Bab 11

Logging & Monitoring Server part 1

SEKOLAH DEVOPS CILSY

Detail Materi

Pengenalan Logging

Indikator
Memahami konsep logging pada Kubernetes - Memahami cara melakukan konfigurasi EFK pada Kubernetes - Memahami cara membaca data pada log yang tersedia

Implementasi EFK pada Kubernetes

Modul Sekolah DevOps Cilsy

Penulis : Adi Saputra, Irfan Herfiandana, Tresna Widiyaman, Estu Fardani
Editor: Taufik Maulana, Muhammad Fakhri Abdillah, Iqbal Ilman, Rizal Rahman & Tresna Widiyaman

Revisi Batch 9

Penerbit : **PT. Cilsy Fiolution Indonesia**
Web Site : https://cilsyfiolution.com , https://devops.cilsy.id

Daftar Isi

11.
Logging dan Monitoring Server: EFK pada Kubernetes

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami konsep logging pada Kubernetes
2. Memahami cara melakukan konfigurasi EFK pada Kubernetes
3. Memahami cara membaca data pada log yang tersedia

Outline Materi

1. Pengenalan Logging System
2. Instalasi EFK Stack
3. Kesimpulan

Sumber Bacaan: https://www.digitalocean.com/community/tutorials/how-to-set-up-an-elasticsearch-fluentd-and-kibana-efk-logging-stack-on-kubernetes

11.1. Latar Belakang

Saat menjalankan banyak layanan dan aplikasi pada cluster Kubernetes terpusat, logging dan monitoring pada cluster dapat membantu kami menganalisis data log besar yang dihasilkan dari Pod. Kita dapat menggunakan Elasticsearch, Fluentd dan Kibana (EFK) sebagai salah satu solusi logging terpusat.

11.1.1. EFK Stack

Elasticsearch adalah mesin pencari real-time, terdistribusi, dan skalabel untuk pencarian dan analisis teks terstruktur yang lengkap. Elasticsearch biasanya digunakan untuk mengindeks dan mencari data log dalam jumlah besar, dan juga dapat digunakan untuk mencari berbagai jenis dokumen.

Elasticsearch biasanya digunakan dengan Kibana (data visualizer untuk memvisualisasikan data dan dasbor Elasticsearch). Kibana memungkinkan kita untuk menelusuri data log Elasticsearch melalui interface web dan membuat dasbor dan kueri dengan cepat dan mendapatkan insight dari aplikasi Kubernetes. Dalam praktikum kali ini, kita akan menggunakan Fluentd untuk mengumpulkan, memodifikasi, dan mengirim data log ke Elasticsearch. Fluentd adalah pengumpul data open source yang populer. Kita akan menyiapkannya di node Kubernetes untuk mengekstrak file log dari container, memfilter dan mengubah data log, dan mengirimkannya ke cluster Elasticsearch, tempat log akan diindeks dan disimpan.

Sebelum kita mulai praktek ini, pastikan kita memiliki hal-hal berikut:

- Kubernetes cluster dengan kontrol akses berbasis peran (RBAC) diaktifkan.

- Pastikan cluster kita memiliki resource yang cukup untuk memulai EFK Stack. Jika tidak, upgrade cluster dengan menambahkan worker node. Kami akan menggunakan 3-Pod Elasticsearch cluster (dapat dikurangi menjadi 1 jika perlu) dan 1 Kibana Pod. Setiap node pekerja juga akan menjalankan Fluentd Pod.
- Kubectl command line. tools yang dipasang pada workspace kita, dikonfigurasi untuk menyambungkan ke kluster.

11.2. Instalasi EFK Stack

11.2.1. Setup Minikube

Pada praktik kali ini, kita akan membuat suatu kluster **elasticsearch**, **kibana** dan **fluentd** pada **minikube** yang berjalan diatas **docker**. Kita akan membuat kluster elasticsearch dengan 1 replica saja. Kita akan melakukan setup Minikube dengan perintah **minikube start --driver=docker**

```
controlplane $ minikube start –driver=docker
```

Pastikan kita mendapatkan output dari perintah **minikube status**

```
controlplane $ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent
```

11.2.2. Membuat Namespaces

Sebelum melangkah lebih jauh ke cluster Elasticsearch, langkah pertama yang harus kita ambil adalah membuat namespace untuk semua alat logging. Kubernetes memungkinkan kita menggunakan cluster virtual yang disebut namespace untuk memisahkan objek yang sedang berjalan. Sebelum memulai konfigurasi, mari kita gunakan perintah berikut untuk melihat namespace yang sudah ada di cluster:

```
kubectl get namespaces
```

Output dari perintah tersebut adalah sebagai berikut

```
NAME          STATUS    AGE
default       Active    5m
kube-system   Active    5m
kube-public   Active    5m
```

Namespace **default** berisi objek yang dibuat tanpa menentukan namespace. Namespace **system** kube berisi objek yang dibuat dan digunakan oleh sistem Kubernetes, seperti kube-dns, kube-proxy, dan kubernetes-dashboard. Namespace **kube-public** adalah namespace yang otomatis dibuat, dan dapat digunakan untuk menyimpan objek yang nantinya dapat dibaca dan diakses melalui seluruh cluster, bahkan oleh unauthenticated user.

Pada modul ini, kita akan membuat namespaces bernama **kube-logging**. Buatlah file yaml menggunakan perintah berikut

```
nano efk-ns.yaml
```

Masukkan konfigurasi yaml berikut

```
kind: Namespace
apiVersion: v1
metadata:
  name: kube-logging
```

script dapat dilihat di
https://gist.github.com/sdcilsy/900af2f31314de83c9bb9ff3c78bd8a2

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Setelah membuat file kube-logging.yaml, buatlah namespace menggunakan perintah berikut

```
controlplane $ kubectl apply -f efk-ns.yaml
namespace/kube-logging created
```

Kita dapat memastikan namespace telah terbuat dengan perintah **kubectl get ns**

```
Controlplane $ kubectl get ns
NAME                                 STATUS   AGE
default                              Active   23m
kube-logging                         Active   1m
kube-public                          Active   23m
kube-system   Active   23m
```

Sekarang kita sudah bisa memulai membuat deployment Elasticsearch cluster pada namespace kube-logging.

11.2.3. Membuat Elasticsearch StatefulSet

Sekarang setelah kita membuat namespace untuk menyimpan EFK Stack, kita dapat mulai mengonfigurasi komponen yang tersisa. Pertama, kami akan menerapkan cluster Elasticsearch 1-node.

11.2.3.1. Membuat Headless Service

Sebelum melakukan konfigurasi pada elasticsearch kita harus menentukan domain DNS untuk ketiga node. Pada tahap selanjutnya kita akan membuat file configurasi.

Buatlah file yaml menggunakan perintah berikut.

```
nano p-elasticsearch-svc.yaml
```

Masukan konfigurasi berikut

```
kind: Service
apiVersion: v1
metadata:
  name: elasticsearch
  namespace: kube-logging
  labels:
    app: elasticsearch
spec:
  selector:
    app: elasticsearch
  clusterIP: None
  ports:
    - port: 9200
      name: rest
    - port: 9300
      name: inter-node
```

script dapat dilihat di

https://gist.github.com/sdcilsy/c15ca69f3a118aa5ad28b4d031dbe0a7

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Buat service menggunakan perintah berikut.

```
kubectl apply -f p-elasticsearch-svc.yaml
service/elasticsearch created
```

Setelah melakukan konfigurasi di atas cek kembali apakah layanan telah berhasil dibuat dengan menggunakan perintah.

```
Controlplane $ kubectl get svc -n kube-logging
NAME          TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)             AGE
elasticsearch    ClusterIP    None            <none>          9200/TCP,9300/TCP
112s
```

11.2.3.2. Membuat StatefulSet

Kubernetes statefulset memungkinkan kita untuk menetapkan penetapan nama ke node dan menyediakan penyimpanan yang stabil dan tahan lama. Saat rescheduling dan restarting, Elasticsearch membutuhkan penyimpanan yang stabil untuk menjaga data. Sebelum memulai konfigurasi, kita akan membuat file yaml untuk statefulset.

Buatlah file yaml menggunakan perintah

```
nano p-elasticsearch-statefulset.yaml
```

Masukan konfigurasi berikut

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: es-cluster
  namespace: kube-logging
spec:
  serviceName: elasticsearch
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
```

Kita akan masuk pada spec objek di file konfigurasi yang sama, masukan konfigurasi dibawah konfigurasi sebelumnya

```
spec:
     containers:
     - name: elasticsearch
       image: docker.elastic.co/elasticsearch/elasticsearch:7.11.2
       resources:
          limits:
            cpu: 1000m
          requests:
            cpu: 100m
```

```
        ports:
        - containerPort: 9200
          name: rest
          protocol: TCP
        - containerPort: 9300
          name: inter-node
          protocol: TCP
        volumeMounts:
        - name: data
          mountPath: /usr/share/elasticsearch/data
        env:
          - name: cluster.name
            value: k8s-logs
          - name: node.name
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: discovery.seed_hosts
            value: "es-cluster-0.elasticsearch"
          - name: cluster.initial_master_nodes
            value: "es-cluster-0"
          - name: ES_JAVA_OPTS
            value: "-Xms512m -Xmx512m"
```

Selanjutnya tambahkan konfigurasi di file yang sama untuk mendefinisikan beberapa Kontainer Init yang berjalan sebelum main aplikasi elasticsearch utama. Masukan file konfigurasi sebagai berikut

```
initContainers:
      - name: fix-permissions
        image: busybox
        command: ["sh", "-c", "chown -R 1000:1000
/usr/share/elasticsearch/data"]
        securityContext:
          privileged: true
        volumeMounts:
        - name: data
          mountPath: /usr/share/elasticsearch/data
      - name: increase-vm-max-map
        image: busybox
        command: ["sysctl", "-w", "vm.max_map_count=262144"]
        securityContext:
          privileged: true
```

```
      - name: increase-fd-ulimit
        image: busybox
        command: ["sh", "-c", "ulimit -n 65536"]
        securityContext:
          privileged: true
```

Terakhir tambahkan baris konfigurasi untuk mendefinisikan **volume claim** untuk pod yang akan dibuat. Kita menggunakan storage class **standard** untuk bembuat volume claimnya

```
volumeClaimTemplates:
  - metadata:
      name: data
      labels:
        app: elasticsearch
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: standard
      resources:
        requests:
          storage: 10Gi
```

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Keseluruhan script dapat dilihat di

https://gist.github.com/sdcilsy/06bd8ba45e97591220e4474b89841103

Jalankan file konfigurasi menggunakan perintah sebagai berikut

```
controlplane $ kubectl apply -f p-elasticsearch-statefulset.yaml
statefulset.apps/es-cluster created
```

Kita bisa melakukan monitoring StatefulSet menggunakan perintah berikut

```
controlplane $ kubectl rollout status sts/es-cluster -n kube-logging
Waiting for 1 pods to be ready...
partitioned roll out complete: 1 new pods have been updated...
```

Langkah selanjutnya kita perlu melakukan forwarding untuk menguji layanan elasticsearch menggunakan perintah berikut

```
controlplane $ kubectl port-forward es-cluster-0 9200:9200 --namespace=kube-logging
```

Lakukan pada terminal terpisah untuk melakukan permintaan curl terhadap REST API menggunakan perintah berikut

```
curl http://localhost:9200/_cluster/state?pretty
```

Output dari perintah tersebut adalah sebagai berikut

```
{
  "name" : "es-cluster-0",
  "cluster_name" : "k8s-logs",
  "cluster_uuid" : "1oL_7xnzTSe1ueI-OcYlqA",
  "version" : {
    "number" : "7.11.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "3e5a16cfec50876d20ea77b075070932c6464c7d",
    "build_date" : "2021-03-06T05:54:38.141101Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

jika output muncul, maka kita bisa melanjutkan ke langkah selanjutnya. Kita bisa mematikan perintah **curl http://localhost:9200/_cluster/state?pretty** dengan kombinasi tombol **Ctrl + c**.

11.2.4. Membuat Kibana Deployment and Service

Setelah dengan 2 langkah sebelumnya kita akan melakukan konfigurasi pada kibana, dimana kita bisa melakukan replika dan kita dapat meng scale jumlah replika tergantung pada kebutuhan produksi. Sebelum itu kita perlu membuat membuat file konfigurasi untuk kibana deployment and server.

Buatlah file yaml menggunakan perintah berikut

```
nano p-kibana-dpy.yaml
```

Tambahkan konfigurasi berikut

```
apiVersion: v1
kind: Service
metadata:
  name: kibana
  namespace: kube-logging
  labels:
    app: kibana
spec:
  ports:
  - port: 5601
  selector:
    app: kibana
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kibana
  namespace: kube-logging
  labels:
    app: kibana
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kibana
  template:
    metadata:
      labels:
        app: kibana
    spec:
      containers:
      - name: kibana
        image: docker.elastic.co/kibana/kibana:7.11.2
        resources:
          limits:
            cpu: 1000m
          requests:
            cpu: 100m
        env:
          - name: ELASTICSEARCH_URL
            value: http://elasticsearch:9200
```

```
    ports:
    - containerPort: 5601
```

sricpt dapat dilihat di

https://gist.github.com/sdcilsy/82a197bf770671c64ee7bc0a14f7dee1

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Jalankan file konfigurasi yang kita buat dengan perintah

```
Controlplane $ kubectl apply -f p-kibana-dpy.yaml
service/kibana created
deployment.apps/kibana created
```

Kita bisa melakukan pengecekan apakah kibana telah berjalan dengan baik menggunakan perintah **kubectl rollout status deployment -n kube-logging kibana**

```
Controlplane $ kubectl rollout status deployment -n kube-logging kibana
deployment "kibana" successfully rolled out
```

Untuk melakukan akses interface kibana kita perlu melakukan forwarding local port node kubernetes yang menjalankan kibana dengan perintah berikut.

```
Controlplane $ kubectl get pods -n kube-logging
```

Output yang dari konfigurasi tersebut sebagai berikut

```
NAME                  READY   STATUS    RESTARTS   AGE
es-cluster-0          1/1     Running   0          6m15s
kibana-64b7cf9db7-4hqs9   1/1     Running   0          21m
```

Kita bisa melihat node kibana yang berjalan dengan nama **kibana-64b7cf9db7-4hqs9** (nama tidak akan sama) yang kemudian kita perlu lakukan forwarding local port 5601 to port 5601 dengan perintah berikut

```
Controlplane $ kubectl port-forward kibana-64b7cf9db7-4hqs9 5601:5601 -
n kube-logging
Forwarding from 127.0.0.1:5601 -> 5601
```

```
Forwarding from [::1]:5601 -> 5601
```

Sekarang buka pada web browser masing masing

```
http://localhost:5601
```

Jika tampilan halaman welcome kibana sudah muncul maka kita telah berhasil melakukan konfigurasi pada kibana pada cluster kubernetes

11.2.5. Membuat Domain Kibana

Silahkan atur subdomain efk.domain.id sebagai cname dari ELB Ingress. Kemudian buatlah file konfigurasi ingress seperti berikut:

```
nano p-ekf-ingress.yaml
```

Tambahkan konfigurasi berikut

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: efk-ingress
  namespace: kube-logging
  annotations:
    ## Untuk menaikan batas upload file
    nginx.org/client-max-body-size: "10m"
spec:
  rules:
  - host: efk.domain.id
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kibana
            port:
              number: 5601
```

script dapat dilihat di

https://gist.github.com/sdcilsy/d5555b889c2af53bf78ffc41e00b1a96

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Jalankan file konfigurasi yang kita buat dengan perintah

```
kubectl apply -f p-ekf-ingress.yaml
```

11.2.6. Membuat Fluentd DaemonSet

Pada langkah ini kita akan menetapkan Fluentd sebagai DaemonSet, yang merupakan jenis beban kerja pada Kubernetes yang menjalankan salinan node yang diberikan pada setiap Node di kluster Kubernetes. Dimana Fluentd Pod akan mengekor file-file log ini, menyaring peristiwa-peristiwa log, mentransformasikan data log, dan mengirimkannya ke backend logging Elasticsearch yang kita gunakan pada Langkah 2 bahkan Fluentd akan mengikuti log komponen sistem Kubernetes seperti kubelet, kube-proxy, dan log Docker. Untuk melihat daftar lengkap sumber yang di-tailed oleh agen logging Fluentd, lihat file kubernetes.conf.

Sebelum melakukan konfigurasi kita akan membuat file konfigurasi dengan menggunakan perintah berikut.

```
nano p-fluentd-dpy.yaml
```

Tambahkan pada file konfigurasi

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentd
  namespace: kube-logging
  labels:
    app: fluentd
```

Selanjutnya kita akan menambahkan konfigurasi untuk mendefinisikan ClusterRole yang disebut fluentd yang kita berikan izin get, list, dan watch pada objek pod dan namespaces sebagai berikut.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluentd
  labels:
    app: fluentd
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - namespaces
  verbs:
  - get
  - list
  - watch
```

Tambah konfigurasi berikut

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: kube-logging
```

Selanjutnya konfigurasi untuk mendefinisikan spesifikasi dari DaemonSet

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-logging
  labels:
```

```
    app: fluentd

spec:
  selector:
    matchLabels:
      app: fluentd
  template:
    metadata:
      labels:
        app: fluentd
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd
        image: fluent/fluentd-kubernetes-daemonset:v1.4.2-debian-
elasticsearch-1.1
        env:
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: "elasticsearch.kube-logging.svc.cluster.local"
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
          - name: FLUENT_ELASTICSEARCH_SCHEME
            value: "http"
          - name: FLUENTD_SYSTEMD_CONF
            value: disable
```

Terakhir untuk konfigurasi kita tambahkan untuk menspesifikasikan source yang akan digunakan sesuai kebutuhan.

```
resources:
          limits:
            memory: 512Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
```

```
      readOnly: true
  terminationGracePeriodSeconds: 30
  volumes:
  - name: varlog
    hostPath:
      path: /var/log
  - name: varlibdockercontainers
    hostPath:
      path: /var/lib/docker/containers
```

Keseluruhan script dapat dilihat di script dapat dilihat di

https://gist.github.com/sdcilsy/937017374d4440d02a5df8d84e676037

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Jalankan DaemonSet yang sudah tadi dikonfigurasi menggunakan perintah

```
kubectl apply -f p-fluentd-dpy.yaml
```

Output dari perintah tersebut

```
serviceaccount/fluentd created
clusterrole.rbac.authorization.k8s.io/fluentd created
clusterrolebinding.rbac.authorization.k8s.io/fluentd created
daemonset.extensions/fluentd created
```

Kita akan melakukan verifikasi apakah DaemonSet telah berjalan dengan menggunakan perintah berikut

```
kubectl get ds -n kube-logging
```

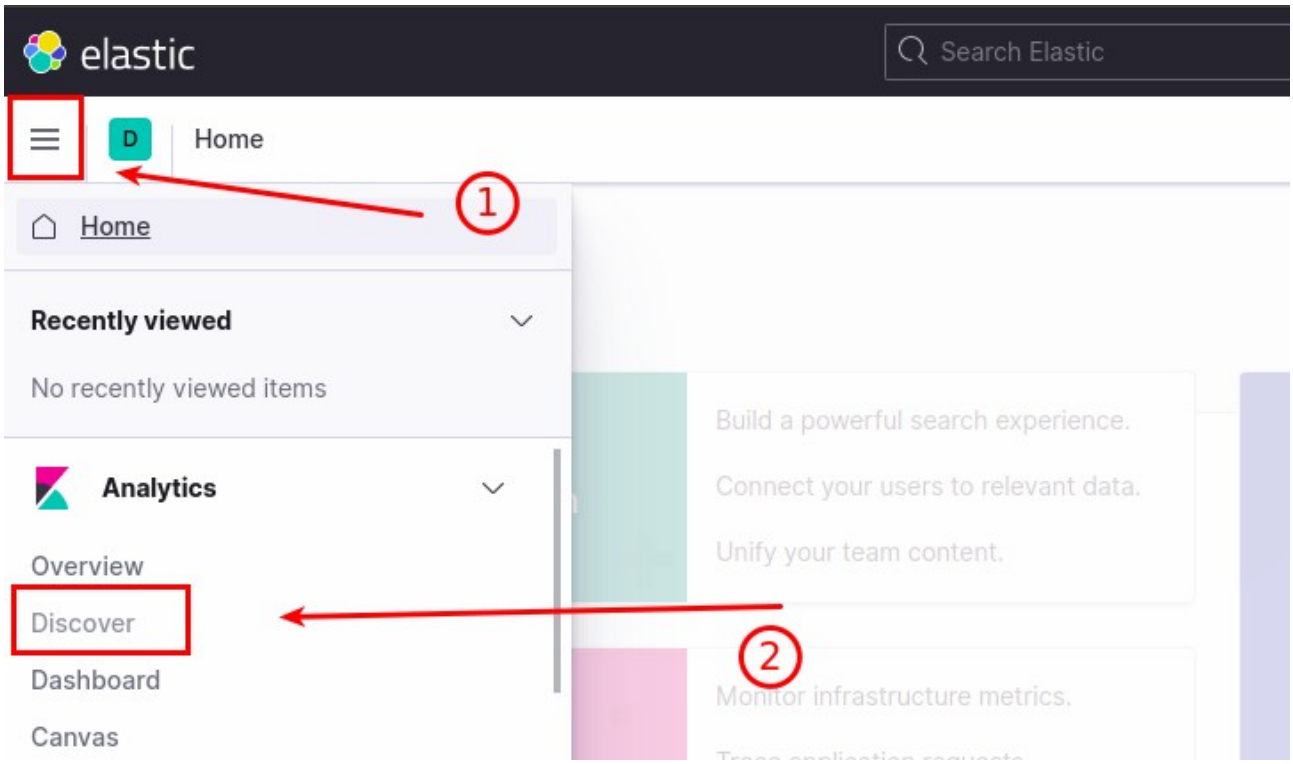
Output dari perintah tersebut

```
NAME     DESIRED   CURRENT   READY    UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
fluentd  1         1         1        1            1           <none>          99s
```

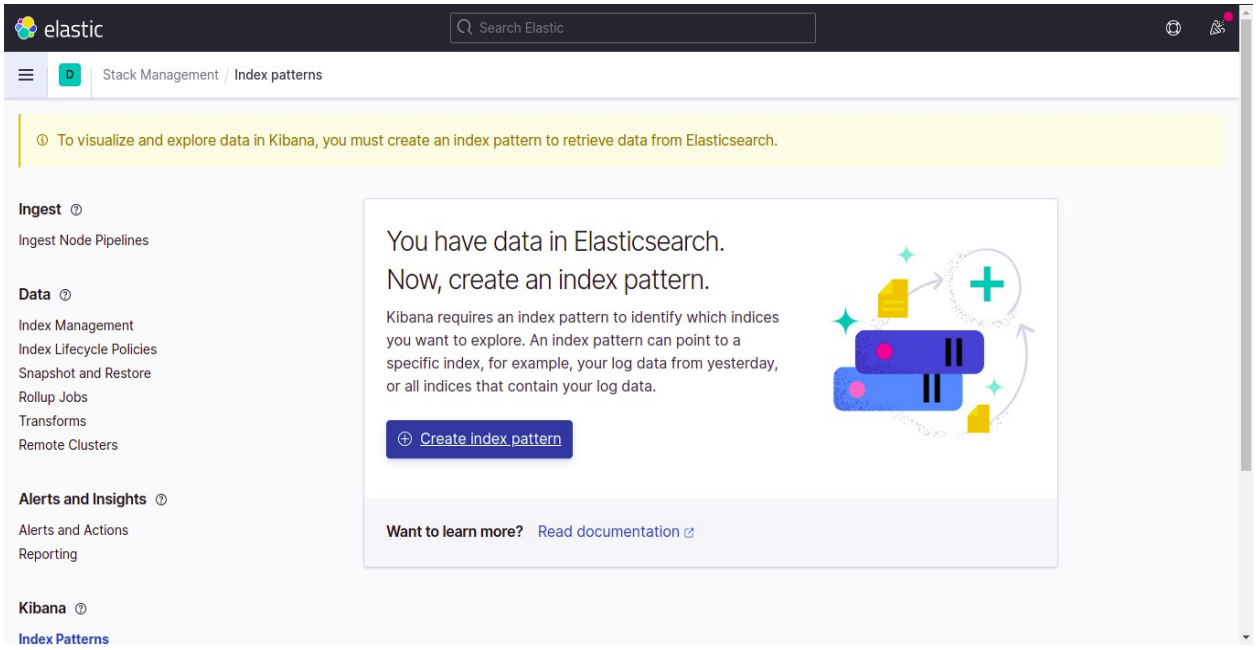
kita sekarang dapat memeriksa Kibana untuk memverifikasi bahwa data log dikumpulkan dengan benar dan dikirim ke Elasticsearch

Dengan perintah **kubectl port-forward**, kita masuk kembali ke **Kibana**

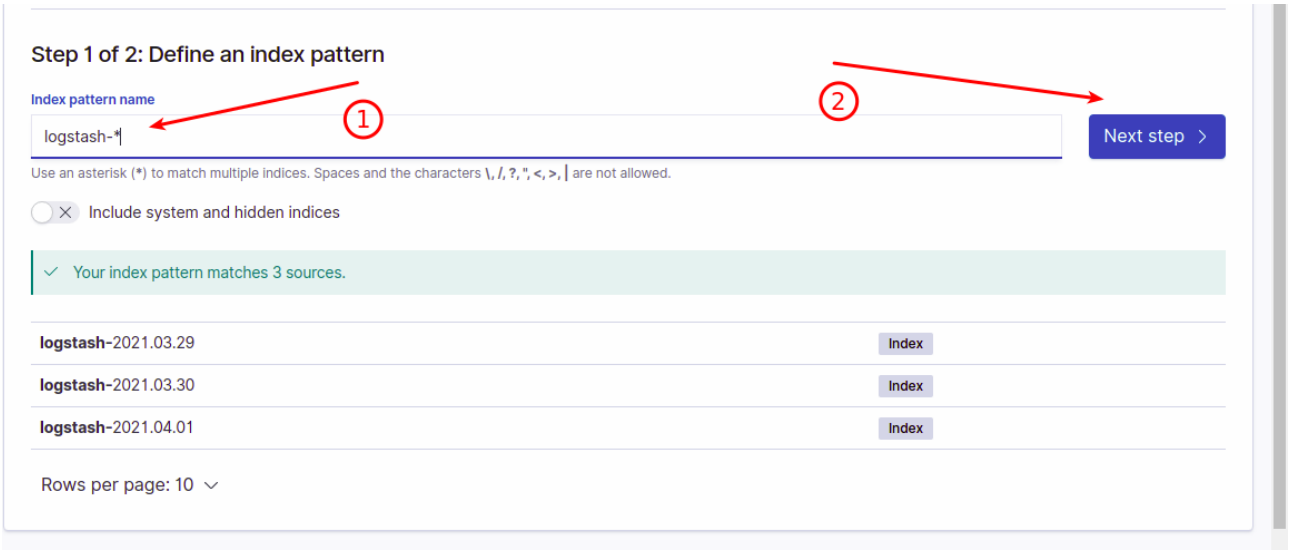
Klik Discover pada menu samping kiri. Caranya kita klik icon bergaris 3 diatas kiri, lalu klik **Discovery**.



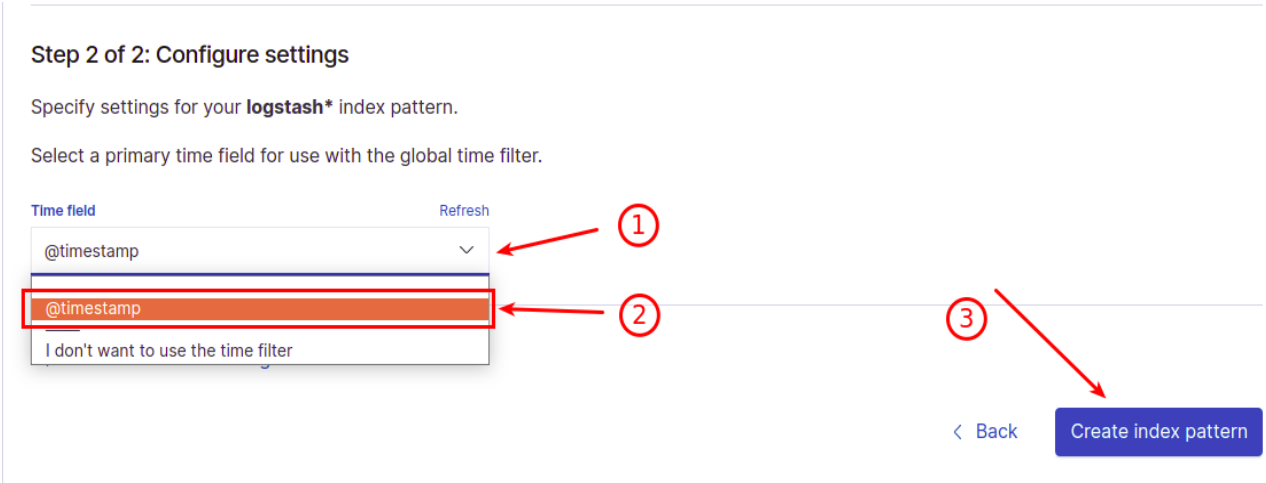
Kita akan melihat konfigurasi setup seperti gambar berikut. Klik **Create index pattern**



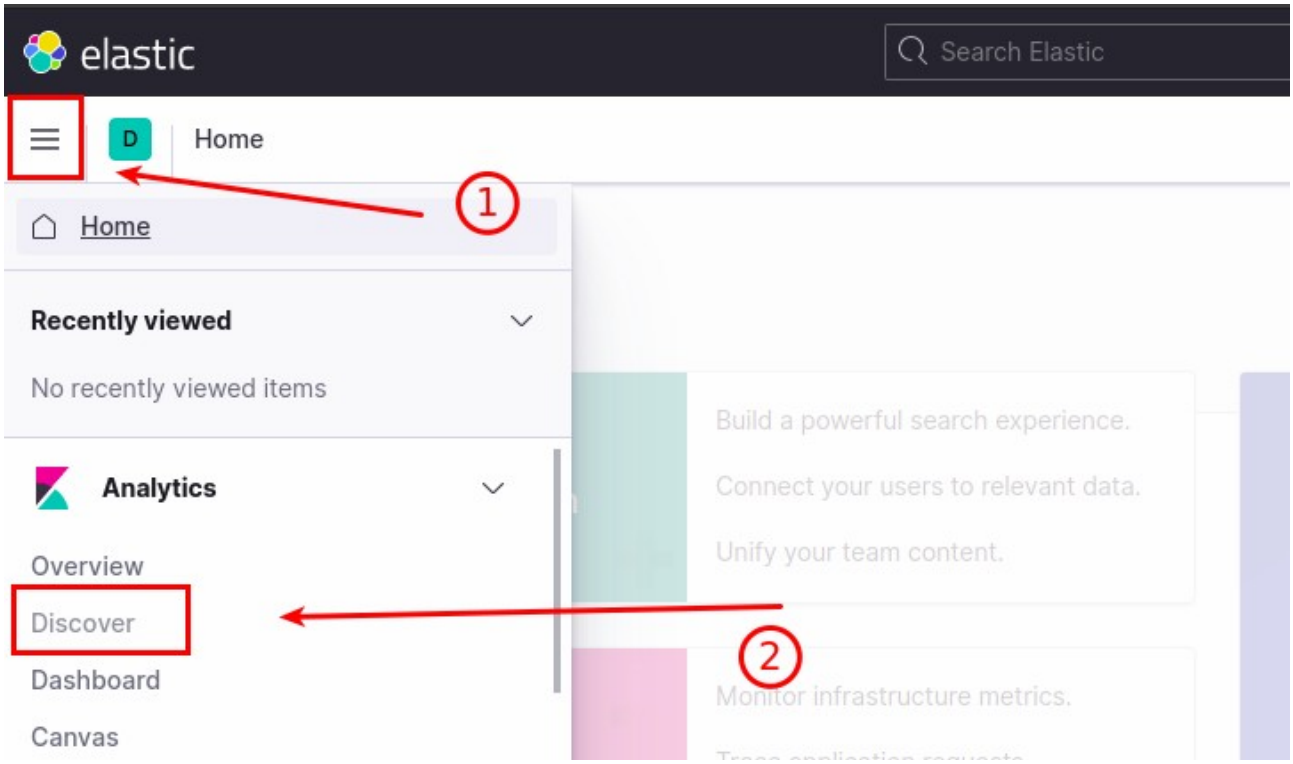
Kita masukan index pattern bernama **logstash-*** lalu kita klik next step



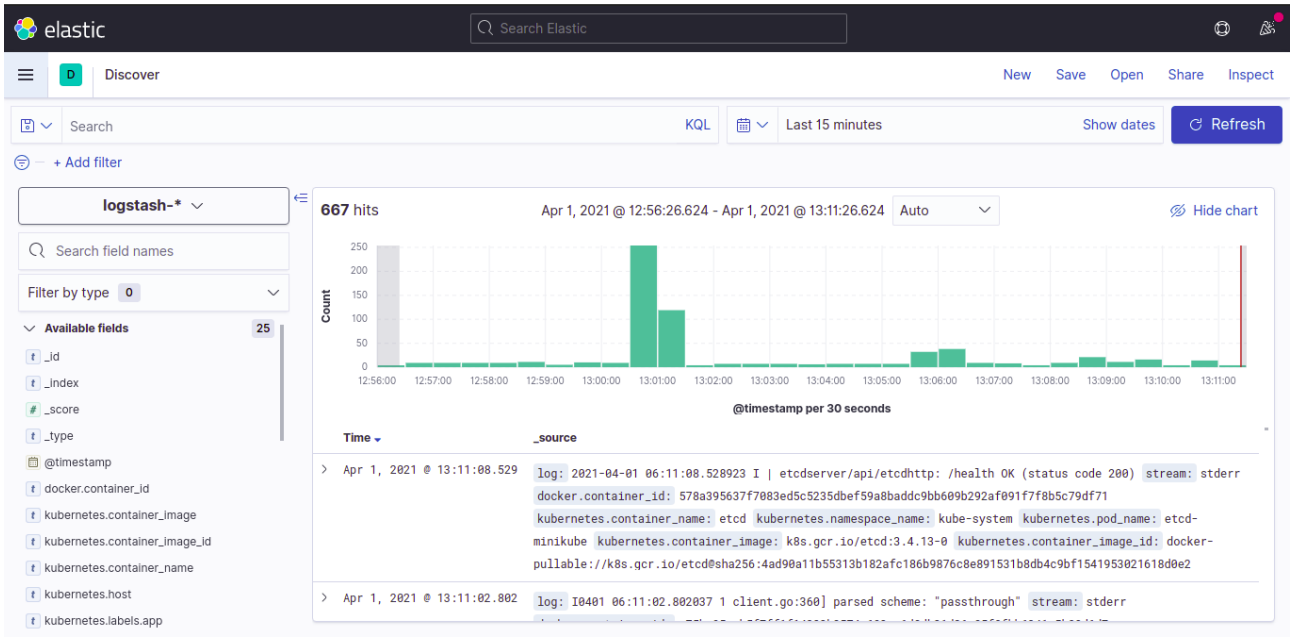
Langkah selanjutnya adalah menentukan field @timestamp pada index tersebut. Cara nya adalah mengeklik dropdown, lalu pilih @timestamp. Setelah itu, klik **Create Index Pattern**.



Index pattern sudah terbuat. Kita buka kembali menu discover yang ada pada menu bagian kiri. Caranya seperti yang sudah dilakukan sebelumnya.



Pada tampilan ini, kita bisa melihat log yang dikirimkan dari fluentd ke elasticsearch.



11.2.7. Testing Container Logging

Untuk mendemonstrasikan kasus penggunaan dasar Kibana menjelajahi log terbaru untuk node yang diberikan, kita akan menggunakan node penghitung minimal yang mencetak nomor berurutan ke stdout.

Buat file konfigurasi dengan nama counter.yaml dengan menggunakan perintah berikut

```
nano p-counter-pod.yaml
```

Masukan konfigurasi berikut

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: count
    image: busybox
    args: [/bin/sh, -c,
            'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']
```

script dapat dilihat di

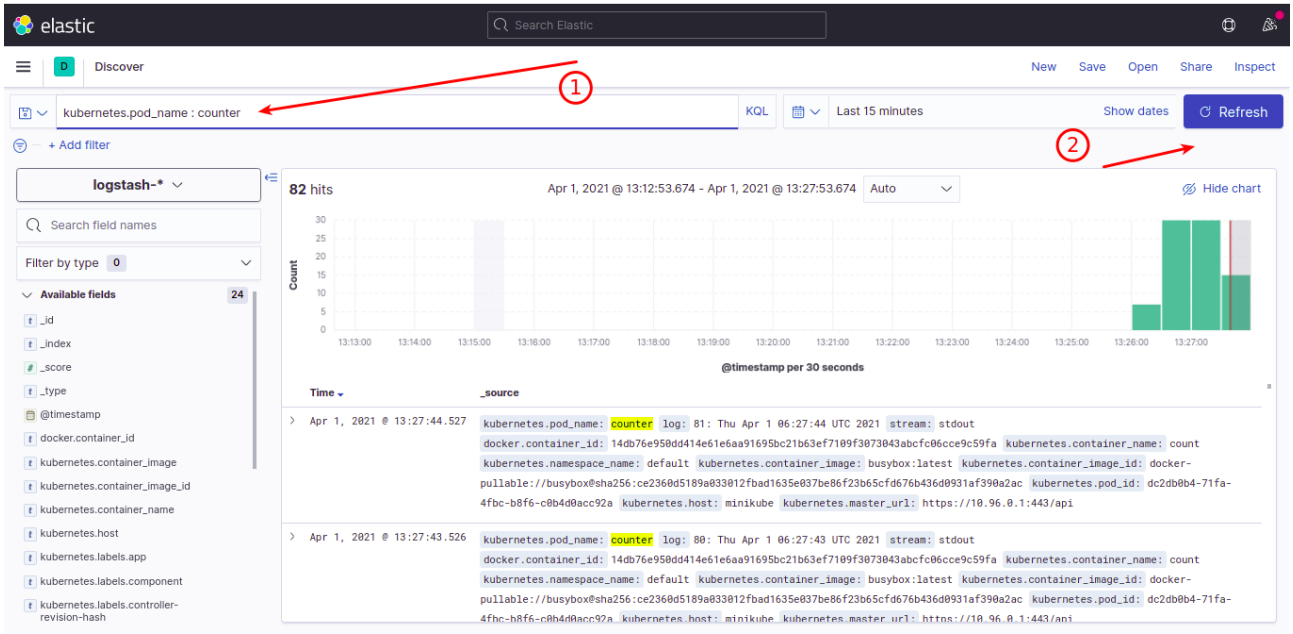
https://gist.github.com/sdcilsy/6f43372d9a1b4abe9ff2dcd976d21160

Save dan tutup text editor nano dengan menekan CTRL+X lalu tekan Y dan enter.

Jalan menggunakan perintah berikut

```
kubectl apply -f p-counter-pod.yaml
```

Dari halaman Discover, di bagian **search**, masukkan **kubernetes.pod_name: counter**. Lalu klik tombol refresh seperti pada gambar untuk memfilter data log.



11.3. Kesimpulan

- Dalam modul ini, kita telah menunjukkan cara mengatur dan mengkonfigurasi Elasticsearch, Fluentd, dan Kibana di cluster Kubernetes. .
- Sebelum menyebarkan logging stack ini ke cluster Kubernetes production kita, yang terbaik adalah menyesuaikan persyaratan dan batasan sumber daya seperti yang ditunjukkan dalam panduan ini. Arsitektur logging yang kita gunakan di sini terdiri dari 1 node Elasticsearch, satu node Kibana tunggal dan satu set node Fluentd diluncurkan sebagai DaemonSet.
- Kubernetes juga memungkinkan untuk arsitektur agen logging yang lebih kompleks yang mungkin lebih sesuai dengan kasus penggunaan