**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICA MILITARĂ „FERDINAND I”**

**Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională**



**Student: Sd. Sg. Maj. Pesu Ciprian-George**

**Cuprins**

[**Cuprins** 2](#_Toc91057876)

[**Scopul Proiectului** 2](#_Toc91057877)

[**Diagrama servicii** 3](#_Toc91057878)

[**Apache Kafka :** 3](#_Toc91057879)

[**Apache ZooKeeper :** 4](#_Toc91057880)

[**Elasticsearch :** 4](#_Toc91057881)

[**Kibana :** 5](#_Toc91057882)

[**Kubernetes :** 5](#_Toc91057883)

[**Persistent Volumes :** 7](#_Toc91057884)

[**Persistent Volume Claims:** 7](#_Toc91057885)

[**MicroK8s :** 7](#_Toc91057886)

[**Serviciu de crawling pentru fluxurile RSS :** 7](#_Toc91057887)

[**Flux RSS :** 7](#_Toc91057888)

[**Implementare :** 8](#_Toc91057889)

[**Integrare Kafka :** 8](#_Toc91057890)

[**Crearea imaginii pentru container :** 9](#_Toc91057891)

[**Serviciu de crawling pentru știri :** 9](#_Toc91057892)

[**Implementare :** 10](#_Toc91057893)

[**Integrare Kafka :** 10](#_Toc91057894)

[**Crearea imaginii pentru container :** 10](#_Toc91057895)

[**Serviciu ce v-a gestiona conexiunea cu Elasticsearch :** 11](#_Toc91057896)

[**Crearea imaginii pentru container :** 12](#_Toc91057897)

[**Clustrer Kubernates** 13](#_Toc91057898)

[**Concluzii** 16](#_Toc91057899)

[**Referinte** 16](#_Toc91057900)

# 

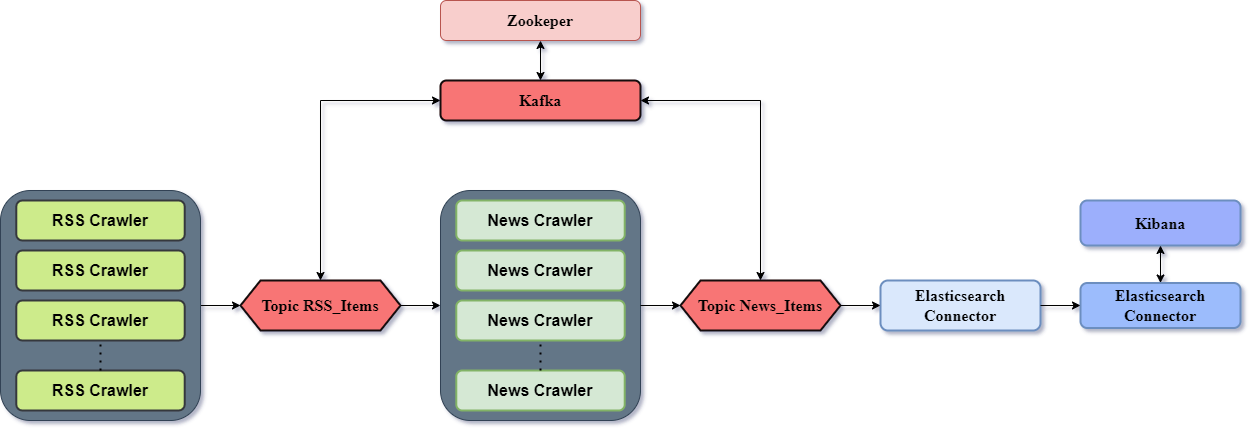
# **Scopul Proiectului**

Scopul proiectului este realizarea unei arhitecturi de servicii pentru procesarea, stocarea si centralizarea de știri în limba engleză publicate în internet (fluxuri RSS preluate de pe site-uri consacrate de știri) .

Această platformă va fi compusă din:

* Un serviciu de crawling ce va parsa fluxurile RSS ale platformelor de știri .
* O componenta de ingestie a acestor știri folosind cozi de mesaje (ex. Apache Kafka)
* Un serviciu de crawling ce va extrage textul si conținutul știrilor extrase din fluxurile RSS.
* O componenta de stocare si centralizare a datelor bazata pe Elasticsearch
* Un serviciu ce v-a gestiona conexiunea cu Elasticsearch
* O interfață de tip web care va permite vizualizarea știrilor cat si a datelor statistice în timp real

## **Diagrama servicii**

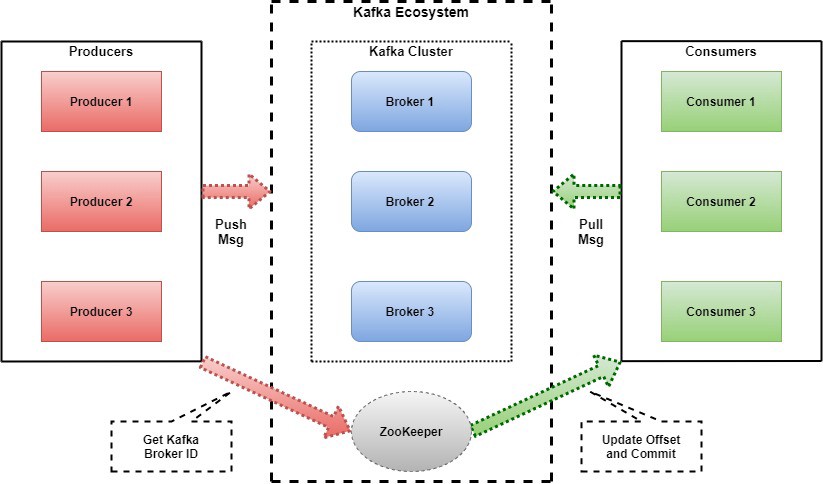


**Tehnologii folosite**

## **Apache Kafka :**

Kafka stochează mesaje cheie-valoare care pot provii de la multe procese numite producători. Datele pot fi împărțite în diferite partiții în cadrul diferitelor subiecte(„topics”). În cadrul unei partiții, mesajele sunt ordonate strict în funcție de offset(poziția unui mesaj în cadrul unei partiții) și sunt indexate și stocate împreună cu un timestamp. Alte procese numite „consumatori” pot citi mesaje din partiții.

Kafka rulează pe un cluster de unul sau mai multe servere (numite brokeri), iar partițiile tuturor subiectelor sunt distribuite în nodurile clusterului. În plus, partițiile sunt replicate pe mai mulți brokeri pentru a sporii redundanta datelor. Această arhitectură îi permite lui Kafka să livreze fluxuri masive de mesaje într-un mod tolerant la defecte



## **Apache ZooKeeper :**

Apache ZooKeeper este un serviciu care se specializează pe configurarea si sincronizarea sistemele distribuite. In arhitectura propusa Zookeeper ține evidența stărilor nodurilor clusterului Kafka si a topicelor si partițiilor acestuia. Acesta se asigura ca mai mulți consumatori sau producători lucrează simultan cu aceleași date din topicele clusterului Kafka

## **Elasticsearch :**

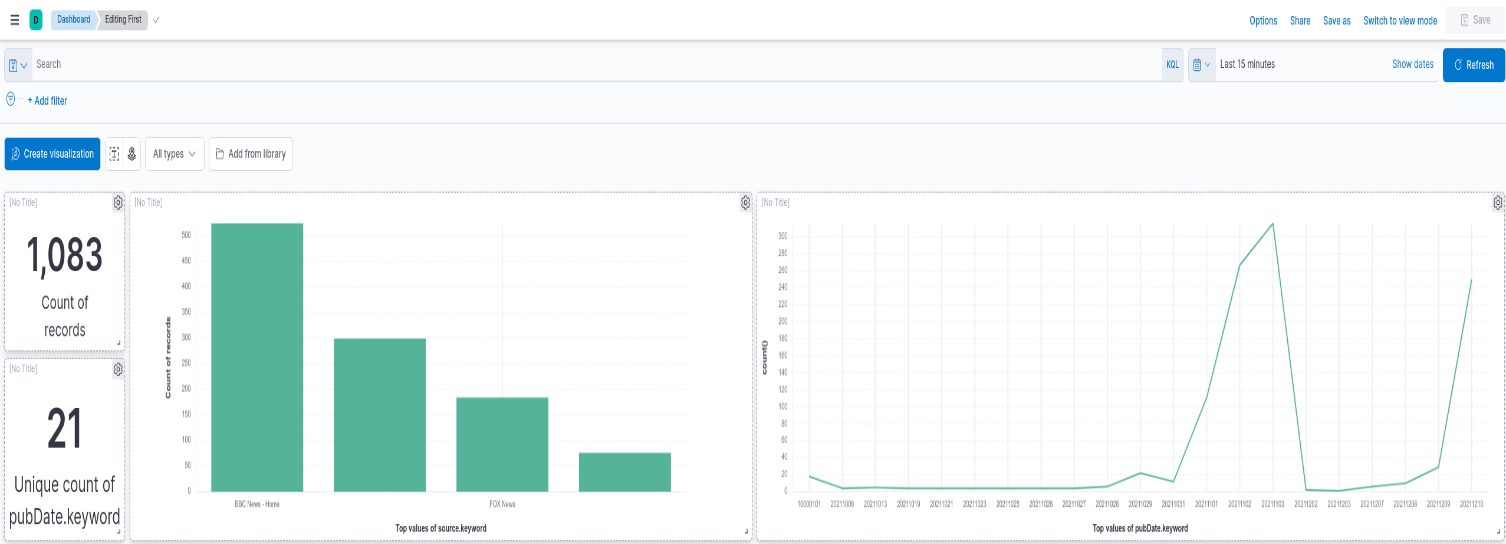
Elasticsearch este un motor de căutare și analiză distribuit construit pe Apache Lucene. Acesta permite stocarea, căutarea și analiza a volume uriașe de date rapid, oferind răspunsuri la interogări în milisecunde. Aceasta eficienta se datorează modului in care stochează datele, folosind documente JSON si indici.

Documentele sunt unitatea de bază de informații care poate fi indexată în Elasticsearch. Acestea pot fi orice structurate codificate în JSON precum text, numere, vectori sau date binare. Fiecare document are un ID unic și un anumit tip de date, care descrie ce fel de entitate este documentul.

Un index este o colecție de documente care au caracteristici similare. Acesta reprezintă entitatea de cel mai înalt nivel la care puteți interoga în Elasticsearch. Elementele dintr-un index sunt de obicei legate logic similar cu un tabel dintr-o baza de date relaționala.

## **Kibana :**

Kibana permite căutarea și vizualizarea datelor indexate în cluster Elasticsearch și analizarea acestora prin crearea de grafice, diagrame circulare, tabele, histograme și hărți prin intermediul unei interfețe web . In arhitectura propusa Kibana este expusa pe portul 30601 in rețeaua locala.



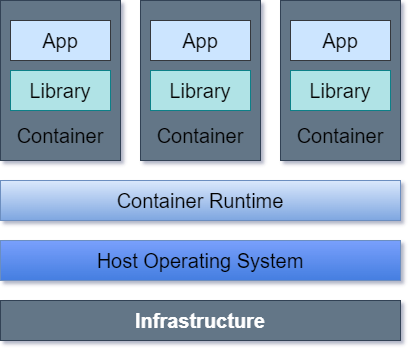
## **Kubernetes :**

Kubernetes este un sistem open-source pentru automatizarea implementării, scalarea și gestionarea aplicațiilor containerizate.

Un cluster Kubernetes este format dintr-un set de mașini de lucru, numite noduri, care rulează aplicații containerizate, fiecare cluster are cel puțin un nod de lucru. Pentru a permite rularea continerelor fiecare nod din clasterul Kubernetes trebuie sa aibă instalat un **continer runtime**. Pentru arhitectura noastră am ales utilizarea **Docker Engine** in acest rol deoarece la momentul actual acesta este cel mai utilizat si testat runtime disponibil.

**O imagine de container** este un pachet software, independent și executabil, care include tot ceea ce este necesar pentru a rula o aplicație: cod, sistemului de operare (OS și dependențele necesare pentru a rula codul **.**

**Containerele**  sunt similare cu o mașinile virtuale , dispun de propriul sistem de fișiere, memorie, spațiu de stocare si pot comunica intre ele prin canale bine definite . Acesta este creat dintr-o imagine de continer de catre continer runtime.



Kubernetes definește un set de blocuri de construcție ("primitivele" bazate pe CPU, memorie  sau resurse specifice), care împreună oferă un mecanism ce implementează, mențin și scalează aplicațiile. Acestea sunt:

* **Pods -** sunt obiecte de bază ale Kubernetes responsabile de încapsularea containerelor, resurselor de stocare și IP-urilor de rețea.
* **Delpoyments -** sunt obiecte Kubernetes care gestionează podurile. Acestea permit precizarea numărului de poduri pe care dorim sa le pornim cat si argumentele acestora.
* **Services -** reprezintă o modalitate abstractă de a expune o aplicație care rulează pe un set de Pods ca serviciu de rețea.

### **Persistent Volumes :**

Un PersistentVolume (PV) este o resursa de stocare dintr-un cluster care a fost furnizată de un administrator sau furnizată dinamic folosind Storage Class. PV-urile au un ciclu de viață independent fata de podul de care este utilizat , permițând astfel ca datele din acesta sa rămână disponibile si dapăr ștergerea sau repornirea podului .

## **Persistent Volume Claims:**

Un PersistentVolumeClaim (PVC) este o solicitare de spațiu de stocare a unui utilizator. Este similar cu un Pod. Pod-urile consumă resurse de nod, iar PVC-urile consumă resurse PV. Pod-urile pot solicita niveluri specifice de resurse (CPU și memorie). Revendicările pot solicita anumite dimensiuni și moduri de acces

### **MicroK8s :**

MicroK8s este un sistem open-source pentru automatizarea implementării, scalarea și gestionarea aplicațiilor containerizate. Oferă principale funcționalități oferite de Kubernetes cu un consum mai mic de resurse. Datorita resurselor limitate pe care le avem la dispoziție am ales sa folosesc Microk8s pentru arhitectura de servicii descrisa.

# **Serviciu de crawling pentru fluxurile RSS :**

## **Flux RSS :**

Un flux RSS preia titlul, rezumatul ,link-ul ,data si timpul publicaților site-ului și le încapsulează intr-un fișier XML. Acest conținut este distribuit în timp real, astfel încât primele rezultate din fluxul RSS să fie întotdeauna cel mai recent conținut publicat pentru un site web.

In urma cercetării am ales trei fluxuri RSS aparținând unor publicații de renume pe plan mondial

**BBC -** [**http://feeds.bbci.co.uk/news/rss.xml**](http://feeds.bbci.co.uk/news/rss.xml)

**CNN -** [**http://rss.cnn.com/rss/edition.rss**](http://rss.cnn.com/rss/edition.rss)

**FOX -** [**http://feeds.foxnews.com/foxnews/latest**](http://feeds.foxnews.com/foxnews/latest)

## **Implementare :**

Programul este implementat in limbajul de programare Java si primește ca argument un link către un flux RSS . Pentru a evita o utilizare neeficienta a resurselor, o data la 10 minute, programul accesează linkul primit da argument, preia fișierul XML si verifica daca au apărut știri noi fata de ultima verificare. In cazul in care au fost detectate noi adiții, crawlerul le preia si le trimite topicul-ui Kafka responsabil in format JSON.

Pentru o mai buna vizualizare a știrilor pe care le extrage programul din fluxul RSS am implementat o componenta de logare care salvează in fișierul „Logs/log.txt” logurile de forma :

***STARTING AT 2021-10-28 at 08:55:47 EEST***

***2021-10-28 at 08:55:48 --> There has been a shift in Tory philosophy - minister***

***2021-10-28 at 08:56:48 --> Alec Baldwin shooting: Criminal charges may be filed,***

## **Integrare Kafka :**

Deoarece acest serviciu reprezintă punctul de intrare a datelor in pipeline-ul nostru de servicii, el nu are rolul de consumator pentru nici unul dintre topicele Kafka însă are rolul de producător pentru topicul „Items\_RSS ”.

Exemplu obiect JSON trimis către topicul Kafka „Items\_RSS ” :

**{**

**"link": "https://www.bbc.co.uk/sport/football/59033038 ",**

**"description": "Wolves' impressive Premier League form continues as they inflict a defeat on Everton which will surely worry beaten manager Rafael Benitez.",**

**"source": "BBC News - Home",**

**"title" : "Wolves' fine run continues as sloppy Everton punished again",**

**"pubDate": "20211101",**

**"pubTime": "22:32:52",**

**}**

## **Crearea imaginii pentru container :**

Imaginea a fost creata dintr-o imagine de Ubuntu in care a fost copiat fișierul jar rezultat din codul programului . Pentru a putea rula programul a mai fost necesara instalarea OpenJDK-8 si pornirea procesului conform următorului DockerFile:

**FROM ubuntu:latest**

**COPY ./myapp/\* /usr/src/myapp/**

**WORKDIR /usr/src/myapp**

**RUN apt-get update && apt-get install -y openjdk-8-jre**

**RUN chmod 755 /usr/src/myapp/phantomjs**

**RUN java -jar rssparser.jar**

# **Serviciu de crawling pentru știri :**

## **Implementare :**

Programul este implementat in limbajul de programare Python si are ca scop preluarea conținutului text a știrilor publicate in online .

Crawlerul își începe execuția când recepționează un mesaj de la topicul Kafka „Items\_RSS ”. Acesta extrage codul HTML al paginii la care pointeaza linkul din documentul JSON primit , elimina tag-urile HTML si identifica începutul si finalul știri propriu-zise. După extragerea textului dorit , acesta este adăugat la documentul primit inițial si trimis la topicul Kafka „Items\_news ”

## **Integrare Kafka :**

Acest serviciu are rolul de consumator pentru topicul Kafka „Items\_RSS” si de producător pentru topicul „Items\_news” .

Exemplu JSON trimis către topicul Kafka „Items\_news ”:

**{**

**"link": "https://www.bbc.co.uk/sport/football/59033038? ",**

**"description": "Wolves' impressive Premier League form continues as they inflict a defeat on Everton which will surely worry beaten manager Rafael Benitez.",**

**"source": "BBC News - Home",**

**"title": "Wolves' fine run continues as sloppy Everton punished again",**

**"pubDate": "20211101",**

**"pubTime": "22:32:52",**

**"content": " Max Kilman's first Wolves goal and Raul Jimenez's 50th were enough to send Bruno Lage's men into seventh place in the Premier League and condemn Everton to a third successive defeat. Wolves dominated the opening period of an entertaining contest and were worthy of a comfortable half-time cushion provided by Kilman's far-post header and a lovely clipped finish ...”**

**}**

## **Crearea imaginii pentru container :**

Imaginea a fost creata dintr-o imagine de Ubuntu in care a fost copiat scriptul de Python . Pentru a putea rula programul au fost instalate dependințele de Python prin intermediul utilitarului pip iar apoi lansat in execuție scriptul conform următorului DockerFile :

**FROM ubuntu:latest**

**COPY url\_crawler.py /usr/src/myapp/url\_crawler.py**

**WORKDIR /usr/src/myapp**

**RUN apt-get update && apt-get install -y python3**

**RUN apt-get update && apt-get install -y python3-pip**

**RUN pip install --no-input Unidecode**

**RUN pip install --no-input confluent-kafka**

**RUN pip install --no-input BeautifulSoup4**

**RUN pip install --no-input goose3**

**RUN python3 url\_crawler.py**

# **Serviciu ce v-a gestiona conexiunea cu Elasticsearch :**

Programul este implementat in limbajul de programare Python si are ca scop stabilirea conexiunii cu ElasticSearch in introducerea mesajelor din topicul Kafka „News\_Items” in componenta de stocare a acestuia.

## **Crearea imaginii pentru container :**

Imaginea a fost creata dintr-o imagine de Ubuntu in care a fost copiat scriptul de Python . Pentru a putea rula programul au fost instalate dependințele de Python prin intermediul utilitarului pip iar apoi lansat in execuție scriptul conform următorului DockerFile :

**FROM ubuntu:latest**

**COPY ElasticConnection.py /usr/src/myapp/ElasticConnection.py**

**COPY bash.sh /usr/src/myapp/bash.sh**

**WORKDIR /usr/src/myapp**

**RUN apt-get update && apt-get install -y python3**

**RUN apt-get update && apt-get install -y python3-pip**

**RUN apt-get update && apt-get install -y gcc**

**RUN pip install --no-input confluent-kafka**

**RUN pip install --no-input elasticsearch**

**RUN python3 elastic\_conection.py**

# **Clustrer Kubernates**

Kubernates oferă posibilitatea automatizării deployment-urilor prin intermediul fișierelor de configurare YAML. Pentru a inițializa arhitectura de servicii propusa am configurat un fișier YAML pentru fiecare deployment necesar .

**Url\_crawler.yaml** pornește containerul pentru serviciu de crawling pentru fluxurile RSS pe un număr de poduri precizat ( in arhitectura prezentata numărul de poduri pornite este 1)

**Volumes.yaml** definește doua Persistent Volume Claims necesare pentru a asigura persistenta datelor stocate :

* elastic-volume are o dimensiune de 3Gi si are rolul de a deservi containerul de Elasticsearch
* rssparser-volume are o dimensiune de 1Gi deservește containerul destinat serviciu de crawling pentru fluxurile RSS

**Elastic-connect.yaml** pornește serviciul care asigura introducerea datelor prelucrate de pipeline-ul nostru un in componenta de stocare Elasticsearch.

**RSSparser.yaml** pornește containerul pentru de crawling pentru știripe un număr de poduri precizat ( in arhitectura prezentata numărul de poduri pornite este 1) cu variabilele de environment necesare setate:

* **volumeMounts:**

**name: rssparser-volume**

**mountPath: /usr/src/myapp/Logs/**

Aceasta variabila precizează ce cale din sistemul de fișiere ale containerului sa rămână pe spațiul de stocare al PV-ului atribuit

**Zoo.yaml** acest fișier are rolul de a porni containerul de ZooKeeper si de a expune portul sau 2181 ( necesar pentru conectarea serviciului Kafka ) in rețeaua locala prin intermediul unui serviciu

**Kibana.yaml** acest fișierul are rolul de a porni containerul de Kibana cu valorile de environment necesare pentru interconectarea cu Elasticsearch precum si un serviciu ce expune portul 5601 al interfeței web oferite de Kibana pe portul 30601 in rețeaua locala

* **name: ELASTICSEARCH\_HOSTS**
  + **value:** [**http://elasticsearch-9200:9200/**](http://elasticsearch-9200:9200/)

**ELASTICSEARCH\_HOST reprezintă adresa URL care trebuie folosita de Kibana pentru toate interogările.**

**Elasticsearch.yaml** are rolul de a inițializa containerul de Elasticsearch si setarea valorilor de environment necesare pentru configurarea serviciului cat si expunerea porturilor 9200 si 9300 in rețeaua locala:

* **name: discovery.type**
  + **value: single-node**
* **name: ES\_JAVA\_OPTS**
  + **value: -Xms512m -Xmx512**

**discovery.type** specifică dacă Elasticsearch ar trebui să formeze un cluster cu mai multe noduri sau sa ruleze doar pe un singur nod .

**ES\_JAVA\_OPTS** utilizat împreuna cu valoarea -Xms512m -Xmx512 limitează programul la folosirea a maxim 512mb din memoria RRAM a sistemului **.**

**Kafka.yaml** acest fișier are rolul de containerul de Kafka si setarea valorilor de environment necesare pentru configurarea serviciului cat si expunerea portului 2181 in rețeaua locala:

* **name: KAFKA\_LISTENERS**
  + **value: 10.0.0.1:2181**
* **name: KAFKA\_ZOOKEEPER\_CONNECT**
  + **value: 10.0.0.1:2181**
* **name: KAFKA\_CREATE\_TOPICS**
  + **value: rss\_**[**data:1:1,crawled\_news:1:1**](data:1:1,crawled_news:1:1)
* **name: KAFKA\_ADVERTISED\_HOST\_NAME**
  + **value:** [**localhost**](http://elasticsearch-9200:9200/)

**KAFKA\_LISTENERS** este o lista, separată prin virgulă, precum IP:PORT pe care le asculta Kafka pentru pachete inbound .

**KAFKA\_ZOOKEEPER\_CONNECT** reprezintă adresa si portul prin care Kafka poate comunica cu ZooKeeper .

**KAFKA\_CREATE\_TOPICS** este o variabila de environment prin intermediul căreia la inițializarea containerului se declara topicele necesare

**KAFKA\_ADVERTISED\_HOST\_NAME** reprezintă hostname-ul pe care rulează Kafka

# **Concluzii**

In concluzie arhitectura prezentată reprezintă un cluster peste Kubernetes, scalabil si tolerant la defecte, cu capabilitatea de fii extins prin adăugarea a unor multitudini de noi servicii în pipeline-ul facilitat de componenta Kafka. a

**Bibliografie**

* [Elastic Cloud on Kubernetes | Deploy and Orchestrate Elasticsearch on Kubernetes](https://www.elastic.co/elastic-cloud-kubernetes)
* [Deploy a Kibana instance | Elastic Cloud on Kubernetes [1.9] | Elastic](https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-deploy-kibana.html)
* [MicroK8s - Zero-ops Kubernetes for developers, edge and IoT](https://microk8s.io/)
* [Apache ZooKeeper](https://zookeeper.apache.org/)
* [Apache Kafka](https://kafka.apache.org/)