node cluster
* Elastic search clusters can keep running even when there are hardware failures such as node failure, network failure etc i.e. Elastic search is designed to scale and is fault-tolerant

**Installing Elastic Search**

* [Refer Here](https://www.elastic.co/downloads/elasticsearch) to download elastic search
* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html#install-elasticsearch) for the installation of elastic search
* For understanding features of elastic search lets try to run elastic search locally
  + [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/zip-windows.html)
* Unzip the downloaded file and do the following 
* Run the command curl [http://localhost:9200?pretty](http://localhost:9200/?pretty) from git bash 
* Open browser and navigate to [http://localhost:9200?pretty](http://localhost:9200/?pretty) 

**Install Kibana**

* Download kibana zip from [Refer Here](https://www.elastic.co/cn/downloads/kibana)
* Extract the file and RUN bin\kibana.bat 
* Now open [http://localhost:5601](http://localhost:5601/) in your browser 

**Lab Setup**

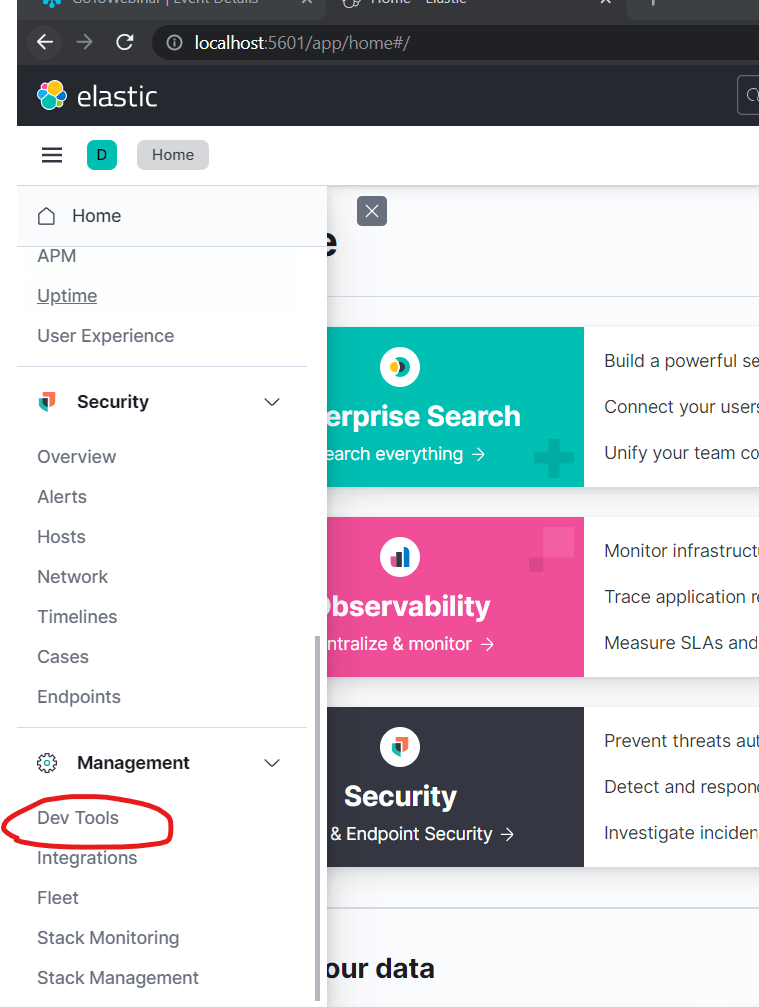
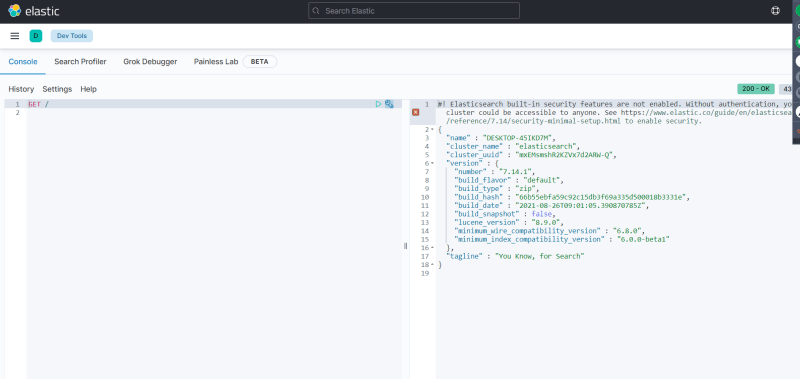
* Installing necessary softwares on Windows [Refer Here](https://www.youtube.com/watch?v=mRILfUNbsIo&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=14)
* Install Windows Terminal [Refer Here](https://www.youtube.com/watch?v=qLVn2EvPsYc&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=11)

**Share this:**

SEPTEMBER 17, 2021

DevOps Classroom Series – 17/Sept/2021

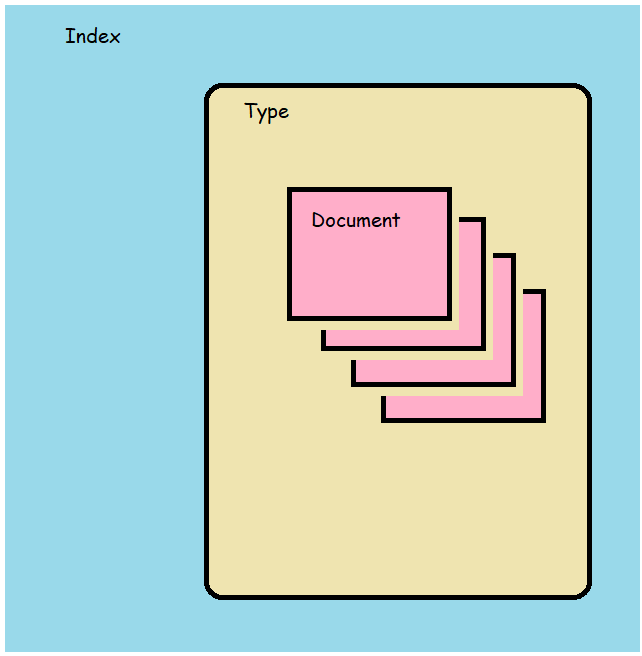
**Using Kibana Console**

* To communicate with Elastic Serach we need to use REST API
* REST stands for Representational State Transfer. Its an architectural style that is used to make system inter operative and interact with each other. REST has evolved over HTTP protocol.
* HTTP protocol supports different methods
  + GET
  + POST
  + PUT
  + DELETE
  + HEAD
* Kibana Console gives us a Console UI which is part of developer tools which makes it easier to interact with Elastic Search
* So we will be using Kibana Dev Tools to understand Elastic Search  

**Core Concepts of Elastic Search**

* The JSON documents in Elastic Search are organized as Elastic Search is a document-oriented store.
* Following are core concepts of Elastic search
  + Indexes
  + Types
  + Documents
  + Clusters
  + Nodes
  + Shards and replicas
  + Mappings and Types
  + Inverted Indexes

**Index**

* An index is a container that stores and manages documents of single type in Elastic Search 

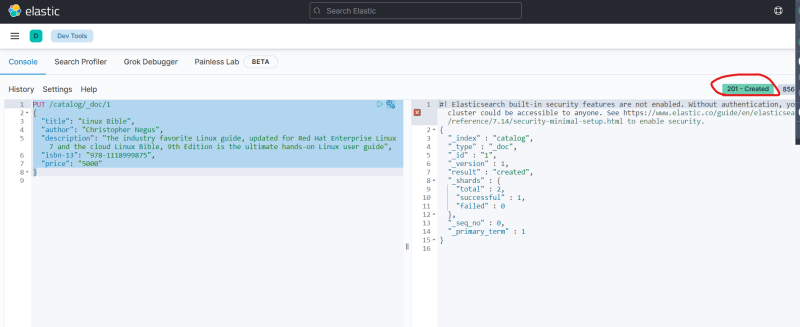
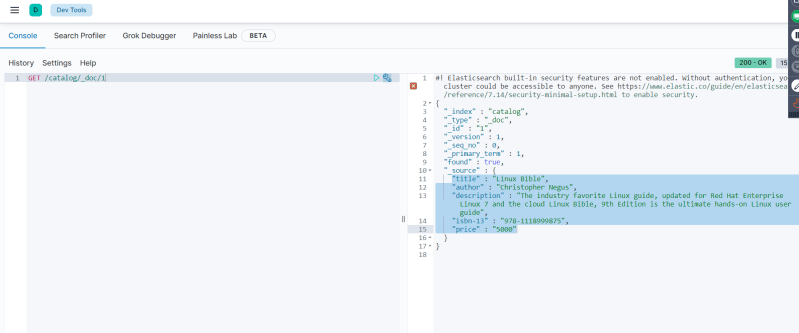
**Type**

* In practice, we should avoid mixing different entitites such as customers and products into single type, It makes sense to store them in seperate types with seperate Entities

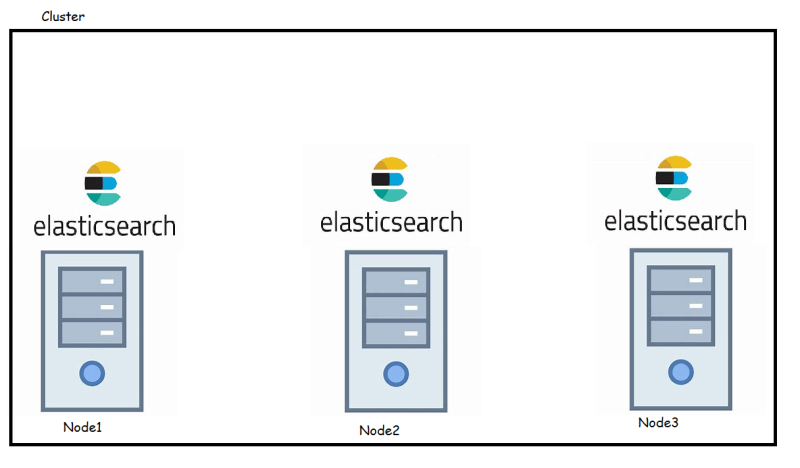
**Documents**

* Documents contains multiple fields & each field in JSON document is of particular type
* In addition to the fields sent by user in the document, Elastic search maintains internal meta fields
  + \_id: This is unique identifier of the document within a type
  + \_type: This field contains the type of the document
  + \_index: This field contains the index name of the document

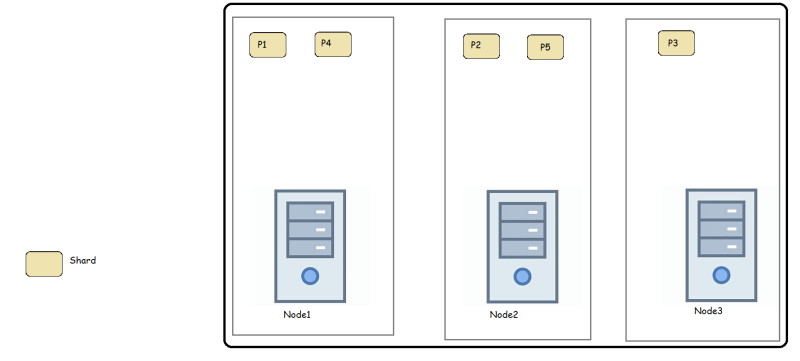
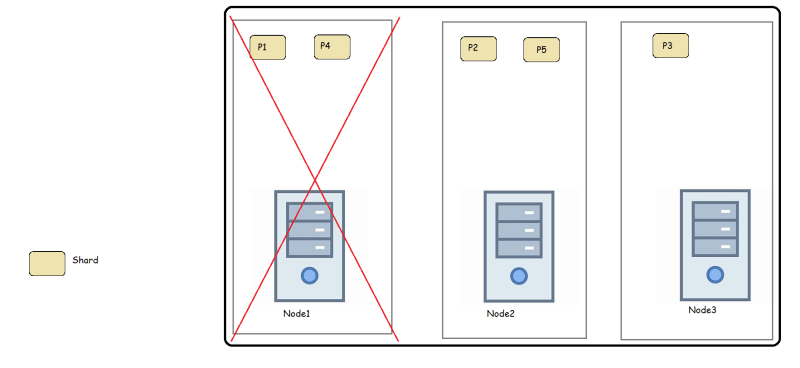
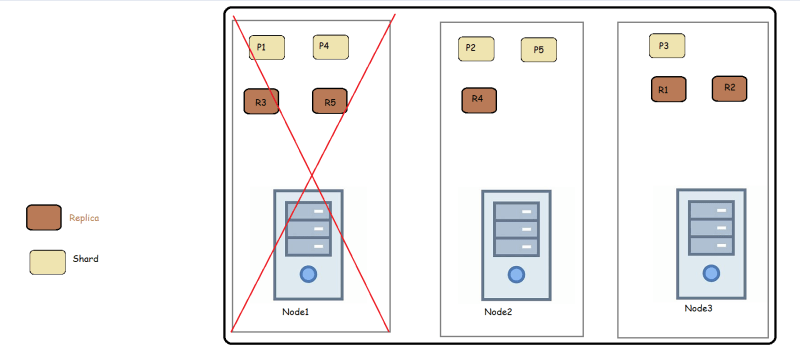
**Scenarios**

* Create a new index and document 
* Get the document created in the index 
* Note: HTTP Status codes [Refer Here](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)

**Nodes and Clusters**

* Elastic search is a distributed system and it consits of multiple processes running on different machines that communicate
* Each machine which runs elastic search is a Node
* Every node is associated with unique id and name
* Node details can be configured in elasticsearch.yml
* A cluster is formed by one or more nodes. Every Elastic search node is always part of cluster.
* By default, every Elastic search node tries to join a cluster with the name Elasticsearch
* A cluster consists of multiple nodes, where each node takes the responsibility for storing, managing its share of data.
* One cluster can host one or more indexes. 

**Shards and replicas**

* Lets try to understand what shard is, An index contains documents & Shards help in distributing an Index over cluster.
* Shards help in dividing the documents of single index over multiple nodes (Shards split data of single index over the cluster), hence allowing the storage, memory and processing capacities of cluster to be utilized
* The process of dividing data among shards is called sharding
* By default every index is configured to have five shards in Elastic Search
* While creating index we can specify number of shards 
* Now lets assume Node 1 is down 
* Distributed systems such as Elastic search are expected to run inspite of hardware failures. This issue is address by replica shards or replicas.
* Each shard in index is configured to have zero or more replica shards. Replica Shards are extra copies of the orginal shards and they created for high availability of data 
* As you can observe in the image even if the Node 1 is down, then the data is getting served from other nodes with replicas ()

**Share this:**

SEPTEMBER 19, 2021

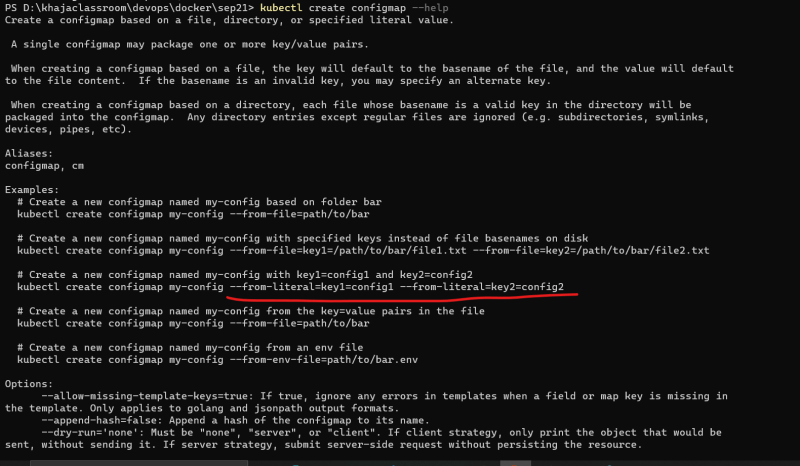
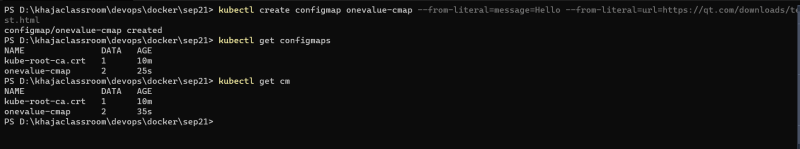
DevOps Classroom Series – 18/Sept/2021

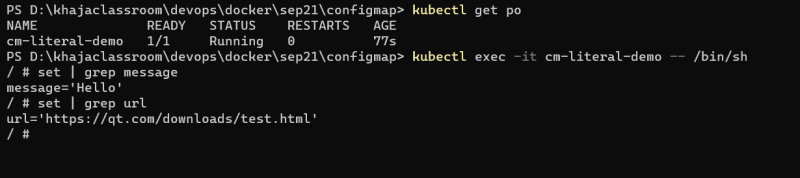
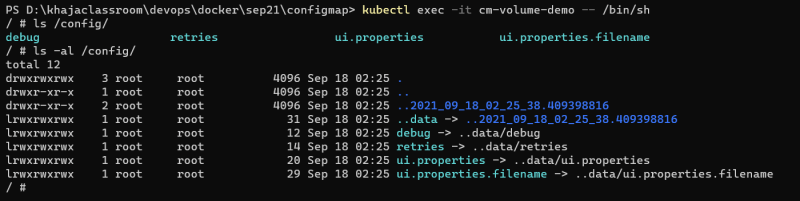
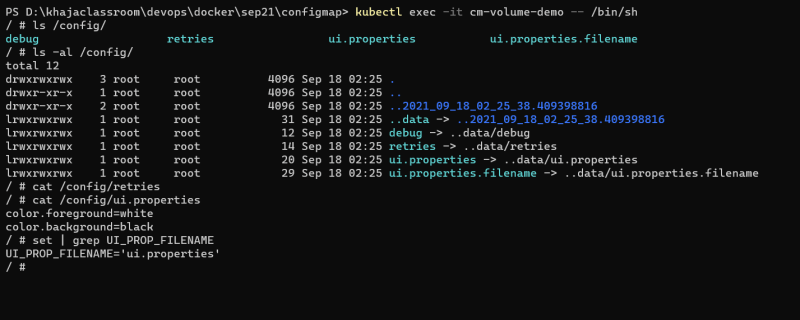
**Config Maps**

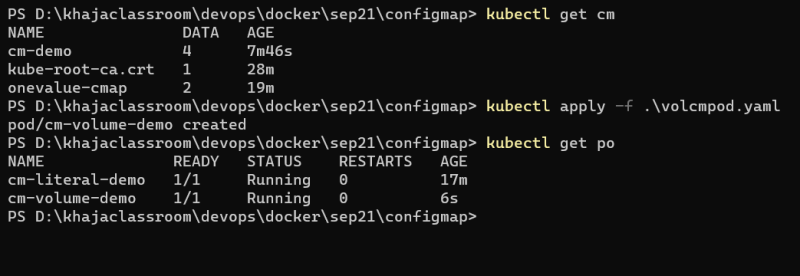
* One way to think of ConfigMap is as a k8s object that defines a small file system.
* Another way is a set of variables that can be used when defining the environment or command line for your container
* In config map we generally define application related configuration data. We can create ConfigMap through a literal value or from a file
* Creating config map from literal values and mount it to the Pod

kubectl create configmap --help

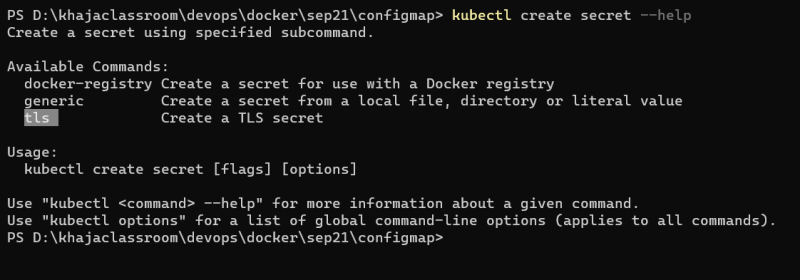
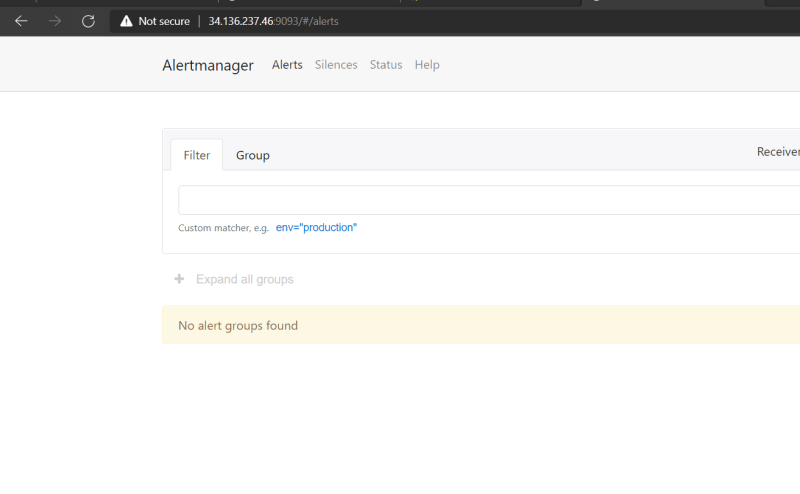
kubectl create configmap onevalue-cmap --from-literal=message=Hello --from-literal=url=https://qt.com/downloads/test.html

* Now lets create a Pod and mount the config maps as environment variables [Refer Here](https://github.com/asquarezone/DockerZone/commit/064895752159014f104e51279048a62d1c2d0a6d) for the changeset and apply this manifest
* Now lets login into the container and check environment variables 
* Now lets create a configmap as Manifest [Refer Here](https://github.com/asquarezone/DockerZone/commit/88d043b981cffcdab2563532bf64671686ce942c) for the changeset and apply the manifests  
* When you change the config maps and want to apply it your deployment use kubectl rollout restart [Refer Here](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#-em-restart-em-)



**Secrets**

* They are much like config map, but in secrets values are base64 encoded  
* Lets create the following secrets

kubectl create secret generic my-secret --from-literal=username=qtdevops --from-literal=password=qtdevops

* [Refer Here](https://github.com/asquarezone/DockerZone/commit/0eb18ccd8c360b5473e8121ddc5814457ccf9449) for the changeset

**Stateful Sets**

* Kubernetes statefulsets are useful for running things in cluster e.g hadoop cluset, mysql cluster where each Pod is expected to have its own storage
* In Statefulset also we would have replicas, but each replica will have its own Persitent volume claim placed

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

clusterIP: None

selector:

app: nginx

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx # has to match .spec.template.metadata.labels

serviceName: "nginx"

replicas: 3 # by default is 1

template:

metadata:

labels:

app: nginx # has to match .spec.selector.matchLabels

spec:

terminationGracePeriodSeconds: 10

containers:

- name: nginx

image: k8s.gcr.io/nginx-slim:0.8

ports:

- containerPort: 80

name: web

volumeMounts:

- name: www

mountPath: /usr/share/nginx/html

volumeClaimTemplates:

- metadata:

name: www

spec:

accessModes: [ "ReadWriteOnce" ]

storageClassName: "my-storage-class"

resources:

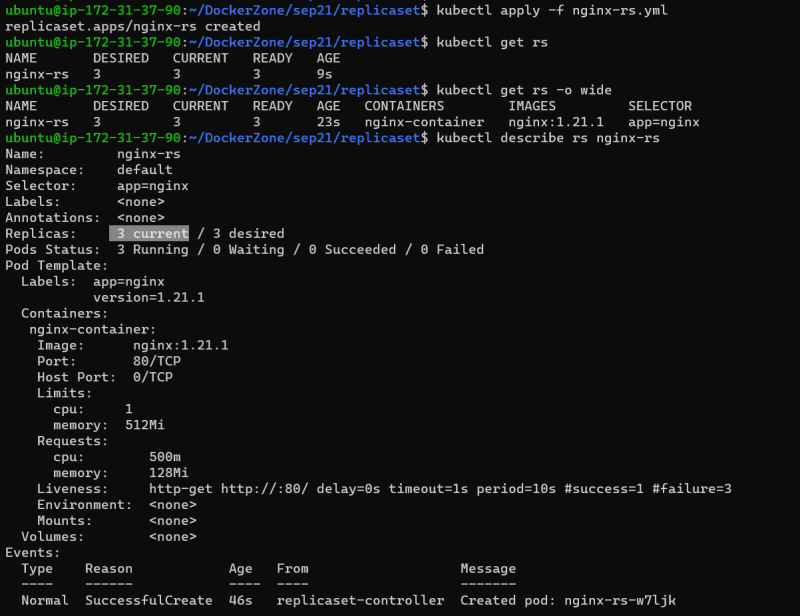
requests:

storage: 1Gi

**Elastic Kubernetes Services (EKS)**

* To setup the EKS [Refer Here](https://docs.aws.amazon.com/eks/latest/userguide/create-cluster.html)
* In this case i would be using the eksctl way of bringing up kubernetes cluster
* To install eksctl [Refer Here](https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html)
* [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html) to create and configure iam user
* Now lets create a k8s cluster

eksctl create cluster --name my-cluster --version 1.21 --with-oidc --without-nodegroup



* Lets add nodes to the eks cluster

eksctl create nodegroup `

--cluster my-cluster `

--region us-west-2 `

--name my-mng `

--node-type t2.large `

--nodes 2 `

--nodes-min 1 `

--nodes-max 2 `

--ssh-access `

--ssh-public-key docker

* Deleting the node group

eksctl delete nodegroup --cluster my-cluster --name my-mng

* Deleting the cluster

eksctl delete cluster --name my-cluster

**Exercise**

* [Refer Here](https://directdevops.blog/2019/11/02/deploying-the-docker-application-and-mysql-with-volume-support-into-kubernetes-from-code-to-docker-registries-like-acr-ecr-and-then-to-eks-aks/) for the article which is about the journey of application from containers to k8s

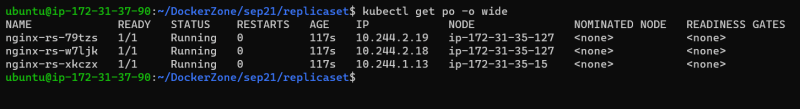
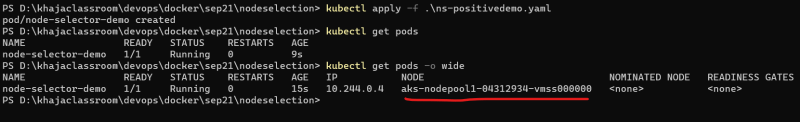
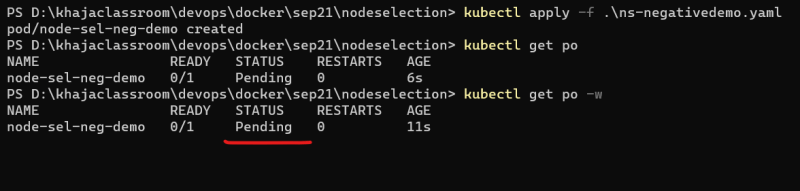
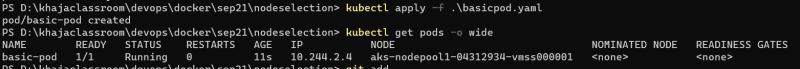
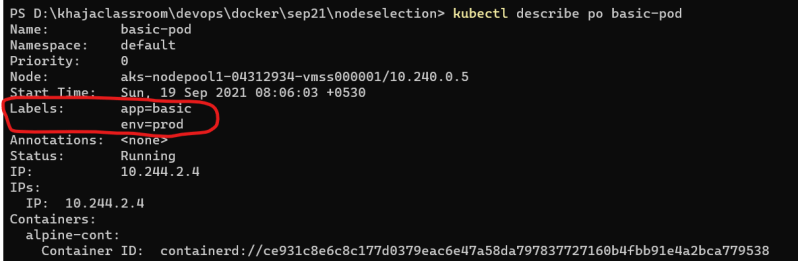
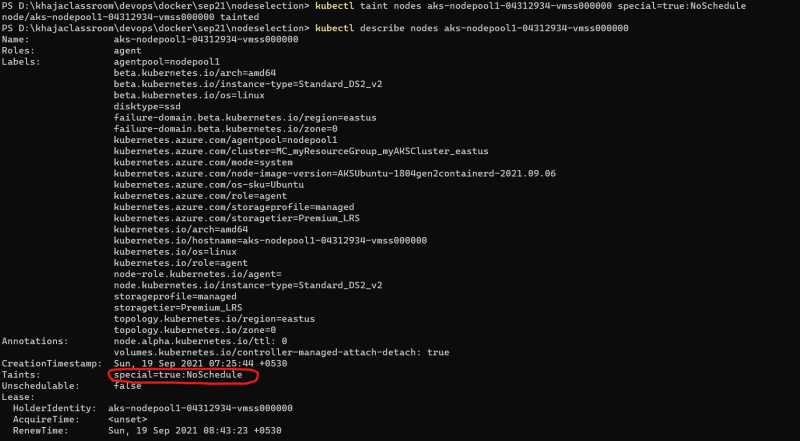
**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2021/09/19/devops-classroom-series-18-sept-2021/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2021/09/19/devops-classroom-series-18-sept-2021/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2021/09/19/devops-classroom-series-18-sept-2021/?share=jetpack-whatsapp&nb=1)

SEPTEMBER 19, 2021

DevOps Classroom Series – 19/Set/2021

**Assigning Pods to nodes**

* Generally scheduler will automatically do the reasonable plcement of your pods accross nodes, but there are some circumstances where you want to control which node the pod deploys
* Node Selector:
  + *nodeSelctor* is the simplest for node selection constraint.
  + This specifes a map of key value pairs.
  + For the pod to be elibigle to run on a node, node musht have indicated key value pairs as labels
  + Step 1: Attach label to nodes 
  + Step 2: Attach nodeSelector field to Pod Manifest [Refer Here](https://github.com/asquarezone/DockerZone/commit/6fa764f1282930c9754d2dc2c47f39b1e4fccad0)
  + Now apply the configuration and verify the pod schedule details 
  + Now lets try to do a negative scenario, lets try to use nodeSelector with labels that donot exist and see what happens
  + Pod spec [Refer Here](https://github.com/asquarezone/DockerZone/commit/971afc2a69fe06586b10af97e812b82a258642ad)
  + After applying we observe the pods will not be schedule by scheduler as no node has matching labels 
* Nodes in k8s come with pre-populate standard set of labels [Refer Here](https://kubernetes.io/docs/reference/labels-annotations-taints/)
* Affinity and anti-affinity:
  + The affinity/anti-affinity feature, greately expands types of constraints. The key enhancements from nodeSelector are
    - The affinity/anti-affinity language is more expressive, it offers more matching rules besides exact mathc created
    - You can indicate that a rule is soft or preference rather than hard requirement, so if the scheduler cannot statisfy, pods will still be scheduled
    - You can constraint aginst labels on other pods running on the node, rather than against labels of node itself, which allows rules about which pods can and cannot be located
  + Node affinity:
    - There are two types of node affinity
      * requiredDuringSchedulingIgnoredDuringExecution:
        + Hard requirement
      * prefferedDuringSchedulingIgnoredDuringExecution
        + Soft requirement
    - [Refer Here](https://github.com/asquarezone/DockerZone/commit/6ffe2d85cece08d16dc588a18b6d71d8073816f9) for the node affinity example
    - Created a basic pod with some labels [Refer Here](https://github.com/asquarezone/DockerZone/commit/385a64b41a8dc93489befd40ae1c04e312a7e47d)  
    - Write a Pod manifest
      * to schedule the new pod (pod-1) in the same node as basic-pod (podAffinity)
      * to schedule the new pod (pod-2) in the different node as basic-pod (podAntiAffinity)
* Taints and Tolerations:
  + Node affinity is property of Pods that attracts them to set of nodes.
  + Taints are opposite – they allow node to run a set of pods
  + Tolerations are applied to pods and allow the pods to schedule on the nodes with matching taints.
  + Taints and Tolerations work together to ensure pods are not scheduled into inappropriate nodes.
  + To apply a taint to the node kubectl taint nodes <node-name> key1=value1:NoSchedule
  + To remove the applied taint kubectl taint nodes <node-name> key1=value1:NoSchedule-
  + Lets apply a taint to any node 
  + Currently the following taints are supported
    - OutofDisk: node.kubernetes.io/out-of-disk
    - MemoryPressure: node.kubernetes.io/memory-pressure
    - DiskPressure: node.kubernetes.io/disk-pressure
    - PIDPressure: node.kubernetes.io/pid-pressure
  + Generally in some case it is advised to write tolerations based on node conditons
  + [Refer Here](https://github.com/asquarezone/DockerZone/commit/b064e8dad65ad5784f8970f2c8fd0ce941220fbe) for the yaml manifests

**Helm**

* To install softwares in linux we use packaging tools such as apt,yum, in mac we have brew and in windows we have choco. These software are called as package managers
* Helm is an opensource packaging tool for k8s to deploy and manage the lifecycle of your application
* Installing Helm [Refer Here](https://helm.sh/docs/intro/install/)
* In a very simple terms helm charts make the k8s manifest dynamic
* consider the following two deployments
* deployment 1

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: myapp-deploy

spec:

replicas: 2

selector:

matchLabels:

app: myapp

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 50%

maxUnavailable: 50%

template:

metadata:

labels:

app: myapp

version: v3

spec:

containers:

- name: myapp-cont

image: shaikkhajaibrahim/myapp:3.0

ports:

- name: http

containerPort: 80

* the other deployment

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: otherapp-deploy

spec:

replicas: 3

selector:

matchLabels:

app: otherapp

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 50%

maxUnavailable: 50%

template:

metadata:

labels:

app: otherapp

version: v1

spec:

containers:

- name: otherapp-cont

image: nginx

ports:

- name: http

containerPort: 80

protocol: TCP

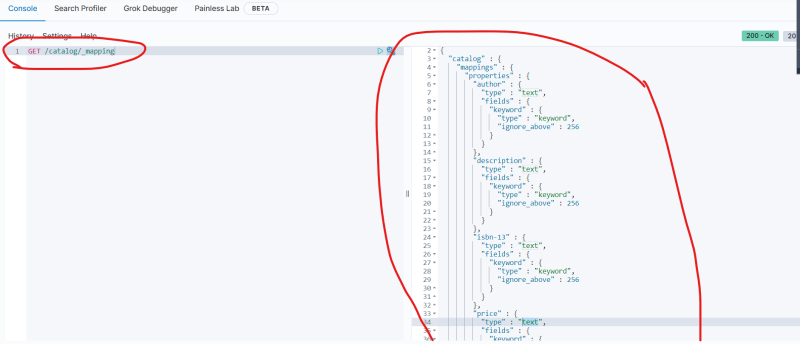
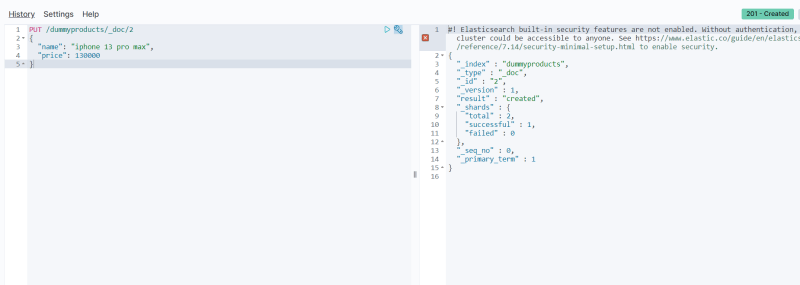
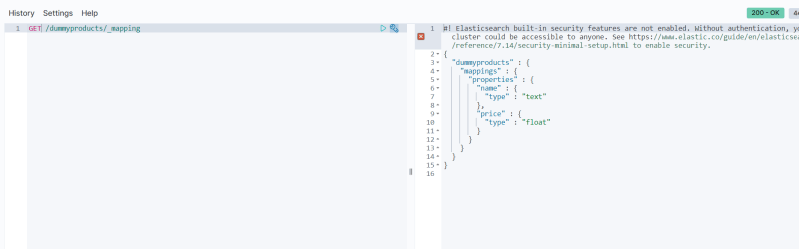
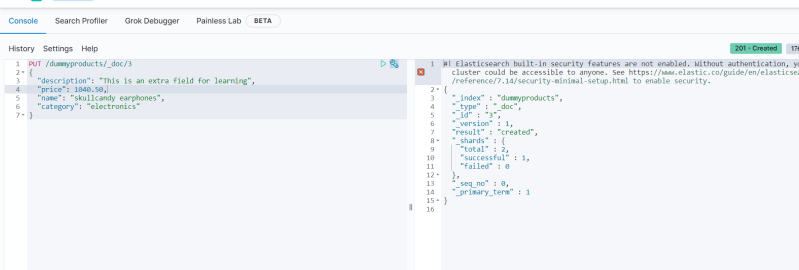
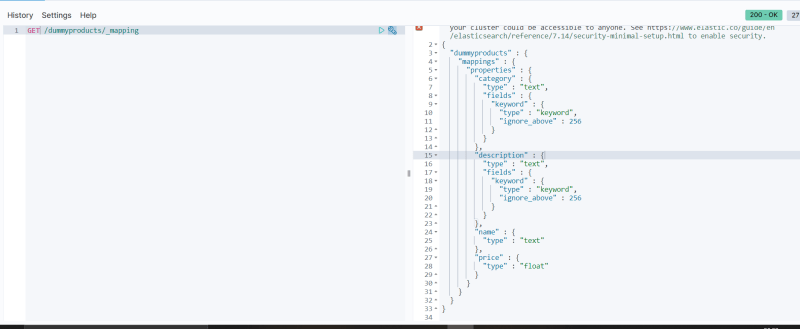
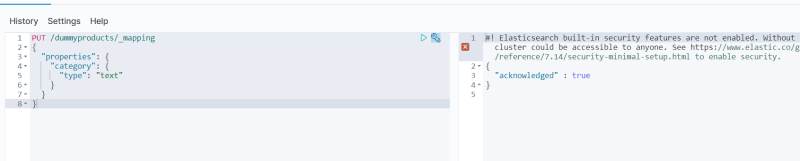
* Helm use go templates to parametrize and value.yaml file which will have values defined
* Try installing any chart [Refer Here](https://bitnami.com/stacks/helm)

**Share this:**

SEPTEMBER 20, 2021

DevOps Classroom Series – 20/Sept/2021

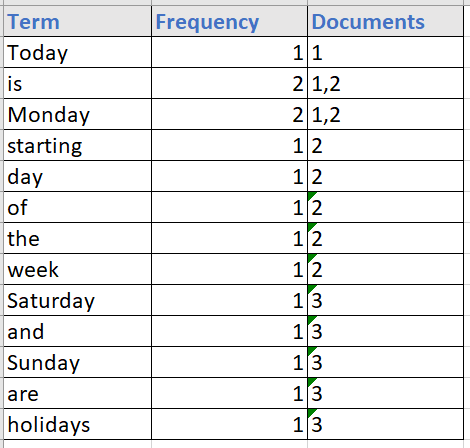
**Mappings and Datatypes**

* Elastic search supports wide variety of data types
  + Core datatypes:
    - String datatypes:
      * text: The text data type is useful for supporting full-text search for fields that contain a description or lengthy text values.
      * keyword: The keyword type enables analytics on string fields to support sorting, filtering and aggregations
    - Numeric datatypes:
      * byte (8), short (16),integer(32) and long(64)
      * float (32) and double (64)
      * half\_float (16)
      * scaled\_float
    - Date Datatype
      * date
    - Boolean datatype
      * boolean
    - Binary data type:
      * binary
    - Range datatypes
      * integer\_range, float\_range, long\_range, double\_range and date\_range
  + Complex datatypes
    - Array datatype
    - Object datatype
    - Nested datatype
  + Other datatypes:
    - Geo-point
    - Geo-shape
    - Ip datatype
* Get a mapping for the index 
* Lets create an index dummyproducts with some mapping   
* Now create a document in the dummy products with extra fields not specified in mapping 
* Now lets get mapping for dummy products 
* Lets try to update mappings [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html) 

**Inverted Index**

* An inverted index is the core data structure of Elastic search.
* Lets assume we have following documents created

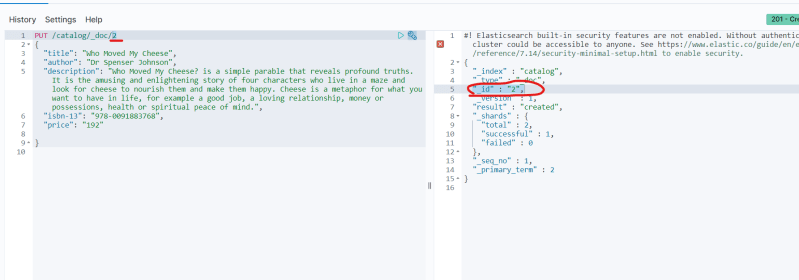
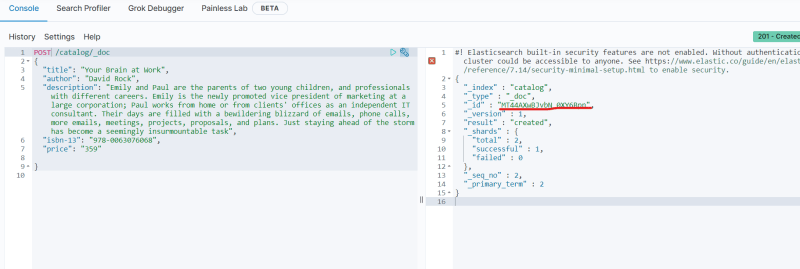
| **Document ID** | **Document** |
| --- | --- |
| 1 | Today is Monday |
| 2 | Monday is starting day of the week |
| 3 | Saturday and Sunday are holidays |

* Elastic search builds a data structure from the above three documents which is called as inverted index 
* When searching for terms in documents, it fast to locate the documents in which given term appears and inverted index is created.
* By default Elastic search builds an inverted index on all the fields in the document

**CRUD Operations**

* To understand how to perform CRUD operations lets try to understand the following APIs
  + Index API
  + Get API
  + Update API
  + Delete API

**Index API**

* In Elastic search terminology, adding a document to a type with in an index is called indexing operation.
* We have already learnt, indexing involves
  + adding the document to the index by parsing all the fields within the document
  + building an inverted index
* There are two ways we can index a document
  + Indexing a document with ID 
  + Indexing a document without provding ID 

**Share this:**

SEPTEMBER 21, 2021

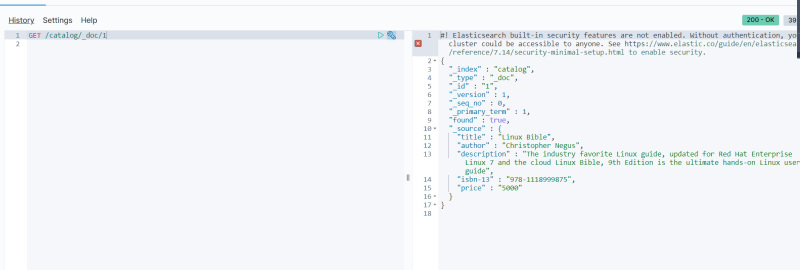
DevOps Classroom Series – 21/Sept/2021

**GET API**

* The Get API is used for retrieving a document where you already know ID

GET /catalog/\_doc/<doc-id>

GET /<index>/<type>/<id>



**UPDATE API**

* The Update API is useful for updating existing document with ID

POST /<index>/\_update/<id>

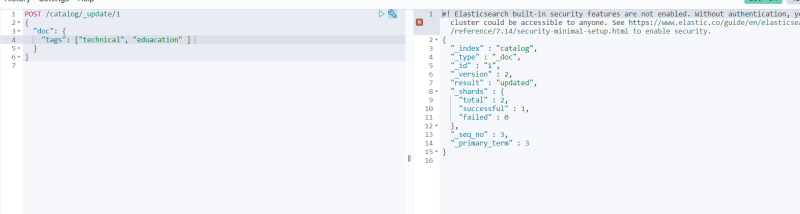
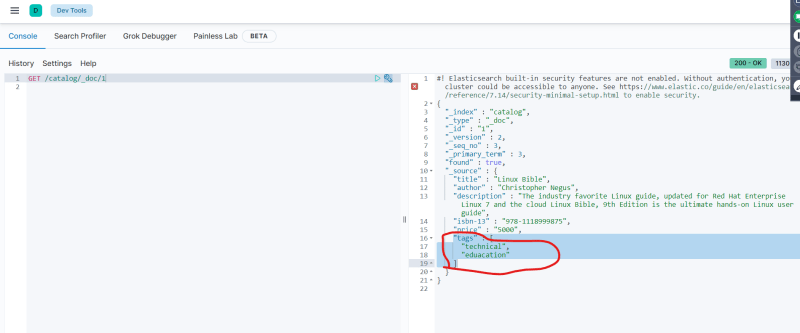
{

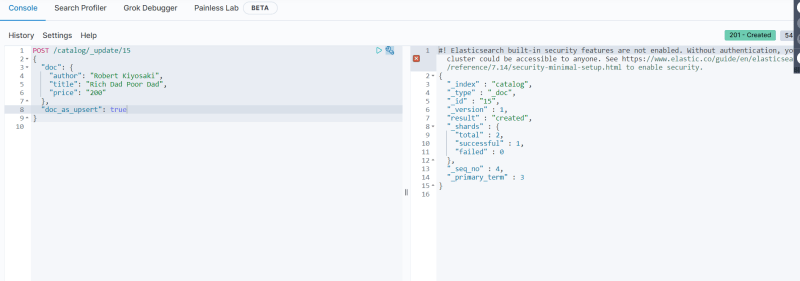
"doc" : {

"tags": ["technical", "eduacation" ]

}

}

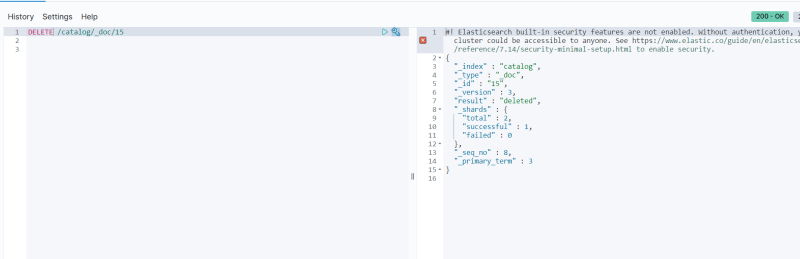
 

* Update the document if present or create a new document doc\_as\_upsert:true does this trick 
* Update the value of the document by its existing fields using script

**DELETE API**

* DELETE API lets you delete the document by ID

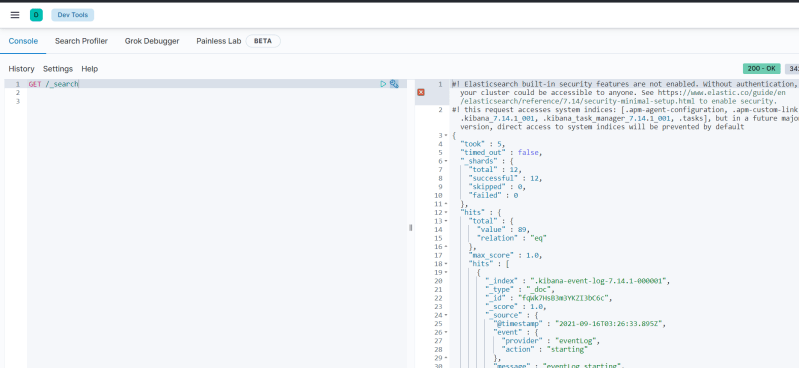
DELETE /<index>/<type>/<id>



**Dealing with multiple indexes**

* Operations such as search and aggregation can run against multiple indexes in same query
* The following query matches all documents

GET /\_search



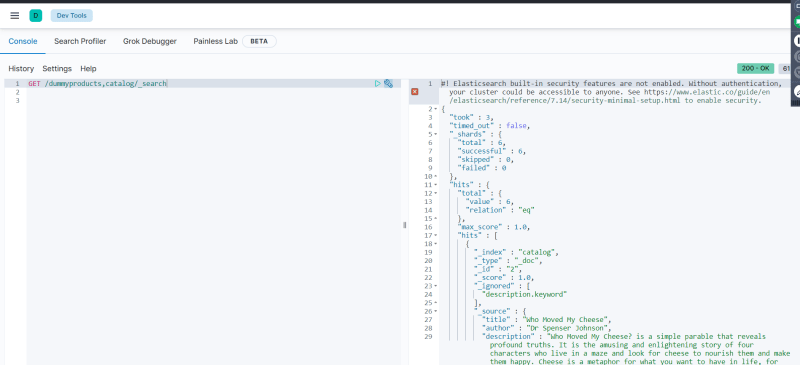
* Searching all documents in one index

GET /<index>/\_search



* Searching all documents in multiple indexes

GET /<index-1>,<index-2>..<index-n>/\_search



**Searching – What is Relavent**

* Text Analysis: All the fields that are of text type are analyzed by what is known as analyzer
  + The main task of analyzer is to take the value of field and break it down to terms
  + Analyzer performs the process of breaking up into terms
    - at the time of indexing
    - at the time of searching
  + Analyzer has the following components
    - Character filter: Zero or more
    - Tokenizer: Exactly one
    - Token filters: Zero or more
  + Elastic search ships with few built-in character filters which we can use or create our own analyzer. Elastic search ships with Mapping Char Filter
  + For example if you are indexing converstations (chats/emails) etc and then if you want to transfer emoticons into some text
    - should be translated to \_smile\_
    - should be translate to \_sad\_
  + This can be acheived through the character filter
* "char\_filter": {
* "my\_char\_filter" :{
* "type": "mapping",
* "mappings": [
* ":) => \_smile\_",
* ":( => \_sad\_"
* ]
* }
* }
  + [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-charfilters.html)
  + The responsibility of a tokenizer is to recieve a stream of characters and generate tokens. These tokens are used to build an inverted index (Token is roughly equivalent to word) [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html) for official docs

[Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenfilters.html) foSEPTEMBER 27, 2021

DevOps Classroom Series – 27/Sept/2021

**Queries in Elastic Search**

* The following structured or term level queries can be done in Elastic search
  + Range Query
  + Exists Query
  + Term Query
  + Terms Query
* Range Query on numeric types:
  + Lets try to create a query to get the products between price 1000 to 10000
* GET /flipkart/\_search
* {
* "query": {
* "range": {
* "discounted\_price": {
* "gte": 1000,
* "lte": 10000
* }
* }
* }
* }
* Range Query on dates

GET /flipkart/\_search

{

"query": {

"range": {

"fetched\_date": {

"gte": "15/03/2016",

"lte": "30/03/2016",

"format": "dd/MM/yyyy"

}

}

}

}

* Exists query

GET /flipkart/\_search

{

"query": {

"exists": {

"field": "fetched\_date"

}

}

}

* Full text search queries with match

GET /flipkart/\_search

{

"query": {

"match": {

"name": "Women's"

}

}

}

* Operator

GET /flipkart/\_search

{

"query": {

"match": {

"name": {

"query": "Women's FabHome",

"operator": "and"

}

}

}

}

* match phrase:

GET /flipkart/\_search

{

"query" : {

"match\_phrase" : {

"description": {

"query": "phone for all"

}

}

}

}

* Multple fields:

GET /flipkart/\_search

{

"query": {

"multi\_match": {

"query": "Women's",

"fields": ["name", "discounted\_price"]

}

}

}

* Bool query

GET /flipkart/\_search

{

"query": {

"bool": {

"must": [

{}

],

"should": [

{}

],

"filter": [

{}

],

"must\_not": [

{}

]

}

}

}

**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2021/09/27/devops-classroom-series-27-sept-2021/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2021/09/27/devops-classroom-series-27-sept-2021/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2021/09/27/devops-classroom-series-27-sept-2021/?share=jetpack-whatsapp&nb=1)

**Share this:**

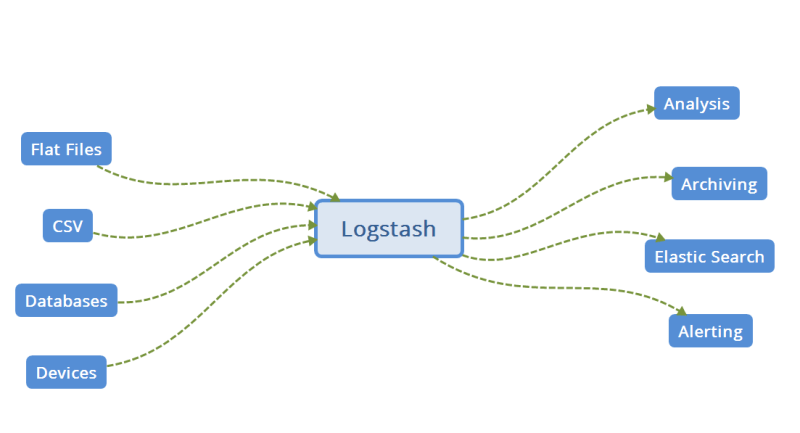
SEPTEMBER 28, 2021

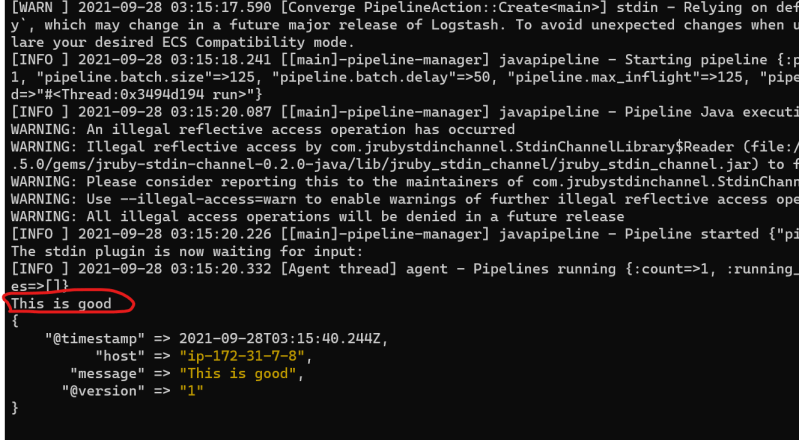
DevOps Classroom Series – 28/Sept/2021

**Analyzing Log Data**

* Log Analysis Challenges:
  + Logs are defined as records of incidents or observations and are generated by a wide variety of resources such as system, applications, devices and so on
  + Log is typically made of two things
    - timestamp
    - data
  + Logs are used for the following reasons
    - Troubleshooting
    - Auditing
    - Predective analytics
  + Some challenges with logs are
    - No common/consistent format
    - Logs are decentralized
    - Data is unstructured

**Logstash**

* This is very popular open source data collection engine with real-time pipelining capabilities. 
* Logstash has input, filter and output plugins & logstash does the work of ETL Engine by extracting information from input, transforming according to filter and loading the information to output plugins
* Features:
  + Pluggable Data pipeline architecture
  + Extensibility
  + Centralized data processing
  + Variety and Volume
* Lab Environment
  + We would create two linux vms
    - one vm for logstash
    - one vm for elastic search and kibana
  + Lets create an ubuntu 20 server and install logstash [Refer Here](https://www.elastic.co/guide/en/logstash/current/installing-logstash.html#_apt)
  + Once you install logstash the
    - /usr/share/logstash will be the folder containing executables
    - /etc/logstash will be folder for configuring logstash
  + Now lets try to run logstash by configuration input => stdin and output is stdout
* cd /usr/share/logstash
* sudo bin/logstash -e "input {stdin {} } output {stdout {} }"



* + Lets try to create a configuration for logstash datapipeline
    - input stdin
    - output a file

sudo bin/logstash -e "input {stdin {} } output { file { path => '/tmp/output'} }"

* + Running the logstash from command line for evey pipeline is not sensible, we would learn how to this as configuration files.

**Share this:**

SEPTEMBER 29, 2021

DevOps Classroom Series – 29/Sept/2021

**Logstash Configuration**

* The logstash pipeline is stored in a configuration file with .conf extension. The three sections of the configuration file are

input

{

}

filter

{

}

output

{

}

* In Each of section, we can have one or more plugin configuration
* Lets create a simple configuration file

input

{

stdin {}

}

filter

{

mutate {

uppercase => [

"message",

"host"

]

}

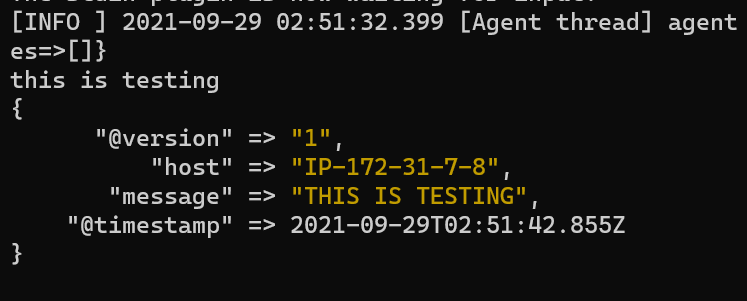
}

output

{

stdout {}

}



* So in the logstash we have plugins and plugins have properties which we can set

**Overview of Logstash Plugins**

* To view all the plugins installed in logstash

sudo /usr/share/logstash/bin/logstash-plugin list

* To install a plugin

sudo /usr/share/logstash/bin/logstash-plugin install <plugin-name>

* To update a plugin

sudo /usr/share/logstash/bin/logstash-plugin update <plugin-name>

* Lets Explore some of the poplular plugins
* File:
  + [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html) for the official documentation
  + Lets look at the following configuration
  + input
  + {
  + file
  + {
  + path => ["/var/log/\*", "/var/log/\*\*/\*.log"]
  + start\_position => "beginning"
  + exclude => ["\*.csv"]
  + discover\_interval => 20
  + stat\_interval => "500ms"
  + }
  + }
  + output
  + {
  + stdout {}
  + }
  + Try to create a logstash config which reads mysql logs stored /var/logs/mysql/\*.log

**Share this:**

SEPTEMBER 30, 2021

DevOps Classroom Series – 30/Sept/2021

**Grok Filter**

* This is powerful and often used plugin for parsing the unstructured data into structured data, which makes the data easily queryable.
* The general syntax of grok pattern is

%{PATTERN:FIELDNAME}

* By default groked field are string and can be cast into float or int

%{PATTERN:FIELDNAME:type}

* Logstash ships about 120 patterns by default [Refer Here](https://github.com/hpcugent/logstash-patterns/blob/master/files/grok-patterns)
* To work with grok pattern [Refer Here](https://grokdebug.herokuapp.com/)
* Converting log message in to multiple fields
* Sample 1
  + Logmessage: 2021-04-29 02:59:23.110 DEBUG 353 --- [nio-8080-exec-7] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to org.springframework.samples.petclinic.system.WelcomeController#welcome()
  + Grok pattern: %{TIMESTAMP\_ISO8601}%{SPACE}%{LOGLEVEL:level}%{SPACE}(?<thread>\d+\s+-\*\s\*\[[\b\w\*\d\*-]\*\]\s)(?<class>[\b\w\*\.]\*)%{SPACE}(?::\s\*)%{GREEDYDATA:logmessage}
* Sample 2: Apache log [Refer Here](https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs)
  + Log message: 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/&quot>; "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"
  + Grok pattern: %{COMBINEDAPACHELOG}
* Sample 3: mysql error logs
  + Log message: 060516 22:38:54 [ERROR] Fatal error: Can't open privilege tables: Table 'mysql.host' doesn't exist
  + Grok patterns: (?<month>\d\d)(?<day>\d\d)(?<year>\d\d)%{SPACE}%{TIME}%{SPACE}(?<error>\[\b\w+\]\s+)%{GREEDYDATA:errormessage}
* [Refer Here](https://coralogix.com/blog/a-practical-guide-to-logstash-parsing-common-log-patterns-with-grok/) for some other samples
* Now lets create a conf which reads a log from stdin for sample 1

input

{

stdin {}

}

filter

{

grok

{

match => { "message" => "%{TIMESTAMP\_ISO8601}%{SPACE}%{LOGLEVEL:level}%{SPACE}(?<thread>\d+\s+-\*\s\*\[[\b\w\*\d\*-]\*\]\s)(?<class>[\b\w\*\.]\*)%{SPACE}(?::\s\*)%{GREEDYDATA:logmessage}" }

}

}

output

{

stdout {}

}

* Now lets try to read apache logs from a file and output to stdout using grok

input

{

file

{

path => "/var/log/apache.log"

start\_position => "beginning"

}

}

filter

{

grok

{

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

}

}

output

{

stdout {}

}

* Also parse syslog messages

input

{

file

{

path => "/var/log/syslog\*"

start\_position => "beginning"

}

}

filter

{

grok

{

match => { "message" => "%{SYSLOGBASE}%{GREEDYDATA:logmessage}"}

}

}

output

{

stdout {}

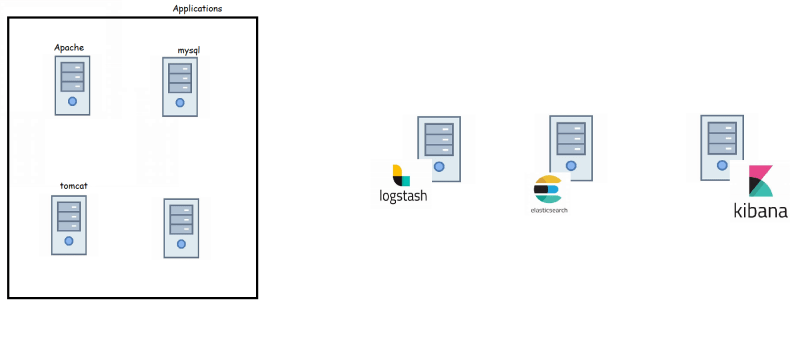
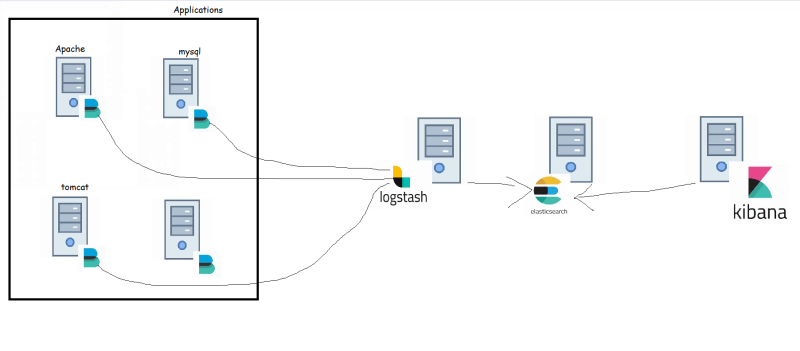
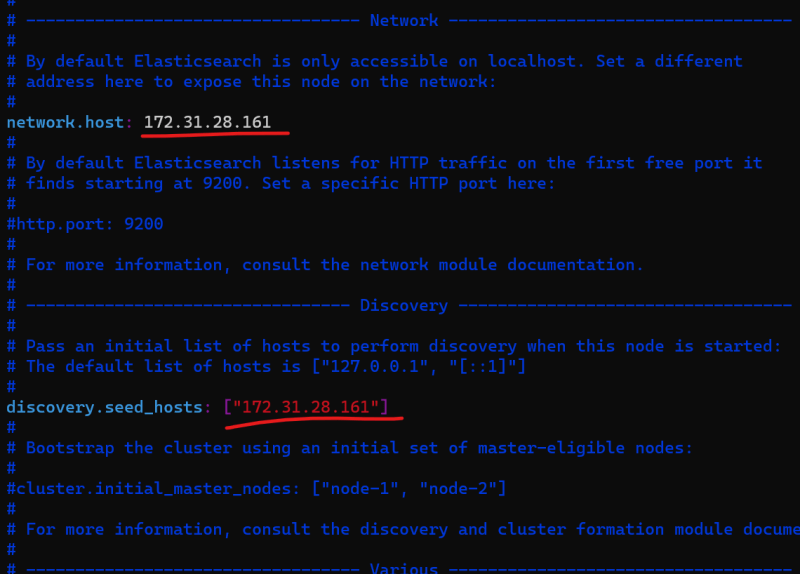
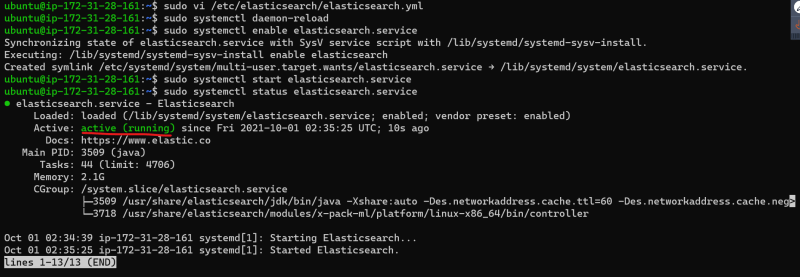
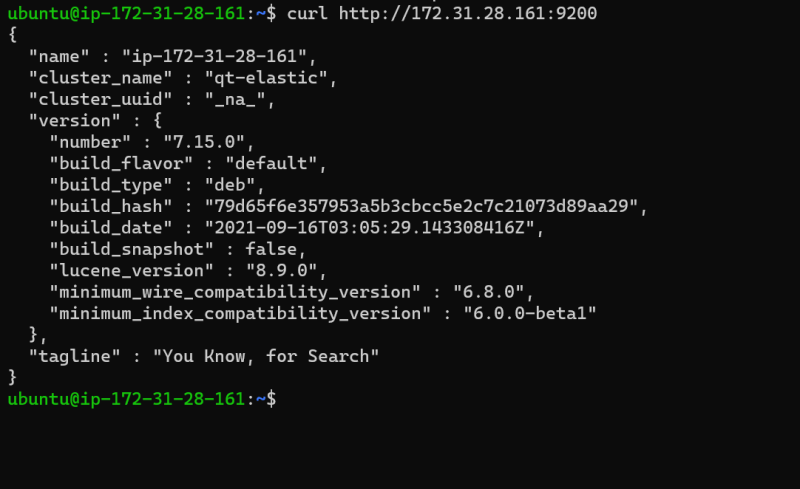
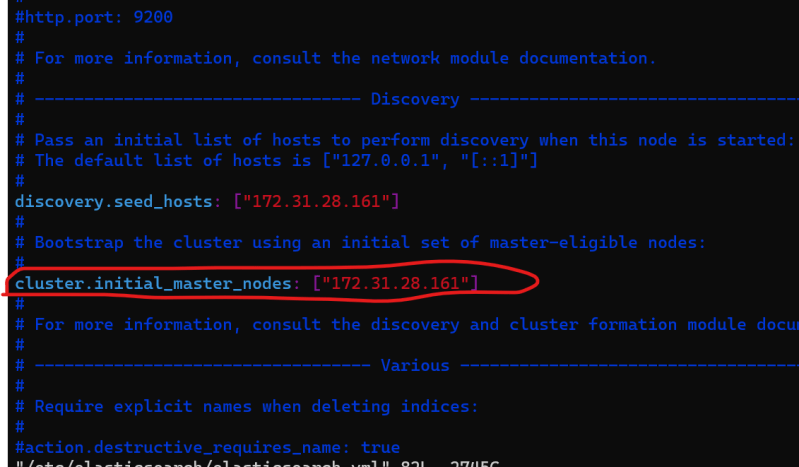
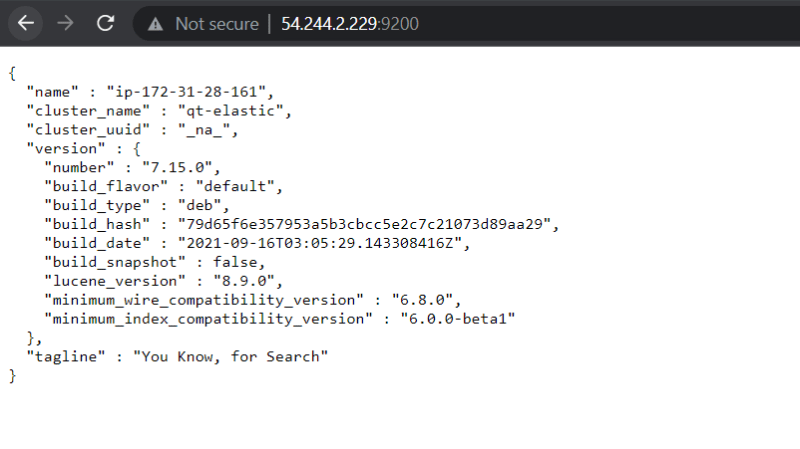
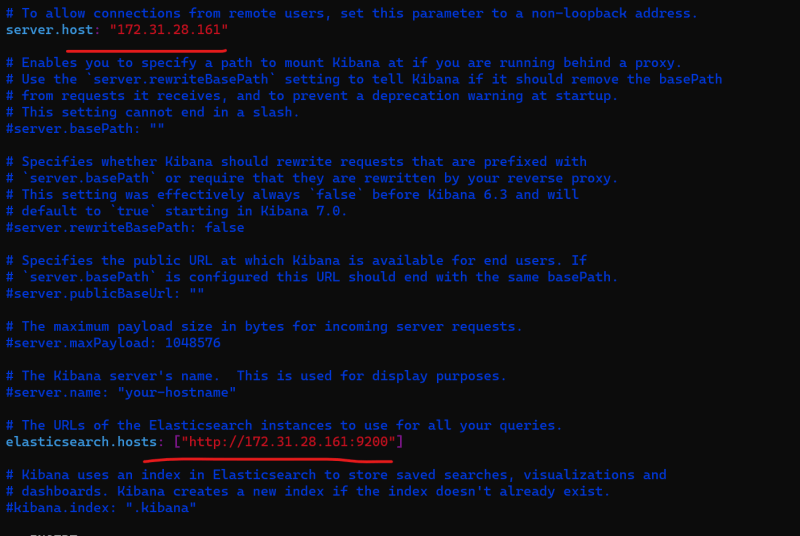
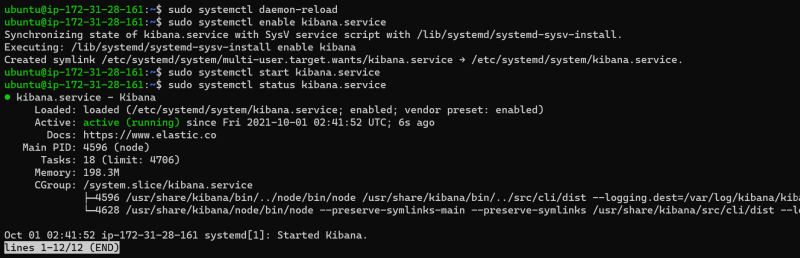
}

**Share this:**

OCTOBER 1, 2021

DevOps Classroom Series – 01/Oct/2021

**Lab Setup**

* Basic Architecture  
* Elastic Search and Kibana Setup
  + Create a vm with at least 4GB of RAM
  + Install elastic search by following instructions [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html#deb-repo)
  + To configure elastic search [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/settings.html)     
  + Now lets install kibana [Refer Here](https://www.elastic.co/guide/en/kibana/current/deb.html#deb-repo)
  + Now lets work with kibana configuration  
  + Now access kibana
* Lets create an apache server and also install file beats to export the logs of apache to logstash
  + Create a free vm
  + Install apache2
* sudo apt update
* sudo apt install apache2 -y
* Now create a configuration file to test whether the logs can be processed by logstash

input

{

stdin {}

}

filter

{

grok

{

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

}

}

output

{

stdout {}

}

* Now lets change the output in the configuration [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-outputs-elasticsearch.html)

input

{

stdin {}

}

filter

{

grok

{

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

}

}

output

{

elasticsearch

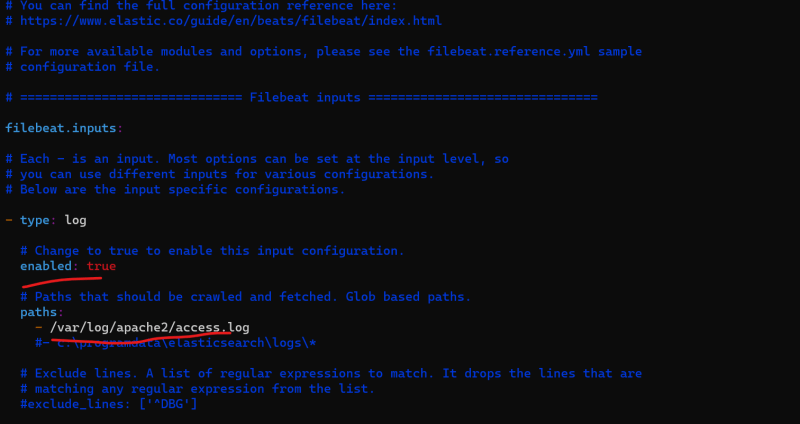
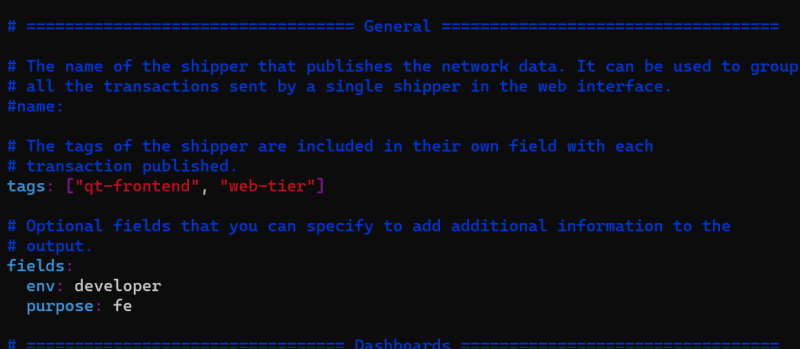
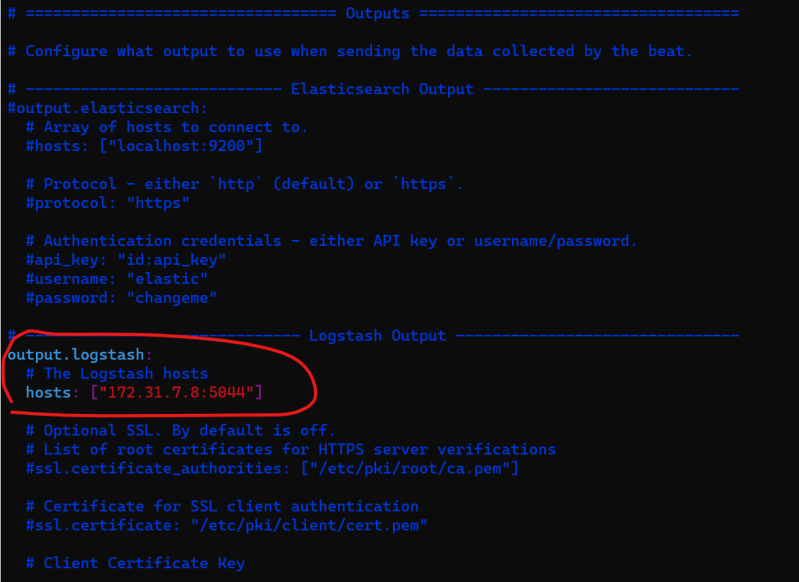
{

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

hosts => "172.31.28.161"

}

}

* Now lets install beats, which can reads logs from local server and send it to the logstash.
  + For this we will be installing file beats [Refer Here](https://www.elastic.co/guide/en/beats/filebeat/current/setup-repositories.html)
  + Configure filebeats to read apache access logs   
* Now we need to change the input of the conf to read from beats. Beats will forward the traffic to some port on logstash. [Refer Here](https://www.elastic.co/guide/en/logstash/current/plugins-inputs-beats.html)

input

{

beats

{

port => 5044

}

}

filter

{

grok

{

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

}

}

output

{

elasticsearch

{

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

hosts => "172.31.28.161"

}

}

* Next Steps:
  + We need to configure logstash to start automatically whenever the linux machine starts
  + We need to place the above configuration in a specific folder
  + Now we need to start beats, which exports the logs to logstash which add fields and stores in elastic search

**Share this:**

OCTOBER 3, 2021

DevOps Classroom Notes – 02/Oct/2021

**Logstash apache configurations**

* Navigate to /etc/logstash/conf.d
* Create a new configuration file apache.conf with the following content

input

{

beats

{

port => 5044

}

}

filter

{

grok

{

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

}

}

output

{

elasticsearch

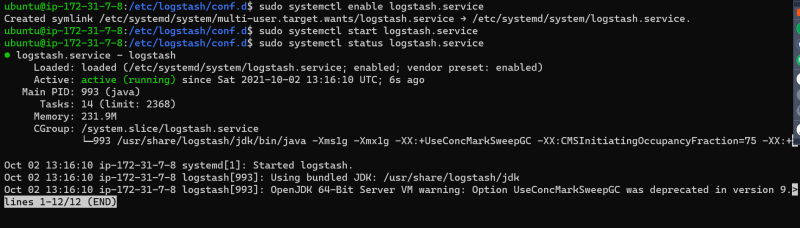
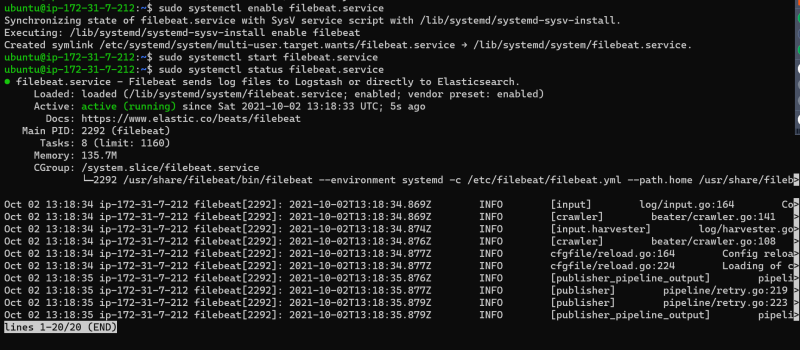
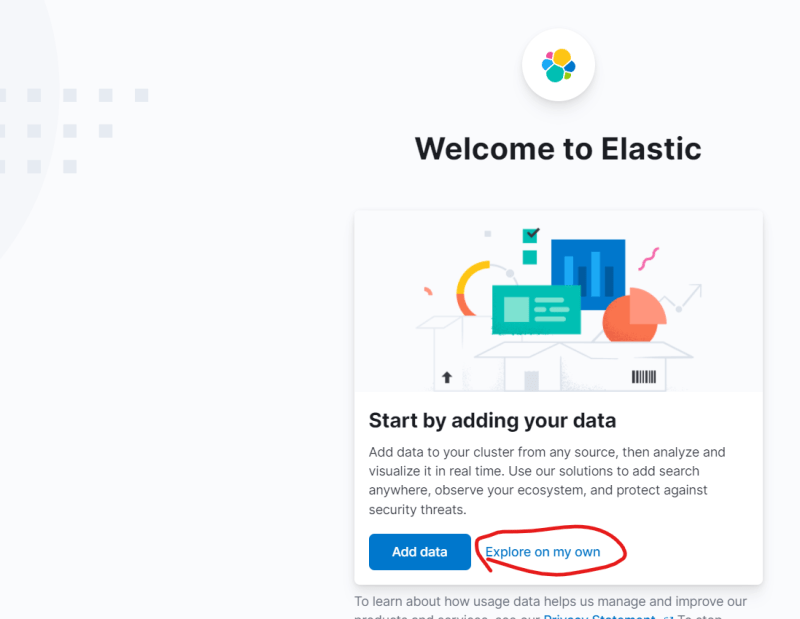
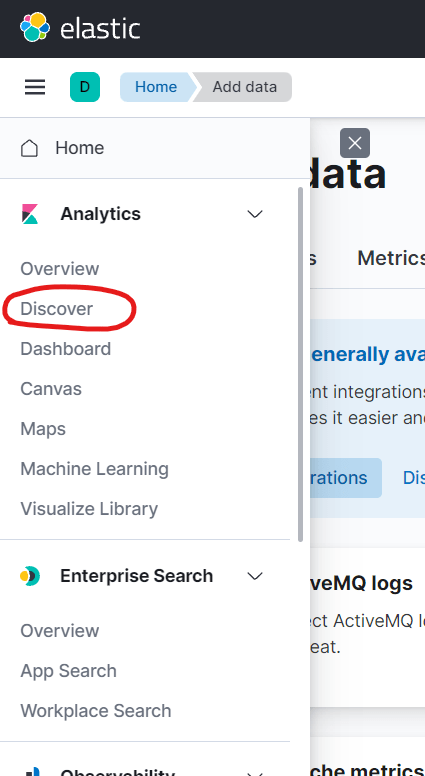
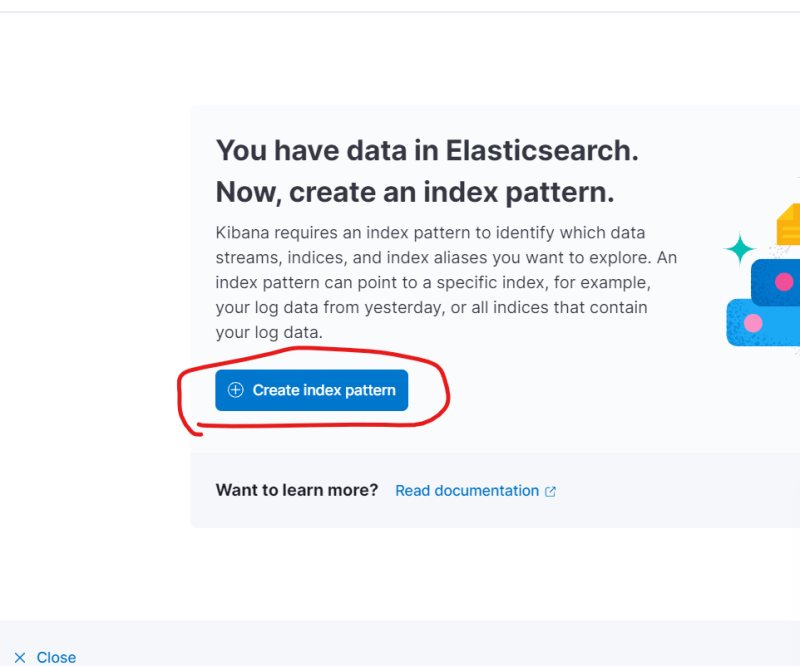
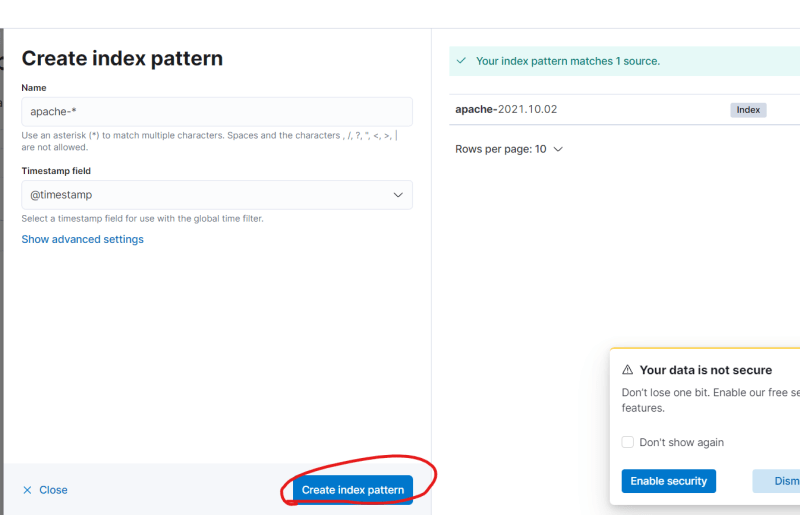
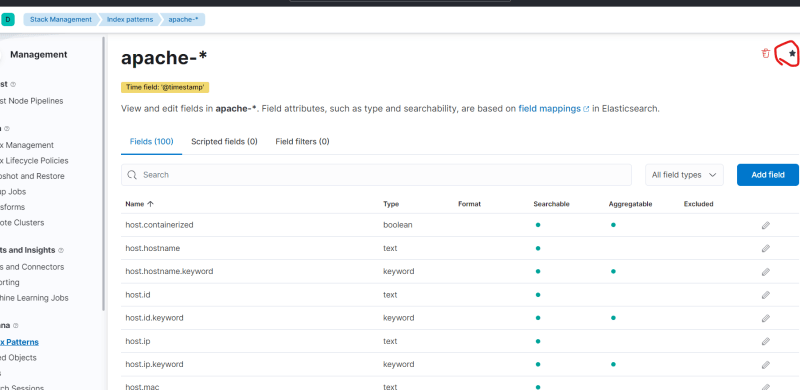
{

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

hosts => "172.31.28.161"

}

}

* enable and start logstash service 
* Now log in to system with apache installed and start beats service 
* Now lets go to the kibana ui to search for elastic search indexes     
* Lets create a sample script to create requests.

while true

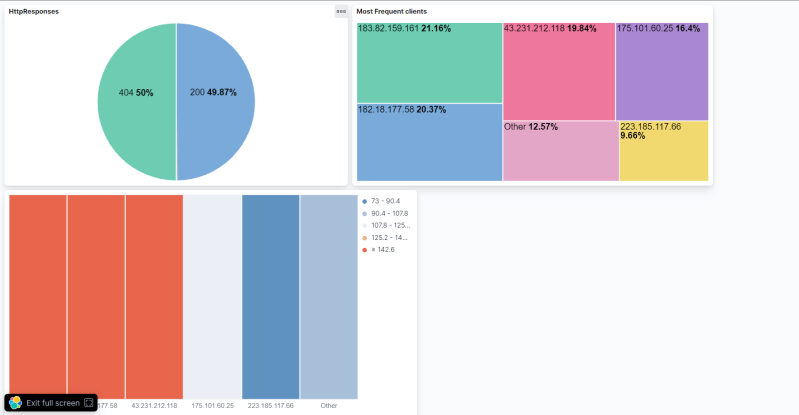
do

curl http://54.191.223.145

curl http://54.191.223.145/test.hml

sleep 10s

done

* Now follow the class room video for ui configurations done on kibana to create visualizations 

**Metric Beats**

* [Refer Here](https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-overview.html) for the official docs
* Lets install metric beats on the apache server & forward metrics directly to elastic search
* Installation [Refer Here](https://www.elastic.co/guide/en/beats/metricbeat/current/setup-repositories.html)
* Now lets change the configurations in /etc/metricbeat/metricbeat.yml
* [Refer Here](https://www.elastic.co/guide/en/beats/metricbeat/current/load-kibana-dashboards.html) to load the kibana dashboards
* metricbeat.yml used in the class

###################### Metricbeat Configuration Example #######################

# This file is an example configuration file highlighting only the most common

# options. The metricbeat.reference.yml file from the same directory contains all the

# supported options with more comments. You can use it as a reference.

#

# You can find the full configuration reference here:

# https://www.elastic.co/guide/en/beats/metricbeat/index.html

# =========================== Modules configuration ============================

metricbeat.config.modules:

# Glob pattern for configuration loading

path: ${path.config}/modules.d/\*.yml

# Set to true to enable config reloading

reload.enabled: false

# Period on which files under path should be checked for changes

#reload.period: 10s

#========================== Modules configuration =============================

metricbeat.modules:

#-------------------------------- System Module --------------------------------

- module: system

metricsets:

- cpu # CPU usage

#- load # CPU load averages

- memory # Memory usage

- network # Network IO

- process # Per process metrics

- process\_summary # Process summary

- uptime # System Uptime

- socket\_summary # Socket summary

- core # Per CPU core usage

- diskio # Disk IO

- filesystem # File system usage for each mountpoint

- fsstat # File system summary metrics

#- raid # Raid

#- socket # Sockets and connection info (linux only)

- service # systemd service information

enabled: true

period: 10s

processes: ['.\*']

# Configure the metric types that are included by these metricsets.

cpu.metrics: ["percentages","normalized\_percentages"] # The other available option is ticks.

core.metrics: ["percentages"] # The other available option is ticks.

#-------------------------------- Apache Module --------------------------------

- module: apache

metricsets: ["status"]

period: 10s

enabled: true

# Apache hosts

hosts: ["http://172.31.7.212"]

# ======================= Elasticsearch template setting =======================

setup.template.settings:

index.number\_of\_shards: 1

index.codec: best\_compression

#\_source.enabled: false

# ================================== General ===================================

# The name of the shipper that publishes the network data. It can be used to group

# all the transactions sent by a single shipper in the web interface.

#name:

# The tags of the shipper are included in their own field with each

# transaction published.

#tags: ["service-X", "web-tier"]

# Optional fields that you can specify to add additional information to the

# output.

#fields:

# env: staging

# ================================= Dashboards =================================

# These settings control loading the sample dashboards to the Kibana index. Loading

# the dashboards is disabled by default and can be enabled either by setting the

# options here or by using the `setup` command.

#setup.dashboards.enabled: false

# The URL from where to download the dashboards archive. By default this URL

# has a value which is computed based on the Beat name and version. For released

# versions, this URL points to the dashboard archive on the artifacts.elastic.co

# website.

#setup.dashboards.url:

# =================================== Kibana ===================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.

# This requires a Kibana endpoint configuration.

setup.kibana:

# Kibana Host

# Scheme and port can be left out and will be set to the default (http and 5601)

# In case you specify and additional path, the scheme is required: http://localhost:5601/path

# IPv6 addresses should always be defined as: https://[2001:db8::1]:5601

host: "172.31.28.161:5601"

# Kibana Space ID

# ID of the Kibana Space into which the dashboards should be loaded. By default,

# the Default Space will be used.

#space.id:

setup.dashboards.enabled: true

# =============================== Elastic Cloud ================================

# These settings simplify using Metricbeat with the Elastic Cloud (https://cloud.elastic.co/).

# The cloud.id setting overwrites the `output.elasticsearch.hosts` and

# `setup.kibana.host` options.

# You can find the `cloud.id` in the Elastic Cloud web UI.

#cloud.id:

# The cloud.auth setting overwrites the `output.elasticsearch.username` and

# `output.elasticsearch.password` settings. The format is `<user>:<pass>`.

#cloud.auth:

# ================================== Outputs ===================================

# Configure what output to use when sending the data collected by the beat.

# ---------------------------- Elasticsearch Output ----------------------------

output.elasticsearch:

# Array of hosts to connect to.

hosts: ["172.31.28.161:9200"]

# Protocol - either `http` (default) or `https`.

#protocol: "https"

# Authentication credentials - either API key or username/password.

#api\_key: "id:api\_key"

#username: "elastic"

#password: "changeme"

# ------------------------------ Logstash Output -------------------------------

#output.logstash:

# The Logstash hosts

#hosts: ["localhost:5044"]

# Optional SSL. By default is off.

# List of root certificates for HTTPS server verifications

#ssl.certificate\_authorities: ["/etc/pki/root/ca.pem"]

# Certificate for SSL client authentication

#ssl.certificate: "/etc/pki/client/cert.pem"

# Client Certificate Key

#ssl.key: "/etc/pki/client/cert.key"

# ================================= Processors =================================

# Configure processors to enhance or manipulate events generated by the beat.

processors:

- add\_host\_metadata: ~

- add\_cloud\_metadata: ~

- add\_docker\_metadata: ~

- add\_kubernetes\_metadata: ~

# ================================== Logging ===================================

# Sets log level. The default log level is info.

# Available log levels are: error, warning, info, debug

#logging.level: debug

# At debug level, you can selectively enable logging only for some components.

# To enable all selectors use ["\*"]. Examples of other selectors are "beat",

# "publisher", "service".

#logging.selectors: ["\*"]

# ============================= X-Pack Monitoring ==============================

# Metricbeat can export internal metrics to a central Elasticsearch monitoring

# cluster. This requires xpack monitoring to be enabled in Elasticsearch. The

# reporting is disabled by default.

# Set to true to enable the monitoring reporter.

#monitoring.enabled: false

# Sets the UUID of the Elasticsearch cluster under which monitoring data for this

# Metricbeat instance will appear in the Stack Monitoring UI. If output.elasticsearch

# is enabled, the UUID is derived from the Elasticsearch cluster referenced by output.elasticsearch.

#monitoring.cluster\_uuid:

# Uncomment to send the metrics to Elasticsearch. Most settings from the

# Elasticsearch output are accepted here as well.

# Note that the settings should point to your Elasticsearch \*monitoring\* cluster.

# Any setting that is not set is automatically inherited from the Elasticsearch

# output configuration, so if you have the Elasticsearch output configured such

# that it is pointing to your Elasticsearch monitoring cluster, you can simply

# uncomment the following line.

#monitoring.elasticsearch:

# ============================== Instrumentation ===============================

# Instrumentation support for the metricbeat.

#instrumentation:

# Set to true to enable instrumentation of metricbeat.

#enabled: false

# Environment in which metricbeat is running on (eg: staging, production, etc.)

#environment: ""

# APM Server hosts to report instrumentation results to.

#hosts:

# - http://localhost:8200

# API Key for the APM Server(s).

# If api\_key is set then secret\_token will be ignored.

#api\_key:

# Secret token for the APM Server(s).

#secret\_token:

# ================================= Migration ==================================

# This allows to enable 6.7 migration aliases

#migration.6\_to\_7.enabled: true

**Packet Beat**

* This can be used to monitor the incoming and outgoing network packets from your system [Refer Here](https://www.elastic.co/guide/en/beats/packetbeat/current/packetbeat-installation-configuration.html)

**Other Beats:**

* Heartbeat: [Refer Here](https://www.elastic.co/guide/en/beats/heartbeat/current/setup-repositories.html)
* Winlogbeat
* Journalbeat

**Share this:**

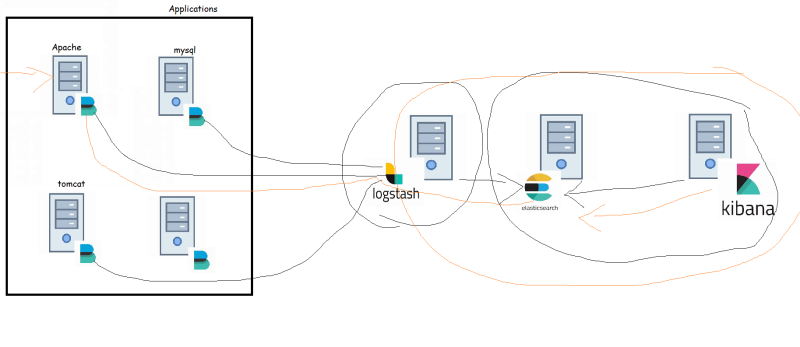
* [Click](https://directdevops.blog/2021/10/03/devops-classroom-notes-02-oct-2021/?share=twitter&nb=1)

OCTOBER 3, 2021

DevOps Classroom Series – 03/Oct/2021

**X-Pack**

* X-Pack is an addon to the elastic stack which provides security, alerting capabilities.
* X-Pack is not free software, you need to purchase x-pack. To start with we can do this with 30 day trail version
* To view features of the alerting & Security lets create an elastic cloud account [Refer Here](https://www.elastic.co/cloud/elasticsearch-service/signup)
* on the Elastic cloud we can create alert rules to perform actions on some patterns found in the logs.



**Share this:**