**Open command prompt:**

**Create a live log server:**

I am using java multithread program to create a file which will continuously write log entries into it within specific time interval:

java program and the file name is “logFile.log”

|  |
| --- |
|  |
| import java.io.File;  import java.io.PrintWriter;  import java.io.IOException;  import java.util.Random;  import java.text.DateFormat;  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.TimeZone;  public class GenerateRandomLog {  //time ISO 8601  private String nowAsISO;  public void date(){  //TimeZone tz = TimeZone.getTimeZone("UTC");  DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:sss'Z'");  //df.setTimeZone(tz);  nowAsISO = df.format(new Date());  //System.out.println("TIme is "+ nowAsISO);  }    //log-level  public enum Season { INFO, WARN, ERRO }    //random string  private static final String CHAR\_LIST =  "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";  private static final int RANDOM\_STRING\_LENGTH = 10;    public String generateRandomString(){    StringBuffer randStr = new StringBuffer();  for(int i=0; i<RANDOM\_STRING\_LENGTH; i++){  int number = getRandomNumber();  char ch = CHAR\_LIST.charAt(number);  randStr.append(ch);  }  return randStr.toString();  }    private int getRandomNumber() {  int randomInt = 0;  Random rand = new Random();  int n = rand.nextInt(62);  return n;  }  //main  public static void main(String[] args){  GenerateRandomLog t = new GenerateRandomLog();  try{  File file = new File("testFile.log");    if(!file.exists()) {  file.createNewFile();  }  PrintWriter pw = new PrintWriter(file);    //Write content here..  for(int i=0; i<100; i++){  //time ISO8601  t.date();  pw.print(t.nowAsISO+ "\t");    //random log-level  Random rand = new Random();  int n = rand.nextInt(3);  pw.print(Season.values()[n]+ "\t");    //random string  String text = t.generateRandomString();  pw.print("[" + text +"]\t");    //fixed string  pw.println("This is demo text \"Message filed\"" + " ");    }    pw.close();  System.out.println("File Created");  } catch(IOException e) {  e.printStackTrace();  }    }  } |
|  |

**Compile the java program and run it**

C:\Users\.....\java> javac ServerLog.java

C:\Users\.....\java> java ServerLog

\*Note: This path will be used by filebeat as input path

**Run Elasticsearch**

C:\Users\.....\ELK\_Stack\elasticsearch-5.4.0\bin> elasticsearch

To verify open localhost:9200 in your browser

**Run Kibana**

C:\Users\.....\ELK\_Stack\kibana-5.4.0-windows-x86\kibana-5.4.0-windows-x86\bin>kibana.bat

To verify open localhost:5601 in your browser

**Write logstash plugin configuration:**

|  |
| --- |
| filename: myconfig.conf |
| input {  beats {  port => 5044  }  }  filter {  #grok match for timestamp match, used for replacing with log timestamp  grok {  match => [ "message", "%{TIMESTAMP\_ISO8601:tstamp}" ]  }  #Replace the log timestamp with logstash @timestamp  date {  match => ["tstamp" , "ISO8601"]  target => "@timestamp"  remove\_field => "tstamp"  }  if [tags] != "\_grokparsefailure" {  grok {  match => [ "message", "%{LOGLEVEL:log-level}" ]  }  }  if [tags] != "\_grokparsefailure" {  grok {  match => [ "message", "\[%{DATA:data\_info}\]" ]  }  }  }  output {  elasticsearch {  hosts => ["localhost:9200"]  index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"  codec => "rubydebug"  }  } |
|  |

C:\Users\.....\ELK\_Stack\logstash-5.4.0\logstash-5.4.0\bin> logstash –f myconfig.conf –config.reload.automatic

**Open filebeat.yml file**

Uncomment filebeat prospectors

filebeat.prospectors:

Give the path to your live server file by commenting the old path and adding new path

- input\_type: log

paths:

- C:\Users\.....\java\logFile.log

Comment Elasticsearch output and uncomment Logstash output

output.logstash:

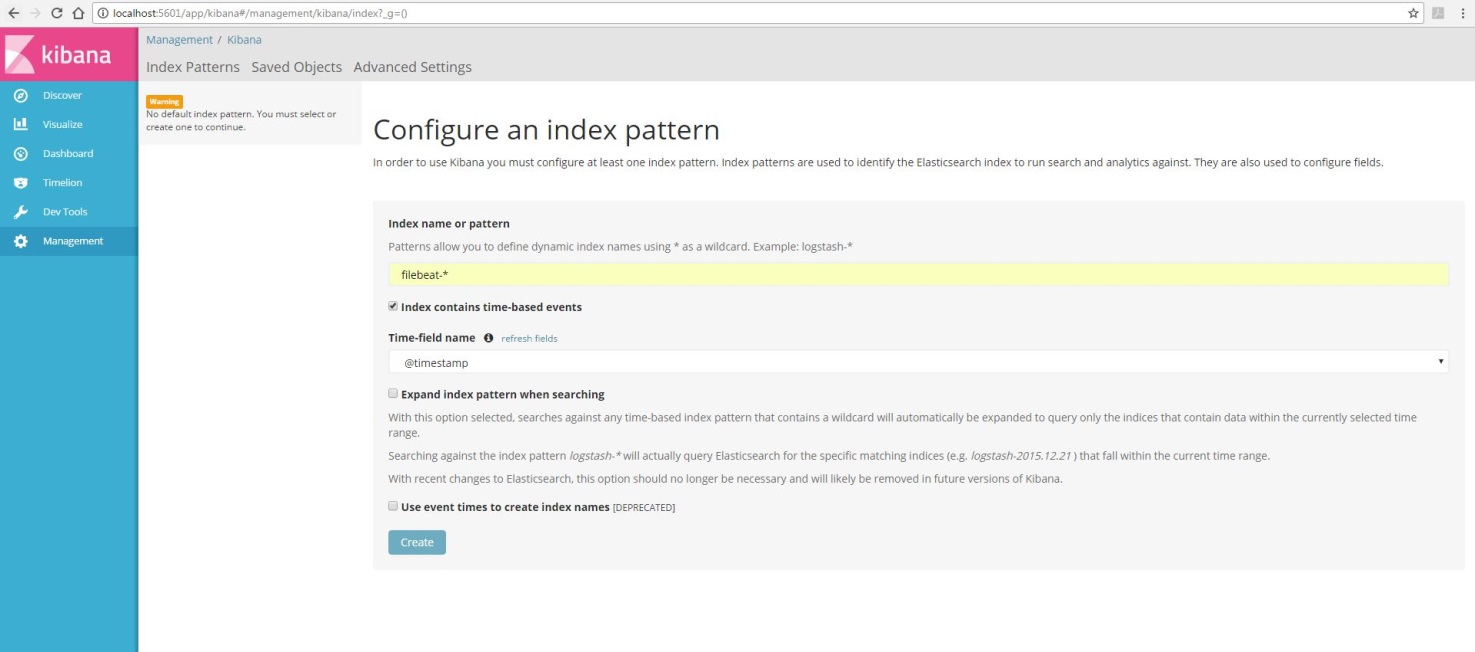
hosts: [“localhost:5044”]

**Run Filebeat**

C:\Users\.....\ELK\_Stack\filebeat-5.4.1-windows-x86\_64> filebeat –e

**Open kibana in your browser and do query and analysis**

Open kibana tab in browser



Set the Index name to “filebeat-\*”

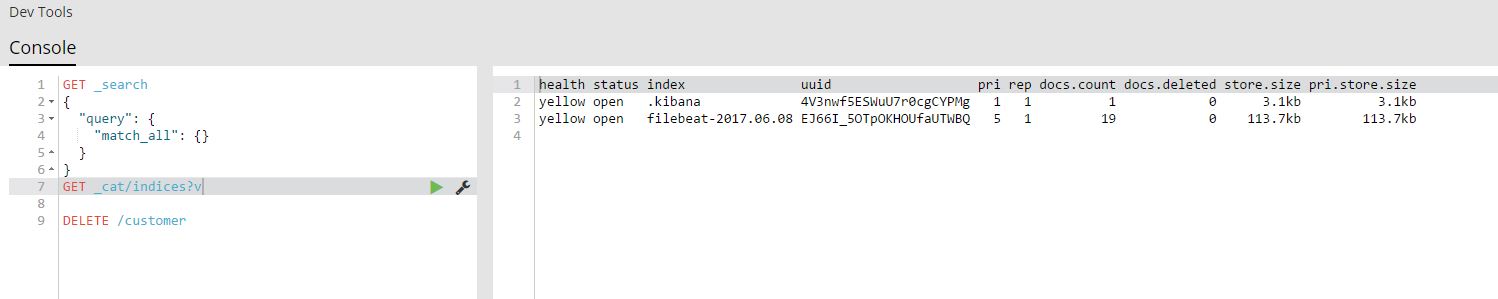
Add Time-field name ’@timestamp’ (It will be shown automatically or click on ‘refresh fields’) and

Click on Create, this will create the index.

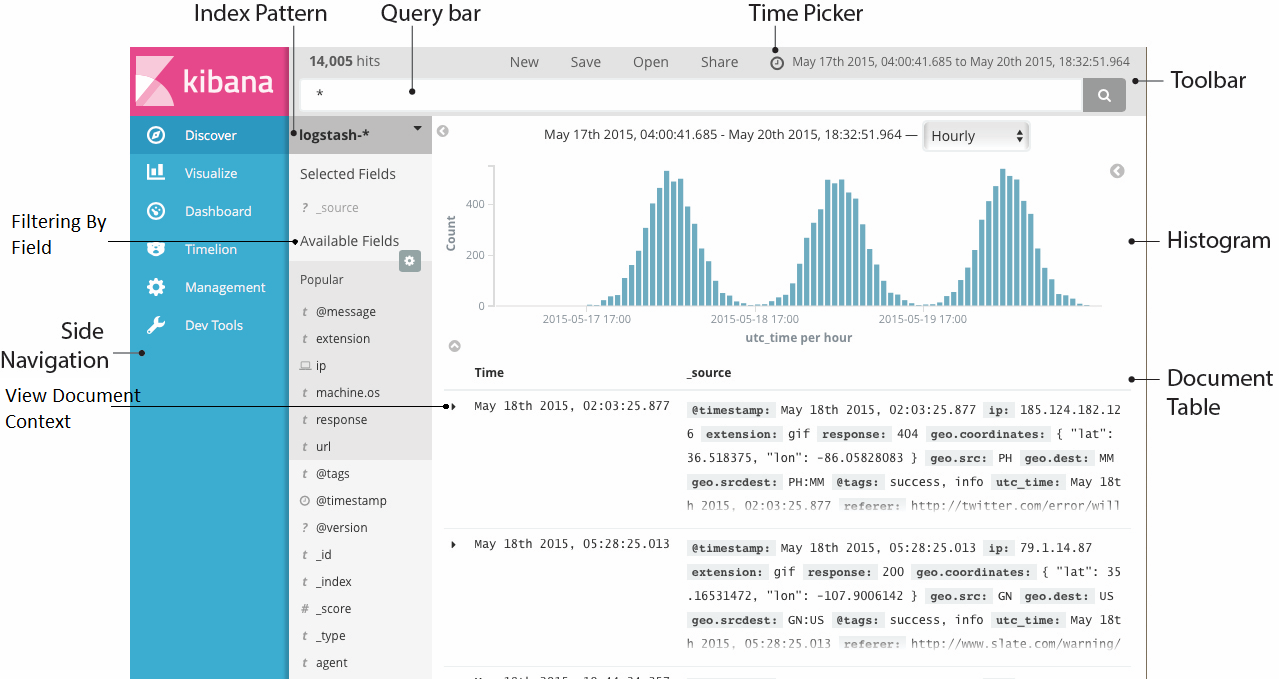
To see whether index is created or not

Go to Dev Tools

Run GET \_cat/indices?v



Go to Discover tab



Search and analyze the log as per your need.

Searching rule for Query bar

|  |  |  |
| --- | --- | --- |
| Search | Search Description | Example (  input\_type:”log” source:”C:\Users\dt216476\Desktop\java\logFile.log” message:”2017-06-13T11:00:016Z INFO [DCejfe2yJ6] ID: 10 This is demo text”  type:”log”  data\_info:”DCejfe2yJ6”  @timestamp:”June 13th 2017, 11:00:01.000”  log-level:”INFO”  @version:”1”  \_id:”AVyf7rV8yBvUkmfe1YtG”) |
| ‘field name’ | It will look for the keyword ‘field name’ | This is demo text |
| ‘fieldname’: ’value’ | It will look for value in a specific field | log-level: INFO |
| [‘value1’ To ‘value2’] | It will look for Value between ‘value1’ To ‘value2’ | @timestamp: ["2017-06-11T11:12:13" TO "2017-06-13T11:20:20"] #Note: format is yyyy-MM-ddTHH:mm:ss |
| AND OR NOT | Boolean search operation | log-level: WARN OR log-level: ERR |
| \* ? | Wildcard Characters | DCejfe2y?? |
| Message field | It is not case-sensitive while searching within message but case sensitive with specific field | log-level: info won’t work instead use log-level: INFO  also  dcejfe2yj6 will match with message field (not case sensitive) but not with data\_info field (case-sensitive) |