链滴

使用 k8s 搭建 elk

作者：Gakkiyomi2019

原文链接：https://ld246.com/article/1605508915097

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

ELK 到底是什么呢？ELK是三个开源项目的首字母缩写，这三个项目分别是：Elasticsearch、Logstash 和 Kibana。Elasticsearch 是一个搜索和分析引擎。Logstash 是服务器端数据处理管道，能够同时多个来源采集数据，转换数据，然后将数据发送到诸如 Elasticsearch 等“存储库”中。Kibana 则可让用户在 Elasticsearch 中使用图形和图表对数据进行可视化。

但是在实际的使用过程中我们还会引入一个新的组件，Filebeat它是 logstash的轻量级Golang语言本。

在这里 我们明确一下分工

- Filebeat： 从每个节点上收集日志
- Logstash： 对收集到的日志进行处理分析,然后输出到Elasticsearch
- Elasticsearch： 提供日志的存储和搜索功能
- Kibana： 提供对于Elasticsearch的可视化界面与可视化操作

ps：本文所使用的docker 镜像是似有仓库的镜像，自己可以使用官方镜像或者其他。

Filebeat

Filebeat是用于转发和集中日志数据的轻量级传送程序。使用Golang语言开发,作为服务器上的代理装，Filebeat监视您指定的日志文件或位置，收集日志事件，并将它们转发到Elasticsearch或Logstath进行索引。

在k8s集群中,由于不知道应用服务会被调度到哪个节点上，或者说服务挂掉后被k8s的scheduler又分到其他节点上，所以我们会使用 DaemonSet 来对每个节点上都起一个filebeat服务

首先我们先准备两个yaml文件cm.yaml,ds.yaml

cm.yaml

这是使用configmap来维护filebeat的配置文件,在k8s中如果需要使用配置文件,我们都回采取configmap的方式来挂载配置文件。关于配置文件的内容可根据自己的业务情况进行调整，具体的字段配置详请参考官方文档配置参考

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config   #声明此config的名字
  namespace: sky
  labels:
    k8s-app: filebeat
    kubernetes.io/cluster-service: "true"
    app: filebeat-config
data:
  filebeat.yml: |
    processors:
      ####
      - add_host_metadata:
      ####
      - add_cloud_metadata:
    ####
    setup.template.settings:
      index.number_of_shards: 3
    setup.kibana:
      host: "kb-single-svc:5601"
    ####
    filebeat.modules:
    - module: system
      syslog:
        enabled: true
      auth:
        enabled: true
    filebeat.inputs:
    - type: log
      paths:
        - /var/log/workflow/*.log
      multiline.pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2}'
      multiline.negate: true
      multiline.match: after
      fields:
        log_source: workflow
    - type: log
      paths:
        - /var/log/eventti/*.log
      multiline.pattern: '^[0-9]{4}\/[0-9]{2}\/[0-9]{2}'
      multiline.negate: true
      multiline.match: after
      fields:
        log_source: eventti
    - type: log
      paths:
        - /var/log/gway/*.log
      multiline.pattern: '^[0-9]{4}\/[0-9]{2}\/[0-9]{2}'
      multiline.negate: true
```

```
    multiline.match: after
    fields:
      log_source: gway
  output.logstash:
    hosts: ['logstash:5045']
  logging.level: info
```

ds.yaml

然后我们使用DaemonSet来启动 filebeat的pod

DaemonSet 确保全部（或者一些）Node 上运行一个 Pod 的副本。当有 Node 加入集群时，也会他们新增一

个 Pod 。当有 Node 从集群移除时，这些 Pod 也会被回收。删除 DaemonSet 将会删除它创建的有 Pod

挂载好需要的日志文件和配置文件

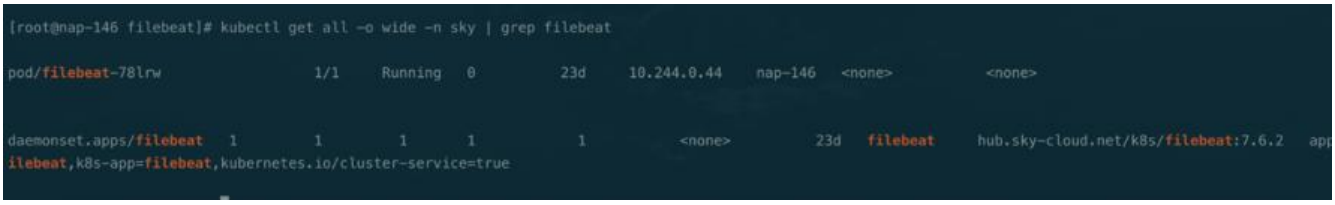
```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: filebeat
  namespace: sky
  labels:
    k8s-app: filebeat
    kubernetes.io/cluster-service: "true"
spec:
  template:
    metadata:
      name: filebeat
      labels:
        app: filebeat
        k8s-app: filebeat
        kubernetes.io/cluster-service: "true"
    spec:
      containers:
      - image: hub.sky-cloud.net/k8s/filebeat:7.6.2
        name: filebeat
        args: [
          "-c", "/home/filebeat-config/filebeat.yml",
          "-e",
        ]
        securityContext:
          runAsUser: 0
        volumeMounts:
        - name: workflow
          mountPath: /var/log/workflow
        - name: eventti
          mountPath: /var/log/eventti
        - name: gway
          mountPath: /var/log/gway
        - name: log
          mountPath: /var/log/kong-mapping
```

```
    - name: "filebeat-volume"
      mountPath: "/home/filebeat-config"
  volumes:
  - name: workflow
    hostPath:
      path: /srv/sky/workflow/log
  - name: eventti
    hostPath:
      path: /srv/sky/eventti/log
  - name: gway
    hostPath:
      path: /srv/sky/gway/log
  - name: log
    hostPath:
      path: /srv/sky/kong-mapping/log
  - name: filebeat-volume
    configMap:
      name: filebeat-config
```

启动

kubectl create -f .

查看启动情况



Logstash

logstash是一个数据分析软件，主要目的是分析log日志。日志经过采集后，就会输入到logstash，由ogstash进行加工过滤分析，再输入到ES中。这样，ES索引中的文档就会获取到所需的字段

logstash的配置分为

1. input
2. filter
3. output

Input

input模块主要用户收集数据。

1. 文件

```
input{
    file{
      #path属性接受的参数是一个数组，其含义是标明需要读取的文件位置
      path => ['pathA', 'pathB']
```

```
        #表示多久去path路径下查看是够有新的文件产生。默认是15秒检查一次。
        discover_interval => 15
        #排除那些文件，也就是不去读取那些文件
        exclude => ['fileName1','fileName2']
        #被监听的文件多久没更新后断开连接不在监听，默认是一个小时。
        close_older => 3600
        #在每次检查文件列 表的时候， 如果一个文件的最后 修改时间 超过这个值， 就忽略这个文
。 默认一天。
        ignore_older => 86400
        #logstash 每隔多 久检查一次被监听文件状态（ 是否有更新） ， 默认是 1 秒。
        stat_interval => 1
        #sincedb记录数据上一次的读取位置的一个index
        sincedb_path => '$HOME/. sincedb'
        #logstash 从什么 位置开始读取文件数据， 默认是结束位置 也可以设置为：beginning 从
开始
        start_position => 'beginning'
      }
    }
```

如果需要每次都从头开始读取文件的话，设置start_position => beginning是没有用的，你可以选择sncedb_path 定义为 /dev/null

2. 数据库

```
input{
   jdbc{
        #jdbc sql server 驱动,各个数据库都有对应的驱动，需自己下载
        jdbc_driver_library => "/etc/logstash/driver.d/sqljdbc_2.0/enu/sqljdbc4.jar"
        #jdbc class 不同数据库有不同的 class 配置
      jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
        #配置数据库连接 ip 和端口，以及数据库
        jdbc_connection_string => "jdbc:sqlserver://200.200.0.18:1433;databaseName=test_d
"
        #配置数据库用户名
        jdbc_user =>
        #配置数据库密码
        jdbc_password =>
        # 定时器 多久执行一次SQL，默认是一分钟
        # schedule => 分 时 天 月 年
        # schedule => * 22  *  *  * 表示每天22点执行一次
        schedule => "* * * * *"
        #是否清除 last_run_metadata_path 的记录,如果为真那么每次都相当于从头开始查询所有的
据库记录
        clean_run => false
        #是否需要记录某个column 的值,如果 record_last_run 为真,可以自定义我们需要表的字段
称，
        #此时该参数就要为 true. 否则默认 track 的是 timestamp 的值.
        use_column_value => true
        #如果 use_column_value 为真,需配置此参数. 这个参数就是数据库给出的一个字段名称。当
该字段必须是递增的，可以是 数据库的数据时间这类的
        tracking_column => create_time
        #是否记录上次执行结果, 如果为真,将会把上次执行到的 tracking_column 字段的值记录下来
保存到 last_run_metadata_path 指定的文件中
        record_last_run => true
        #只需要在 SQL 语句中 WHERE MY_ID > :last_sql_value 即可. 其中 :last_sql_value 取得就
```

```
该文件中的值
        last_run_metadata_path => "/etc/logstash/run_metadata.d/my_info"
        #是否将字段名称转小写。
        #这里有个小的提示，如果你这前就处理过一次数据，并且在Kibana中有对应的搜索需求的
，还是改为true，
        #因为默认是true，并且Kibana是大小写区分的。准确的说应该是ES大小写区分
        lowercase_column_names => false
        #你的SQL的位置，当然，你的SQL也可以直接写在这里。
        #statement => SELECT * FROM tabeName t WHERE  t.creat_time > :last_sql_value
        statement_filepath => "/etc/logstash/statement_file.d/my_info.sql"
        #ES数据类型
        type => "my_info"
        }
    }
}
```

外载的SQL文件就是一个文本文件就可以了，还有需要注意的是，一个jdbc{}插件就只能处理一个SQ语句，如果你有多个SQL需要处理的话，只能在重新建立一个jdbc{}插件。

3. 通过filebeat收集

```
input {
    beats {
        #接受数据端口
        port => 5044
        #数据类型
        type => "logs"
        }
    }
```

4. tcp,通常一些日志通过tcp连接暴露

```
input {
     tcp {
         port => "5044"
         codec => "json"
         add_field => ["log_channel", "kong"]
        }
    }
```

Filter

filter是过滤器，会对收集到的数据进行处理。elastic官方还提供了许多插件，可以协助数据处理

filter是logstash最复杂的一个模块，这里只做简述

1. grok 可以匹配一切数据
2. Geoip 获取 IP 对应的地理位置
3. json 对于 json 格式的 log，可以通过 codec 的 json 编码进行解析
4. split logstash 同样支持将一行数据变成多个事件,它提供了 split 插件，用来把一行数据拆分成多事件
5. mutate logstash 支持在 filter 中对事件中的数据进行修改

6. date 用于处理时间

以下是一个分析eventti日志的filter

首先创建了一个模式文件，用于存取正则模式

```
# /usr/share/logstash/eventti/patterns
DATE1   [0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}\.[0-9]+
LEVEL   (INFO)|(DEBUG)|(ERROR)|(WARN)|(FATAL)
JAVA_SOURCE \[.*\]
JAVALOGMESSAGE (.*)
DATE2   [0-9]{4}\/[0-9]{2}\/[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}
LEVEL2   (\[I\])|(\[D\])|(\[E\])|(\[W\])|(\[F\])
ID      [a-zA-Z0-9-]+
MESSAGE (.*?)
TYPE    [a-zA-Z0-9_]+
TIME1   [0-9]+
USERNAME  (.*)
```

然后在logstash配置中使用filter进行过滤分析

```
filter {
   grok {
      # 正则模式位置
      patterns_dir => ["/usr/share/logstash/eventti/patterns"]
      # 将匹配到的模式转化为字段
      match => {
         "message" => "%{DATE2:logdate} %{LEVEL2:loglevel} id:%{ID:logid} message:%{MESS
GE:logmessage}\stype:%{TYPE:logtype} timestamp:%{TIME1:logtime} user:%{USERNAME:user
ame}"
      }
   }
   date {
      # 匹配时间转化为时间戳
      match => [ "logtime", "UNIX_MS" ]
      target => "@timestamp"
   }
   mutate {
      # 转化格式
      convert =>{"logtime" => "integer"}
   }
}
```

Output

output用于将处理好的数据进行输出

1. 文件

```
output{
   file{

        path => "/home/app/logbak/%{+YYYY.MM.dd}-file.txt"
```

```
        codec => line {format => "%{[collectValue]}"} # 设置根据原始数据格式保存，不会带Json
式
    }
}
```

2. ES

```
output{
    elasticsearch{
        # ES地址
        hosts=>["172.132.12.3:9200"]
        # es要执行的动作 index, delete, create, update
        action=>"index"
        index=>"indextemplate-logstash"
        #document_type=>"%{@type}"
        # 为索引提供document id ， 对重写elasticsearch中相同id词目很有用
        document_id=>"ignore"
        # 有效的filepath 设置自己的template文件路径
        template=>"/opt/logstash-conf/es-template.json"
        template_name=>"es-template.json"
        template_overwrite=>true
    }
}
```

pipeline

Logstash 通常需要处理多个并行事件流，如果总是通过if进行判断则显得臃肿。

logstash在6.0.0之后引入了pipeline。用户可以在配置文件pipelines.yml中添加新的 pipeline 配置指定其配置文件就可以了

cm.yaml

配置文件

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: logstash-config
  namespace: sky
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
data:
    logstash.yml: |
        http.host: "0.0.0.0"
        xpack.monitoring.elasticsearch.hosts: [ "http://es-single-nodeport:9200" ]
    pipelines.yml: |
        - pipeline.id: kong
          path.config: "/usr/share/logstash/kong/kong.conf"
        - pipeline.id: eventti
          path.config: "/usr/share/logstash/eventti/eventti.conf"
    patterns: |
        DATE1    [0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}\.[0-9]+
```

```
    LEVEL   (INFO)|(DEBUG)|(ERROR)|(WARN)|(FATAL)
    JAVA_SOURCE \[.*\]
    JAVALOGMESSAGE (.*)
    DATE2   [0-9]{4}\/[0-9]{2}\/[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}
    LEVEL2   (\[I\])|(\[D\])|(\[E\])|(\[W\])|(\[F\])
    ID      [a-zA-Z0-9-]+
    MESSAGE (.*?)
    TYPE    [a-zA-Z0-9_]+
    TIME1   [0-9]+
    USERNAME  (.*)
    LEVEL3  (\[[I|D|E|W|F]\])

  eventti.conf: |
    input {
      beats {
        port => "5045"
      }
    }

    filter {
      if [fields][log_source] == "eventti" {
        grok {
          patterns_dir => ["/usr/share/logstash/eventti/patterns"]
          match => {
            "message" => "%{DATE2:logdate} %{LEVEL2:loglevel} id:%{ID:logid} message:%{ME
SAGE:logmessage}\stype:%{TYPE:logtype} timestamp:%{TIME1:logtime} user:%{USERNAME:us
rname}"
          }
        }
        date {
          match => [ "logtime", "UNIX_MS" ]
          target => "@timestamp"
        }
        mutate {
          convert =>{"logtime" => "integer"}
        }
      }
      if [fields][log_source] == "workflow" {
        grok {
          patterns_dir => ["/usr/share/logstash/eventti/patterns"]
          match => {
            "message" => "%{DATE1:time1} %{JAVA_SOURCE:source1} %{LEVEL:level1}  %{JAVA
OGMESSAGE:doc}"
          }
        }
      }
      if [fields][log_source] == "gway" {
        grok {
          patterns_dir => ["/usr/share/logstash/eventti/patterns"]
          match => {
            "message" => "%{DATE2:rpcdate} %{LEVEL2:rpcloglevel}?\s%{USERNAME:consumer}
Remote Procedure Call %{USERNAME:provider}"
          }
        }
```

原文链接：使用 k8s 搭建 elk

```
    }
  }

  output {
    if [fields][log_source] == "eventti" {
      elasticsearch {
        hosts => ["http://es-single-nodeport:9200"]
        index => "eventti-%{+YYYY.MM.dd}"
      }
    }
    if [fields][log_source] == "workflow" {
      elasticsearch {
        hosts => ["http://es-single-nodeport:9200"]
        index => "workflow-%{+YYYY.MM.dd}"
      }
    }
    if [fields][log_source] == "gway" {
      elasticsearch {
        hosts => ["http://es-single-nodeport:9200"]
        index => "gway-%{+YYYY.MM.dd}"
      }
    }
  }
kong.conf: |
  input {
    tcp {
      port => "5044"
      codec => "json"
      add_field => ["log_channel", "kong"]
    }
  }

  filter {

    if [log_channel] == "kong" {
        mutate {
          rename =>{"[host]" => "[host][name]"}
          convert =>{"[request][size]" => "integer"}
          convert =>{"[response][size]" => "integer"}
        }

        ruby {
          code => "
          restime = event.get('[route][updated_at]')
          reqtime = event.get('[service][created_at]')
          duration = restime-reqtime
          event.set('duration', duration)
          "
        }

        if [host][name] == "kong.kong" {
        mutate {
            add_field => {
            "[@metadata][index_prefix]" => "kong-all-dev"
```

```
                }
            }
            }

        }
      }

      output {
        elasticsearch {
          hosts => ["http://es-single-nodeport:9200"]
          index => "baas-kong-%{+YYYY.MM.dd}"
        }
      }
```

logstash.yaml

使用deployment控制器启动pod，并且挂载存储和配置文件

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: logstash
  namespace: sky
spec:
  replicas: 1
  selector:
    matchLabels:
      app: logstash
  template:
    metadata:
      name: logstash
      labels:
        app: logstash
    spec:
      containers:
      - image: hub.sky-cloud.net/k8s/logstash:7.6.2
        name: kb
        ports:
        - name: kong
          containerPort: 5044
        - name: filebeat
          containerPort: 5045
        volumeMounts:
        - name: tz
          mountPath: /etc/localtime
        - name: config
          mountPath: /usr/share/logstash/config
        - name: kong
          mountPath: /usr/share/logstash/kong
        - name: eventti
          mountPath: /usr/share/logstash/eventti
      volumes:
      - name: tz
        hostPath:
```

```
        path: /etc/localtime
      - name: config
        configMap:
          name: logstash-config
          items:
          - key: logstash.yml
            path: logstash.yml
          - key: pipelines.yml
            path: pipelines.yml
      - name: kong
        configMap:
          name: logstash-config
          items:
          - key: kong.conf
            path: kong.conf
      - name: eventti
        configMap:
          name: logstash-config
          items:
          - key: eventti.conf
            path: eventti.conf
          - key: patterns
            path: patterns
```

svc.yaml

service为我们对外暴露服务

```
apiVersion: v1
kind: Service
metadata:
  name: logstash
  namespace: sky
spec:
  ports:
  - name: kong
    port: 5044
    targetPort: 5044
  - name: filebeat
    port: 5045
    targetPort: 5045
  selector:
    app: logstash
```

Elasticsearch

ElasticSearch 分布式搜索和分析引擎。它是基于lucene搜索引擎开发的，拥有全文搜索、结构化搜、分析等功能。

ES 通过简单的restful API来隐藏lucene的复杂性，来使全文搜索变得简单

核心概念

es是面向文档的，一切都是json

1. 关系型数据库与es的对比

es将数据存储在索引中，可以使用关系型数据库的结构来类比ES

RDBMS	ES
数据库(db)	索引(index)
表(tables)	类型(types)
行(rows)	文档(documents)
列(columns)	字段(fields)

2. 索引

索引时一组文档的集合，当索引一篇文档时，可以通过 索引 -> 类型 -> 文档ID 找到

3. 分片

es把每个索引划分成多个分片，每个分片可以在集群中的不同服务器间迁移。ES自动管理和组织分片，并在必要的时候对分片数据进行再平衡分配。

当数据被写入分片时，它会定期发布到磁盘上的不可变的 Lucene 分段中用于查询。随着分段数量的长，这些分段会定期合并为更大的分段。 此过程称为合并。 由于所有分段都是不可变的，这意味着使用的磁盘空间通常会在索引期间波动，因为需要在删除替换分段之前创建新的合并分段。 合并可能常耗费资源，特别是在磁盘I / O方面。

实际一个分片就是一个lucene索引。es的索引事实是由多个lucene索引组成

4. 主分片与副本分片

索引会自动存储到各个分片上，ES 默认为一个索引创建 5 个主分片, 并分别为其创建一个副本分片。分片与副本都能处理查询请求，它们的唯一区别在于只有主分片才能处理索引请求。

副本是为提高搜索性能，用户也可在任何时候添加或删除副本。额外的副本能给带来更大的容量, 更的吞吐能力及更强的故障恢复能力。

主分片和副本分片都不会在同一个节点内

5. 文档

es索引和搜索数据的最小单元是文档

文档就是一条条数据，一篇文档包含字段和对应的值{k:v}

ES搜索原理

与传统的数据库不同，在es中，每个字段里面的每个单词都是可以被搜索的。

这种特性是由底层lucene支持的。lucene使用倒排索作为底层，这种结构适用于快速的全文搜索。一索引由文档中所有不重复的列表构成，对于每一个词，都有一个包含它的文档列表。

为了支持这个特性，es中会维护一个叫做“invertedindex”（也叫逆向索引）的表，表内包含了所文档中出现的所有单词，同时记录了这个单词在哪个文档中出现过。

比如以下有三个文档

1. doc1: aaa, bbb, ccc, ddd

2. doc2: bbb, ccc

3. doc3: aaa, bbb, ddd

那么es会维持以下一个数据结构

Term	DOC1	DOC2	DO 3
aaa			
bbb			
ccc			
ddd			

这样当我们随意搜索任意一个单词，es只要遍历一遍这个表，就可以指导有那些文档被匹配到了。

当我们搜索多个词时，es可以完全过滤掉无关的所有数据，提高效率。

关于查询语句的使用和各种语言的sdk调用这里不提及。有需求可参考官方文档：快速开始

svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: es-single-nodeport
  namespace: sky
spec:
  externalIPs:
  - 192.168.1.146
  ports:
  - name: http
    port: 9200
    targetPort: 9200
  - name: tcp
    port: 9300
    targetPort: 9300
  selector:
    app: es-single
```

elasticsearch.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: es-single
  namespace: sky
spec:
  replicas: 1
  selector:
    matchLabels:
      app: es-single
```

```
  template:
    metadata:
      name: es-single
      labels:
        app: es-single
    spec:
      tolerations:
      - key: "node.kubernetes.io/unreachable"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 60
      - key: "node.kubernetes.io/not-ready"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 60
      terminationGracePeriodSeconds: 0              # 异常立即删除
      containers:
      - image: hub.sky-cloud.net/k8s/elasticsearch:6.4.0
        name: es
        securityContext:
          privileged: true
        env:
        - name: network.host
          value: "_site_"
        - name: node.name
          value: "${HOSTNAME}"
        - name: discovery.type
          value: single-node
        - name: discovery.zen.ping.unicast.hosts
          value: "${ES_SINGLE_NODEPORT_SERVICE_HOST}"
        - name: cluster.name
          value: "sky-cloud"
        - name: ES_JAVA_OPTS
          value: "-server -Xms1024m -Xmx1024m"
        volumeMounts:
        - name: es-single-data
          mountPath: /usr/share/elasticsearch/data
        - name: time
          mountPath: /etc/localtime
      volumes:
      - name: es-single-data
        hostPath:
          path: /srv/sky/elk/elasticsearch/data
      - name: time
        hostPath:
          path: /etc/localtime
```

Kibana

Kibana旨在使用Elasticsearch作为数据源。将Elasticsearch视为存储和处理数据的引擎，而Kibana提供可视化服务。

cm.yaml

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: kibana-config
  namespace: sky
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
data:
    kibana.yml: |
        ---
        # Default Kibana configuration from kibana-docker.

        server.name: kibana
        server.host: 0.0.0.0
        server.basePath: "/api/sky-kibana"
        server.rewriteBasePath: true
        elasticsearch.url: http://es-single-nodeport:9200
        xpack.monitoring.ui.container.elasticsearch.enabled: true
        #i18n.locale: "zh-CN"
```

kibana.yaml

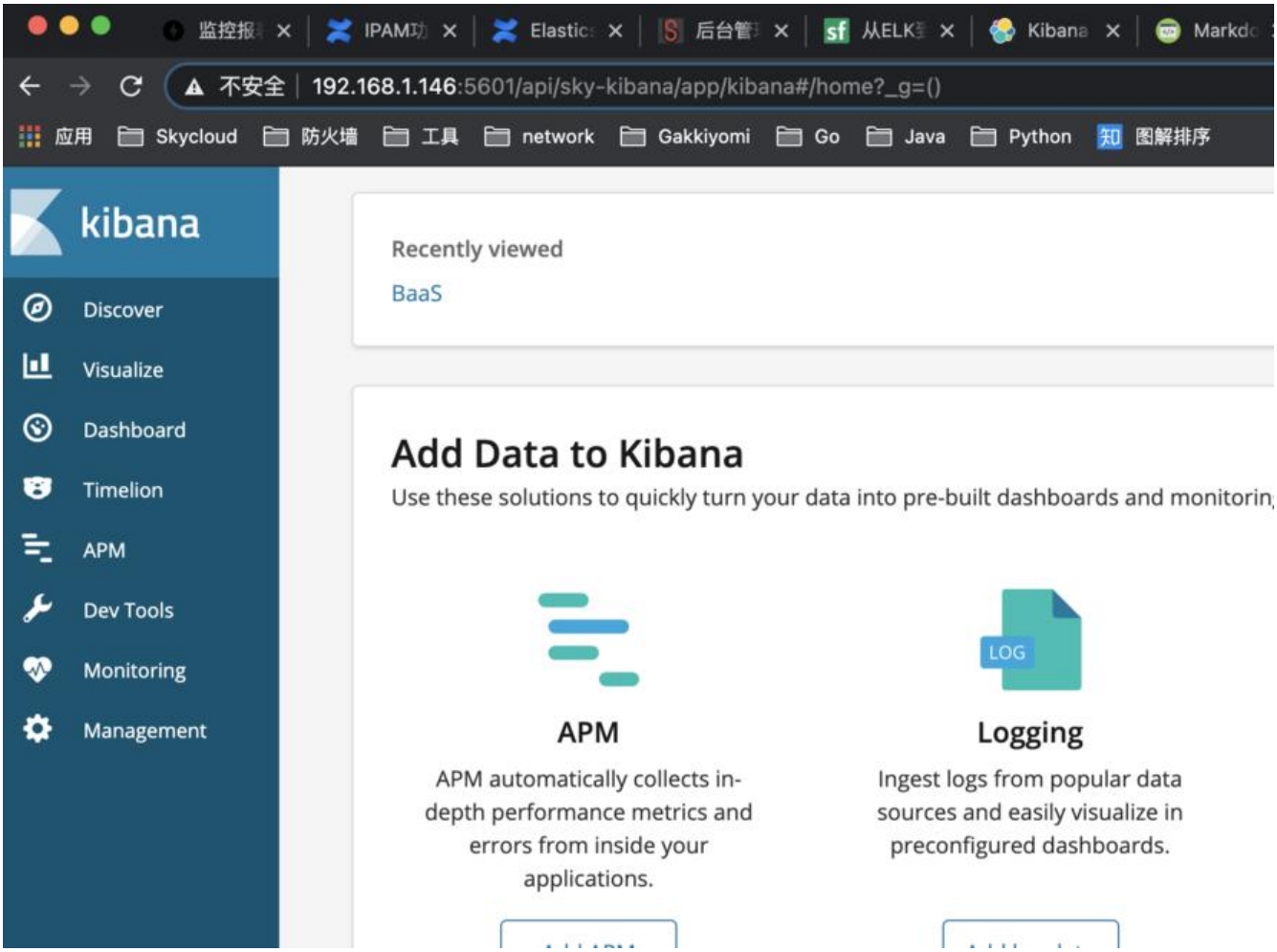
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kb-single
  namespace: sky
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kb-single
  template:
    metadata:
      name: kb-single
      labels:
        app: kb-single
    spec:
      containers:
      - image: hub.sky-cloud.net/k8s/kibana:6.4.0
        name: kb
        ports:
        - name: http
          containerPort: 5601
        volumeMounts:
        - name: tz
          mountPath: /etc/localtime
        - name: config
          mountPath: /usr/share/kibana/config
      volumes:
      - name: tz
        hostPath:
          path: /etc/localtime
      - name: config
```

```
      configMap:
        name: kibana-config
        items:
        - key: kibana.yml
          path: kibana.yml
```

svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: kb-single-svc
  namespace: sky
spec:
  externalIPs:              # 暴露Service到外部IP
  - 192.168.1.146               # IP
  ports:
  - name: http
    port: 5601
    targetPort: 5601
  selector:
    app: kb-single
```

访问 http://192.168.1.146/api/sky-kibana



原文链接：使用 k8s 搭建 elk