

Taller de ELK en local

Sebastian Lopez Garcia

A00377582

## Contenido

Tiempo Estimado del ejercicio .....	2
Prompt.....	2
Implementación: .....	5
Preparación del Entorno .....	5
Iniciamos el minikube:.....	5
Crear Namespace de logging.....	5
Carpeta de recursos para el taller .....	6
Despliegue de Elasticsearch .....	6
Configuración de manifiestos.....	7
Creación de manifiestos .....	8
Comprobación .....	8
Despliegue de Kibana .....	8
Configuración de manifiestos.....	8
Creación de manifiestos .....	9
Acceder a Kibana .....	9
Despliegue de Filebeat .....	10
Creación de Filebeat.....	13
Creación de app para prueba .....	13
Crear namespace.....	13
Configuración .....	14
Creación de la app.....	14
Comprobación .....	14
Configuración final .....	16
Conclusiones .....	17

## Tiempo Estimado del ejercicio

This week										Week total: 02:57:09
Today										Total: 02:57:09
Documentación	<a href="#">+ Project</a>		\$	12:24 - 12:46		00:22:27				
Pruebas	<a href="#">+ Project</a>		\$	11:55 - 12:23		00:28:35				
2 Implementación	<a href="#">+ Project</a>		\$	09:30 - 11:55		01:52:02				
Creación del prompt	<a href="#">+ Project</a>		\$	09:22 - 09:30		00:08:10				
Investigación	<a href="#">+ Project</a>		\$	09:16 - 09:21		00:05:55				

## Prompt

### El prompt usado inicialmente fue:

Actúa como un Ingeniero DevOps Senior con amplia experiencia en Kubernetes, el stack ELK (Elasticsearch, Logstash, Kibana, Filebeat) y la implementación de soluciones de monitoreo y logging en entornos de desarrollo. Soy un estudiante de Ingeniería Telemática cursando 'Plataformas 2', con conocimientos básicos de Kubernetes (entiendo pods, deployments, services, namespaces, volúmenes persistentes, etc) con la capacidad de comprender explicaciones técnicas detalladas.

Necesito tu ayuda para elaborar un taller con guía paso a paso para implementar un stack ELK de manera local en mi PC utilizando WSL Ubuntu y Minikube, partiendo de que ya tengo mi ambiente configurado correctamente y funcional. El objetivo es crear una solución de logging centralizada para monitorear microservicios en un entorno de desarrollo, simulando el caso de una startup.

### Contexto del Proyecto:

- **Objetivo Principal:** Implementar una solución de logging centralizada para microservicios.
- **Entorno de Kubernetes:** Minikube ejecutándose sobre WSL en una máquina local.
- **Nivel de Conocimiento:** Básico en Kubernetes, pero con formación en ingeniería, por lo que busco explicaciones claras de los conceptos y el "por qué" de cada paso, junto con la profundidad técnica necesaria.

### Requisitos Específicos del Stack ELK y Desafíos a Cubrir:

Por favor, estructura tu respuesta para guiarme a través de los siguientes puntos, proporcionando explicaciones conceptuales, comandos exactos (para kubectl, minikube, etc.), manifiestos YAML completos y funcionales, y consejos de troubleshooting:

**1. Preparación del Entorno en WSL y Minikube:**

- Mejores prácticas para configurar Minikube en WSL para este proyecto (drivers, recursos asignados como CPU/memoria, etc.).
- Creación de un namespace dedicado para el stack de logging (ej: logging-stack).

**2. Implementación de Elasticsearch:**

- Despliegue de al menos un nodo de Elasticsearch (StatefulSet es preferible).
- Configuración de PersistentVolume (PV) y PersistentVolumeClaim (PVC) para asegurar la persistencia de datos de Elasticsearch. Explica cómo Minikube maneja la persistencia local.
- Gestión eficiente de recursos (requests y limits de CPU/memoria).
- Exponer Elasticsearch internamente en el clúster.

**3. Implementación de Kibana:**

- Despliegue de Kibana.
- Configuración para conectarse al servicio de Elasticsearch desplegado anteriormente.
- Hacer Kibana accesible externamente desde el navegador de mi máquina host Windows (ej: usando minikube service o configurando un Ingress si es factible y didáctico en Minikube). Explica las opciones.

**4. Implementación de Filebeat:**

- Despliegue de Filebeat como un DaemonSet para asegurar que se ejecute en cada nodo (o el único nodo de Minikube) para recolectar logs de contenedores.
- Configuración de Filebeat para recolectar logs de todos los contenedores del clúster (o de namespaces específicos si es más práctico para el ejemplo).
- Configuración de Filebeat para enviar los logs directamente a Elasticsearch.
- **Importante:** Para probar Filebeat, por favor incluye un ejemplo sencillo de un microservicio (un simple pod con una aplicación que genere logs a stdout, como un Nginx o una app Python/Node.js básica) que podamos desplegar en un namespace diferente (ej: apps) para ver sus logs en Kibana.

**5. Desafíos Técnicos a Resolver (Guía y Ejemplos):**

- **Configuración Básica (ya cubierta en parte arriba):**

- Namespace dedicado.
- Volúmenes persistentes.
- Gestión de recursos.
- **Desafíos Avanzados (al menos conceptualmente o con ejemplos básicos):**
  - **Políticas de Retención de Logs:** ¿Cómo se abordarían con Elasticsearch (ej: Index Lifecycle Management - ILM)? Proporciona un ejemplo básico o una explicación de cómo configurarlo.
  - **Filtrado y Procesamiento de Logs:**
    - Si se usa solo Filebeat y Elasticsearch: ¿Cómo se puede hacer procesamiento básico con Ingest Pipelines en Elasticsearch?
    - **Opcional (Logstash):** Si decidiera incluir Logstash para procesamiento avanzado, ¿cómo se integraría entre Filebeat y Elasticsearch? Describe su despliegue y un ejemplo de pipeline de Logstash sencillo para parsear o enriquecer logs. Indica cuándo recomendarías añadir Logstash.
  - **Crear Dashboard de Monitoreo Básico en Kibana:** Pasos para crear una visualización simple y un dashboard en Kibana para ver los logs recolectados.

**Entregables que espero de tu respuesta (para yo poder generar los míos):**

- **Documentación del Proceso:** Una guía paso a paso clara y detallada.
- **Archivos de Configuración:** Ejemplos completos de manifiestos YAML para cada componente (Elasticsearch, Kibana, Filebeat, PV/PVC, y el microservicio de ejemplo).
- **Scripts de Instalación (Opcional, pero útil):** Si aplica, pequeños scripts bash para automatizar partes del despliegue.
- **Guía de Uso y Acceso:** Cómo acceder a Kibana y cómo verificar que los logs del microservicio de ejemplo están llegando.

**Consideraciones Adicionales (abordar brevemente si es posible):**

- **Seguridad:** Menciones básicas sobre cómo se podría empezar a securizar el stack (ej: usuarios en Elasticsearch, TLS).
- **Alertas:** ¿Cómo se podrían configurar alertas básicas basadas en logs?
- **Optimización:** Consejos generales para optimizar el rendimiento del stack en un entorno Minikube.

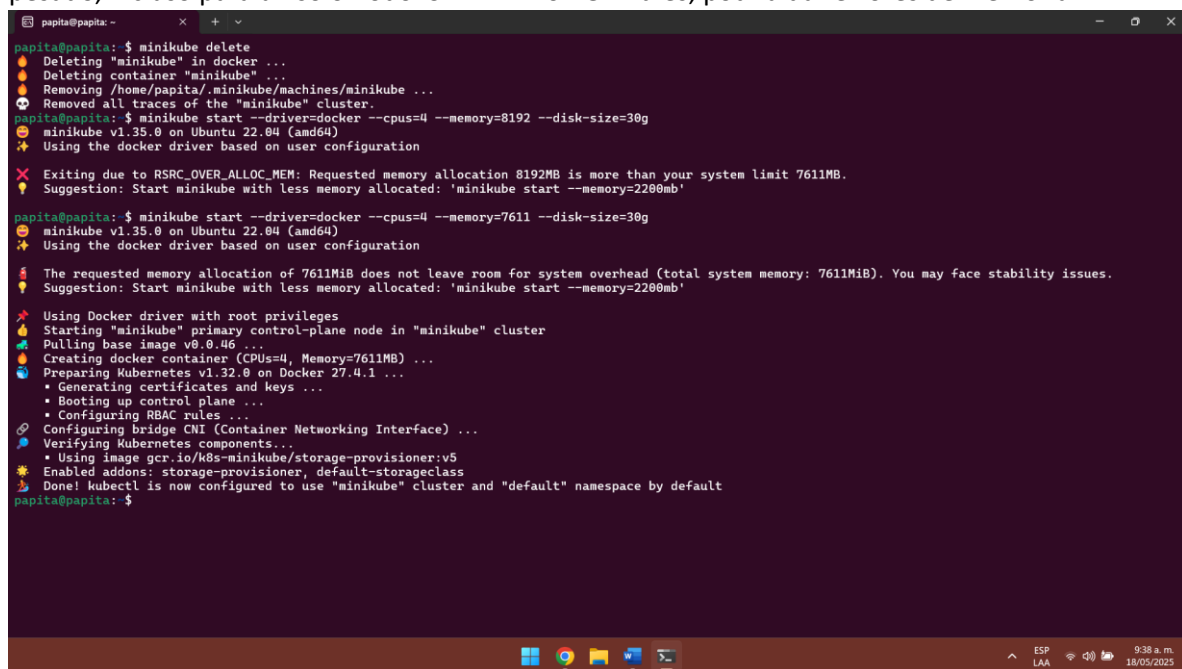
Por favor, asegúrate de que las explicaciones sean lo suficientemente detalladas para alguien que está aprendiendo pero que también sean técnicamente sólidas y reflejen las mejores prácticas. Explica el 'por qué' de las decisiones de configuración clave. ¡Muchas gracias por tu ayuda!

## Implementación:

### Preparación del Entorno

#### Iniciamos el minikube:

Vamos a iniciar el minikube con las siguientes características ya que el Elasticsearch es un sistema pesado, incluso para un solo nodo. Sin mínimo 4GB libres, podría dar errores de memoria.



```
papita@papita: ~$ minikube delete
Deleting "minikube" in docker ...
Deleting container "minikube" ...
Removing /home/papita/.minikube/machines/minikube ...
Removed all traces of the "minikube" cluster.
papita@papita: ~$ minikube start --driver=docker --cpus=4 --memory=8192 --disk-size=30g
minikube v1.35.0 on Ubuntu 22.04 (amd64)
Using the docker driver based on user configuration

Exiting due to RSRC_OVER_ALLOC_MEM: Requested memory allocation 8192MB is more than your system limit 7611MB.
Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

papita@papita: ~$ minikube start --driver=docker --cpus=4 --memory=7611 --disk-size=30g
minikube v1.35.0 on Ubuntu 22.04 (amd64)
Using the docker driver based on user configuration

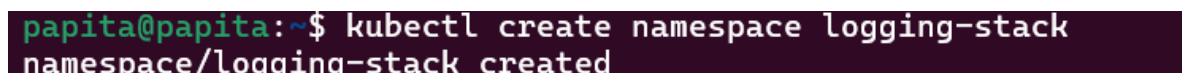
The requested memory allocation of 7611MiB does not leave room for system overhead (total system memory: 7611MiB). You may face stability issues.
Suggestion: Start minikube with less memory allocated: 'minikube start --memory=2200mb'

Using Docker driver with root privileges
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Creating docker container (CPUs=4, Memory=7611MB) ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  Generating certificates and keys ...
  Booting up control plane ...
  Configuring RBAC rules ...
  Configuring bridge CNI (Container Networking Interface) ...
  Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
papita@papita: ~$
```

```
minikube start --driver=Docker --cpus=4 --memory=7611 --disk-size=30g
```

(Se trató de usar la máxima memoria permitida por mi sistema)

#### Crear Namespace de logging



```
papita@papita: ~$ kubectl create namespace logging-stack
namespace/logging-stack created
```

```
papita@papita:~$ kubectl get namespace
NAME          STATUS   AGE
default       Active   2m36s
kube-node-lease  Active   2m36s
kube-public    Active   2m36s
kube-system    Active   2m36s
logging-stack  Active   45s
papita@papita:~$ |
```

## Carpeta de recursos para el taller

```
papita@papita:~$ ls
Uni  cvat  fabiana  go  kubectl  nginx-replicaset.yaml  package-lock.json  php-apache.yaml  terraform
papita@papita:~$ cd Uni/platII/
papita@papita:~/Uni/platII$ ls
ClaseConfigMap  helm  microservice-project  taller-az-DockerFile  taller-dockerNet
papita@papita:~/Uni/platII$ mkdir ELKtaller
papita@papita:~/Uni/platII$ cd ELKtaller/
papita@papita:~/Uni/platII/ELKtaller$
```

## Despliegue de Elasticsearch

Elasticsearch es una base de datos NoSQL orientada a documentos. Filebeat envía logs que Elasticsearch indexa y permite consultar rápidamente.

## Configuración de manifiestos

Vamos a crear nuestro manifiesto completo de Elasticsearch:

```
papita@papita:~/Uni/platII/ELKtaller/elasticsearch$ cat elasticsearch-statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch
  namespace: logging-stack
spec:
  serviceName: "elasticsearch"
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      containers:
        - name: elasticsearch
          image: docker.elastic.co/elasticsearch/elasticsearch:7.17.14
          ports:
            - containerPort: 9200
              name: http
          env:
            - name: discovery.type
              value: single-node
            - name: ES_JAVA_OPTS
              value: "-Xms512m -Xmx512m"
          volumeMounts:
            - name: data
              mountPath: /usr/share/elasticsearch/data
      resources:
        requests:
          memory: "1Gi"
          cpu: "500m"
        limits:
          memory: "2Gi"
          cpu: "1"
      volumeClaimTemplates:
        - metadata:
            name: data
          spec:
            accessModes: ["ReadWriteOnce"]
            resources:
              requests:
                storage: 5Gi
```

También crearemos el servicio:

```
papita@papita:~/Uni/platII/ELKtaller/elasticsearch$ cat elasticsearch-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: elasticsearch
  namespace: logging-stack
spec:
  selector:
    app: elasticsearch
  ports:
    - port: 9200
      targetPort: 9200
```

## Creación de manifiestos

Después de tener los manifiestos configurados como **.yaml**, los aplicamos:

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f elasticsearch-statefulset.yaml
kubectl apply -f elasticsearch-service.yaml
statefulset.apps/elasticsearch created
service/elasticsearch created
```

## Comprobación

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl get pvc -n logging-stack
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
data-elasticsearch-0	Bound	pvc-79f4f827-56a0-44bf-a6af-b8bce3d28574	5Gi	RWO	standard	<unset>	73s

## Despliegue de Kibana

Kibana es la interfaz web de Elasticsearch. Te permite visualizar los datos indexados.

## Configuración de manifiestos

```
papita@papita:~/Uni/platII/ELKtaller/kibana$ cat kibana-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kibana
  namespace: logging-stack
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kibana
  template:
    metadata:
      labels:
        app: kibana
    spec:
      containers:
        - name: kibana
          image: docker.elastic.co/kibana/kibana:7.17.14
          ports:
            - containerPort: 5601
          env:
            - name: ELASTICSEARCH_HOSTS
              value: "http://elasticsearch:9200"
```

Ahora configuramos el Service:

```
papita@papita:~/Uni/platII/ELKtaller/kibana$ cat kibana-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: kibana
  namespace: logging-stack
spec:
  type: NodePort
  ports:
    - port: 5601
      nodePort: 30001
  selector:
    app: kibana
```



## Creación de manifiestos

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f kibana-deployment.yaml
kubectl apply -f kibana-service.yaml
deployment.apps/kibana created
service/kibana created
```

## Acceder a Kibana

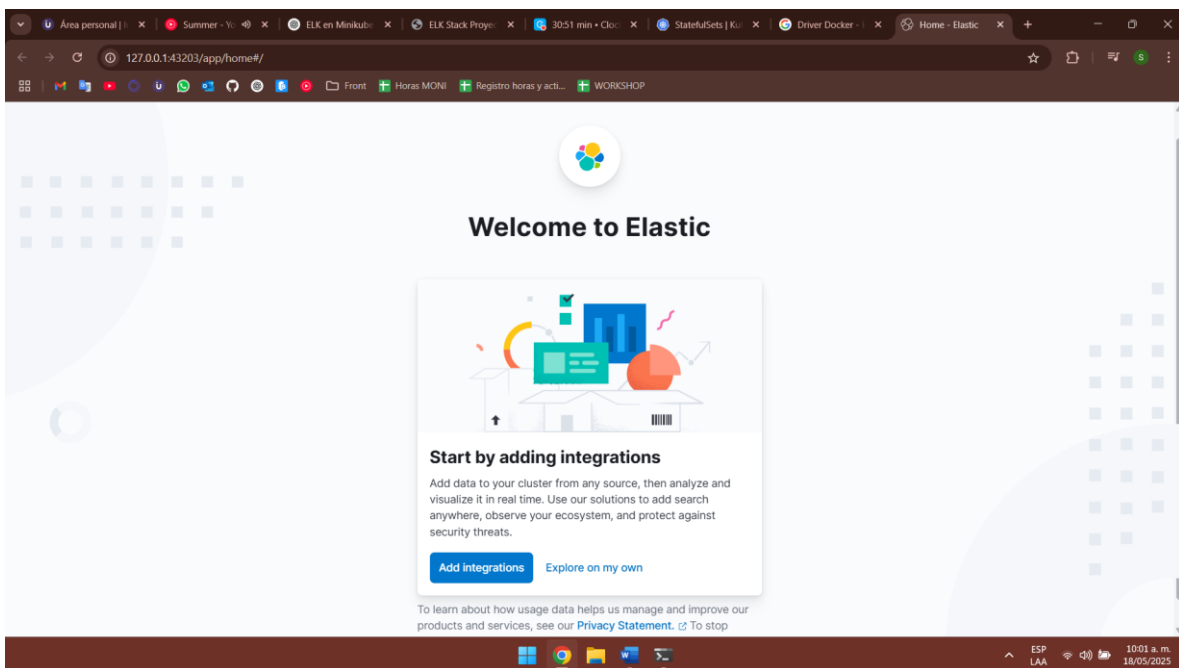
```
papita@papita:~/Uni/platII/ELKtaller$ minikube service kibana -n logging-stack
```

NAMESPACE	NAME	TARGET PORT	URL
logging-stack	kibana	5601	http://192.168.49.2:30001

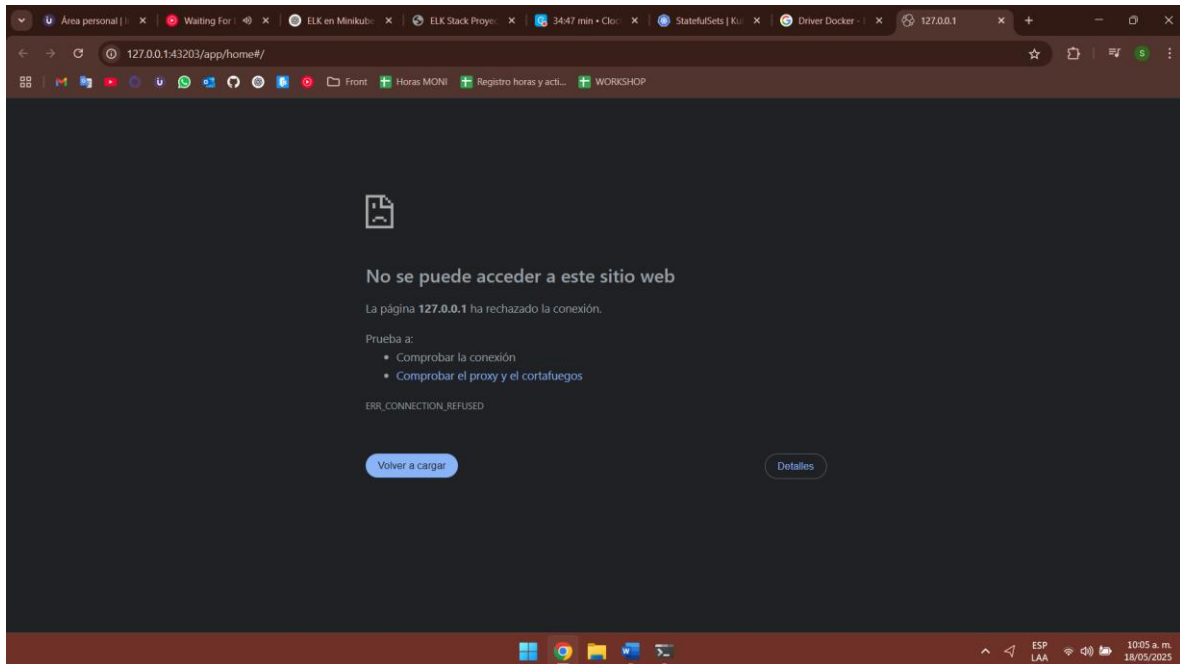
🚀 Starting tunnel for service kibana.

NAMESPACE	NAME	TARGET PORT	URL
logging-stack	kibana		http://127.0.0.1:43203

🌐 Opening service logging-stack/kibana in default browser...  
👉 http://127.0.0.1:43203  
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.



Al rato falló, posiblemente porque consumió los recursos que tenía al abrir dos instancias de Wsl mientras dejaba el tunnel abierto, así que primero acabaremos la configuración.



```
papita@papita: ~/Uni/platII/E
papita@papita: $ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

papita@papita: $ kubectl get deployments
No resources found in default namespace.
papita@papita: $ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP    29m
papita@papita: $ kubectl get pvc -n logging-stack
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
data-elasticsearch-0   Bound     pvc-79f4f827-56a0-44bf-a6af-b8bce3d28574   5Gi        RWO            standard      <unset>              17m
papita@papita: $ kubectl apply -f elasticsearch-statefulset.yaml
kubectl apply -f elasticsearch-service.yaml
error: the path "elasticsearch-statefulset.yaml" does not exist
error: the path "elasticsearch-service.yaml" does not exist
papita@papita: $ cd Uni/platII/ELKtaller/
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f elasticsearch-statefulset.yaml
kubectl apply -f elasticsearch-service.yaml
statefulset.apps/elasticsearch configured
service/elasticsearch unchanged
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f kibana-deployment.yaml
kubectl apply -f kibana-service.yaml
deployment.apps/kibana unchanged
service/kibana unchanged
papita@papita:~/Uni/platII/ELKtaller$ |
```

Comprobamos que efectivamente solo se cayó el tunnel, por lo que lo dejaremos sin subirlo mientras configuramos el Filebeat y el ejemplo de aplicación en este caso una básica con nginx.

## Despliegue de Filebeat

Filebeat recolecta logs desde /var/log/containers, los transforma en JSON y los envía a Elasticsearch.

## Configuración de Filebeat

```
papita@papita:~/Uni/platII/ELKtaller/filebeat$ cat filebeat-daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: filebeat
  namespace: logging-stack
spec:
  selector:
    matchLabels:
      app: filebeat
  template:
    metadata:
      labels:
        app: filebeat
    spec:
      serviceAccountName: filebeat
      containers:
        - name: filebeat
          image: docker.elastic.co/beats/filebeat:7.17.14
          args: ["-c", "/etc/filebeat.yml", "-e"]
          env:
            - name: NODE_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
          volumeMounts:
            - name: config
              mountPath: /etc/filebeat.yml
              subPath: filebeat.yml
            - name: varlog
              mountPath: /var/log
            - name: containers
              mountPath: /var/lib/docker/containers
              readOnly: true
      volumes:
        - name: config
          configMap:
            name: filebeat-config
        - name: varlog
          hostPath:
            path: /var/log
        - name: containers
          hostPath:
            path: /var/lib/docker/containers
```

```

papita@papita:~/Uni/platII/ELKtaller/filebeat$ cat filebeat-rbac.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: filebeat
rules:
- apiGroups: [""]
  resources:
    - pods
    - namespaces
    - nodes
  verbs: ["get", "watch", "list"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: filebeat
  namespace: logging-stack
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: filebeat
subjects:
- kind: ServiceAccount
  name: filebeat
  namespace: logging-stack
roleRef:
  kind: ClusterRole
  name: filebeat
  apiGroup: rbac.authorization.k8s.io

```

**ClusterRole:** Esta define los permisos de acceso que necesita Filebeat, por ejemplo:

- get, watch, list sobre:
  - pods: para ver los pods y sus metadatos (etiquetas, nombres, etc.)
  - namespaces: para saber a qué espacio pertenece un pod
  - nodes: porque Filebeat usa autodiscover y necesita saber en qué nodo está ejecutándose

Esto es clave para que Filebeat pueda hacer autodiscovery y añadir metadatos Kubernetes a los logs.

**ServiceAccount:** Esta crea una cuenta de servicio que usarán los pods de Filebeat. Es la "identidad" bajo la cual Filebeat opera en Kubernetes. Ya que Kubernetes no le da permisos a cualquier pod por defecto. Para que Filebeat tenga acceso al API server, necesita esta cuenta de servicio.

**ClusterRoleBinding:** Esta une el **ClusterRole** con la **ServiceAccount**, permitiendo que la cuenta filebeat tenga los permisos definidos para todo el clúster. Sin este binding, la ServiceAccount no puede hacer nada especial.

```
papita@papita:~/Uni/platII/ELKtaller/filebeat$ cat filebeat-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config
  namespace: logging-stack
data:
  filebeat.yml: |-
    filebeat.autodiscover:
      providers:
        - type: kubernetes
          node: ${NODE_NAME}
          hints.enabled: true
          hints.default_config:
            type: container
            paths:
              - /var/log/containers/*.log

    processors:
      - add_cloud_metadata: ~
      - add_host_metadata: ~
      - add_kubernetes_metadata:
          in_cluster: true

    output.elasticsearch:
      hosts: ["http://elasticsearch:9200"]
```

## Creación de Filebeat

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f filebeat-configmap.yaml
kubectl apply -f filebeat-daemonset.yaml
configmap/filebeat-config created
daemonset.apps/filebeat created
```

## Creación de app para prueba

### Crear namespace

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl create namespace apps
namespace/apps created
```

## Configuración

```
papita@papita:~/Uni/platII/ELKtaller/apps$ cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: apps
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
```

## Creación de la app

```
papita@papita:~/Uni/platII/ELKtaller$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx created
```

## Comprobación

kubectl get pods -n logging-stack

kubectl get deployments -n logging-stack

kubectl get daemonsets -n logging-stack

kubectl get services -n logging-stack

kubectl get configmaps -n logging-stack

```
papita@papita:~/Uni/plat11/ELKtaller$ kubectl get pods -n logging-stack
kubectl get deployments -n logging-stack
kubectl get daemonsets -n logging-stack
kubectl get services -n logging-stack
kubectl get configmaps -n logging-stack
```

NAME	READY	STATUS	RESTARTS	AGE
debug	1/1	Running	0	65m
elasticsearch-0	1/1	Running	0	119m
filebeat-vfqnc	1/1	Running	0	31m
kibana-c85c7c79d-64qhb	1/1	Running	0	111m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
kibana	1/1	1	1	111m

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
filebeat	1	1	1	1	1	<none>	97m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
elasticsearch	ClusterIP	10.99.143.32	<none>	9200/TCP	119m
kibana	NodePort	10.102.14.153	<none>	5601:30001/TCP	111m

NAME	DATA	AGE
filebeat-config	1	97m
kube-root-ca.crt	1	130m

minikube service nginx -n apps

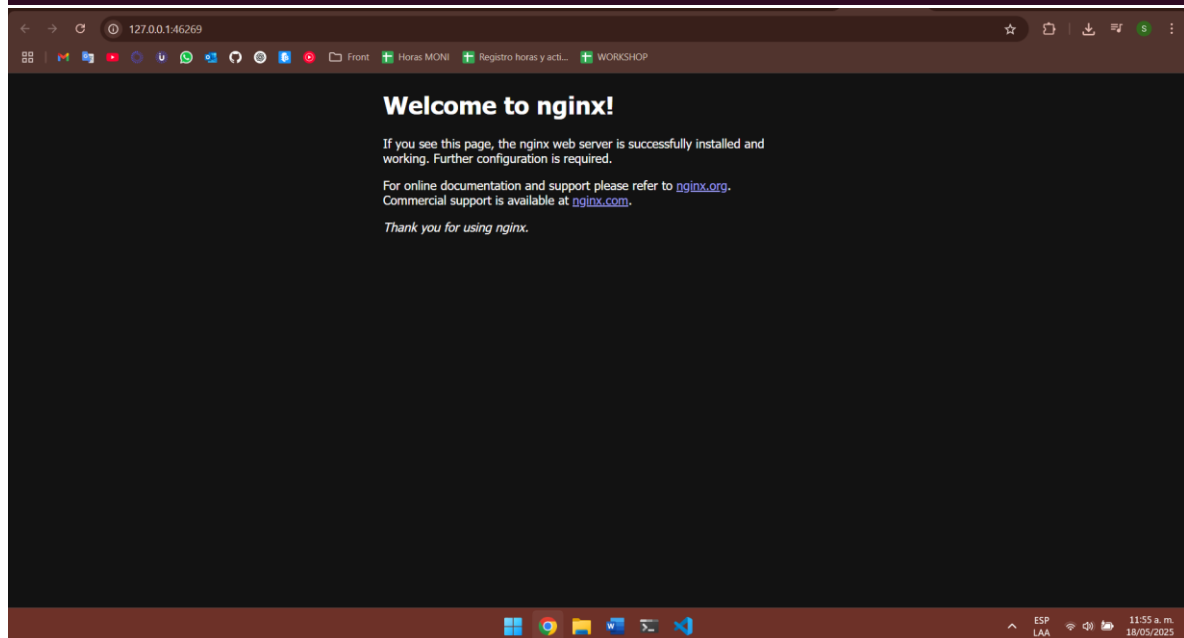
```
papita@papita:~/Uni/plat11/ELKtaller$ minikube service nginx -n apps
```

NAMESPACE	NAME	TARGET PORT	URL
apps	nginx	80	http://192.168.49.2:32181

🚀 Starting tunnel for service nginx.

NAMESPACE	NAME	TARGET PORT	URL
apps	nginx		http://127.0.0.1:46269

🌐 Opening service apps/nginx in default browser...  
 🖱️ http://127.0.0.1:46269  
 ! Because you are using a Docker driver on linux, the terminal needs to be open to run it.



## minikube service kibana -n logging-stack

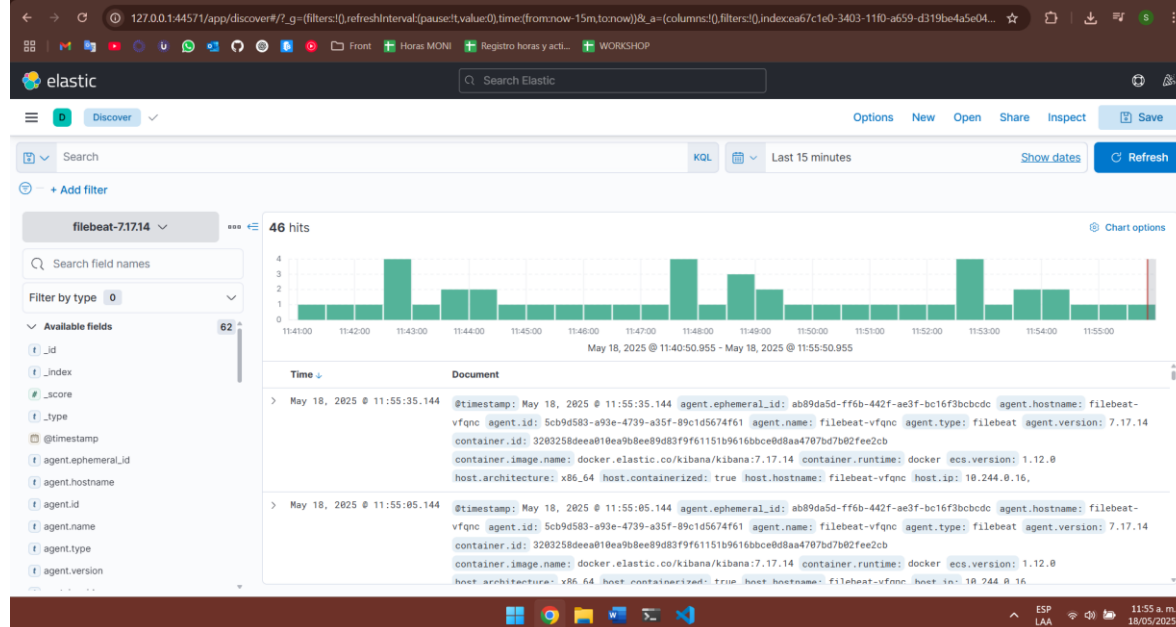
```
papita@papita:~/Uni/plat11/ELKtaller$ minikube service kibana -n logging-stack
```

NAMESPACE	NAME	TARGET PORT	URL
logging-stack	kibana	5601	http://192.168.49.2:30001

🚀 Starting tunnel for service kibana.

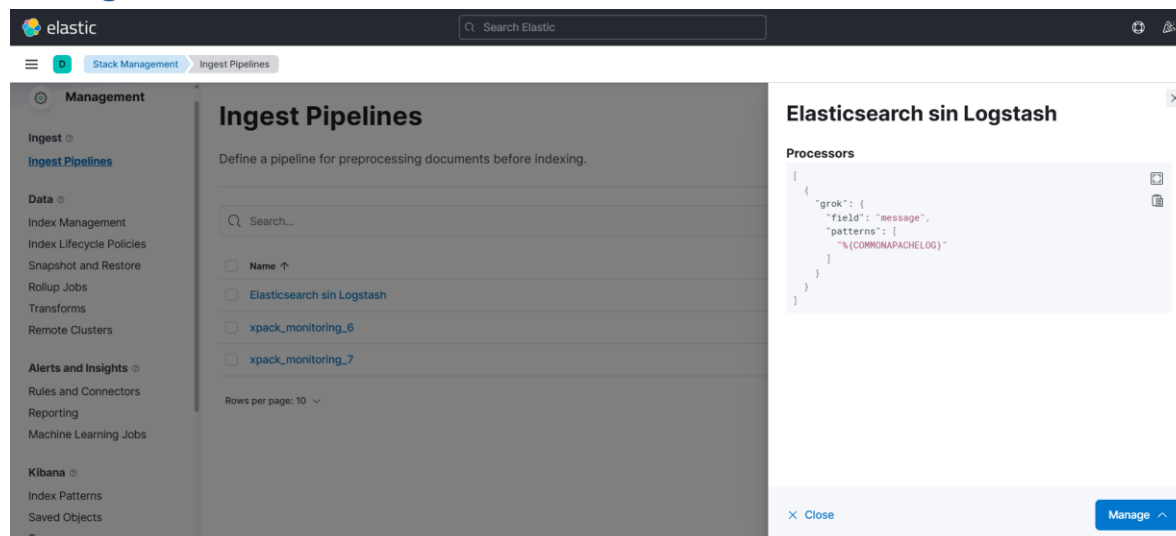
NAMESPACE	NAME	TARGET PORT	URL
logging-stack	kibana		http://127.0.0.1:44571

🐳 Opening service logging-stack/kibana in default browser...  
👉 http://127.0.0.1:44571  
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.



The screenshot shows the Elasticsearch Discover interface. The search bar contains 'filebeat-71714'. The results show 46 hits. A bar chart at the top shows the distribution of hits over time. The table below shows two documents, both from May 18, 2025, at 11:55:05.144. The documents contain log data from filebeat, including agent information, container details, and host architecture.

## Configuración final



The screenshot shows the Elasticsearch Stack Management page. The 'Ingest Pipelines' section is active. The page title is 'Ingest Pipelines'. Below the title, it says 'Define a pipeline for preprocessing documents before indexing.' There is a search bar and a list of pipelines. The list includes 'Elasticsearch sin Logstash', 'xpack\_monitoring\_6', and 'xpack\_monitoring\_7'. The 'Elasticsearch sin Logstash' pipeline is selected. The right sidebar shows the configuration for this pipeline, including the 'Processors' section. The configuration is as follows:

```
{
  "grok": {
    "field": "message",
    "patterns": [
      "%{COMMONAPACHELOG}"
    ]
  }
}
```

At the bottom of the sidebar, there are 'Close' and 'Manage' buttons.



## Conclusiones

### ▼ ELKTALLER

#### ▼ apps

! nginx-deployment.yaml

#### ▼ elasticsearch

! elasticsearch-service.yaml

! elasticsearch-statefulset.y...

#### ▼ filebeat

! filebeat-configmap.yaml

! filebeat-daemonset.yaml

! filebeat-rbac.yaml

! filebeat-serviceaccount.ya...

#### ▼ kibana

! kibana-deployment.yaml

! kibana-service.yaml

📄 .gitignore

\$ deploy.sh

☰ deploy.sh Zone.Identifier

Terminamos con el siguiente repositorio. Se agregó un .sh para optimizar, contiene lo siguiente:

```
$ deploy.sh
1  #!/bin/bash
2
3  kubectl create namespace logging-stack
4  kubectl create namespace apps
5
6  echo "Aplicando Elasticsearch..."
7  kubectl apply -f elasticsearch/elasticsearch-statefulset.yaml -n logging-stack
8  kubectl apply -f elasticsearch/elasticsearch-service.yaml -n logging-stack
9
10 echo "Aplicando Kibana..."
11 kubectl apply -f kibana/kibana-deployment.yaml -n logging-stack
12 kubectl apply -f kibana/kibana-service.yaml -n logging-stack
13
14 echo "Aplicando Filebeat..."
15 kubectl apply -f filebeat/filebeat-configmap.yaml -n logging-stack
16 kubectl apply -f filebeat/filebeat-rbac.yaml -n logging-stack
17 kubectl apply -f filebeat/filebeat-daemonset.yaml -n logging-stack
18
19 echo "Desplegando microservicio de ejemplo (nginx)..."
20 kubectl apply -f apps/nginx-deployment.yaml -n apps
21
22 echo "✅ Stack ELK desplegado. Accede a Kibana vía: minikube service kibana -n logging-stack"
23
```

Se inicia minikube:

```
minikube start --driver=Docker --cpus=4 --memory=7611 --disk-size=30g
```

Se ejecuta con el comando dentro del repositorio:

```
bash deploy.sh
```

```
papita@papita:~/Uni/plat11/ELKtaller$ bash deploy.sh
namespace/logging-stack created
namespace/apps created
Aplicando Elasticsearch...
statefulset.apps/elasticsearch created
service/elasticsearch created
Aplicando Kibana...
deployment.apps/kibana created
service/kibana created
Aplicando Filebeat...
configmap/filebeat-config created
clusterrole.rbac.authorization.k8s.io/filebeat created
serviceaccount/filebeat created
clusterrolebinding.rbac.authorization.k8s.io/filebeat created
error: the path "filebeat/filebeat-serviceaccount.yaml" does not exist
daemonset.apps/filebeat created
Desplegando microservicio de ejemplo (nginx)...
deployment.apps/nginx created
✅ Stack ELK desplegado. Accede a Kibana vía: minikube service kibana -n logging-stack
```

Lo mas probable es que no funcione aun, es necesario esperar y ejecutar las verificaciones y hasta que todo se encuentre en **running**, ahí si acceder.

Luego se verifica con:

```
kubectl get pods -n logging-stack
```

```
kubectl get deployments -n logging-stack
```

```
kubectl get daemonsets -n logging-stack
```

```
kubectl get services -n logging-stack
```

```
kubectl get configmaps -n logging-stack
```

```
papita@papita:~$ kubectl get pods -n logging-stack
kubectl get deployments -n logging-stack
kubectl get daemonsets -n logging-stack
kubectl get services -n logging-stack
kubectl get configmaps -n logging-stack
NAME                READY   STATUS    RESTARTS   AGE
elasticsearch-0      1/1     Running   0           2m27s
filebeat-vz75d       1/1     Running   0           2m24s
kibana-c85c7c79d-5slcn 1/1     Running   0           2m26s
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
kibana  1/1     1            1           2m26s
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
filebeat 1         1         1       1            1           <none>         2m24s
NAME    TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
elasticsearch  ClusterIP   10.100.52.3   <none>        9200/TCP   2m28s
kibana      NodePort    10.106.177.248 <none>        5601:30001/TCP 2m27s
NAME    DATA   AGE
filebeat-config  1       2m26s
kube-root-ca.crt 1       2m29s
```

La app seria con:

```
kubectl get pods -n apps
```

```
kubectl get deployments -n apps
```

```
kubectl get services -n apps
```

```
kubectl get configmaps -n apps
```

```
papita@papita:~$ kubectl get pods -n apps
kubectl get deployments -n apps
kubectl get services -n apps
kubectl get configmaps -n apps
NAME                READY   STATUS    RESTARTS   AGE
nginx-86c57bc6b8-hb552 1/1     Running   0           3m53s
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx  1/1     1            1           3m54s
No resources found in apps namespace.
NAME    DATA   AGE
kube-root-ca.crt 1       3m58s
```

En una consola hacemos:

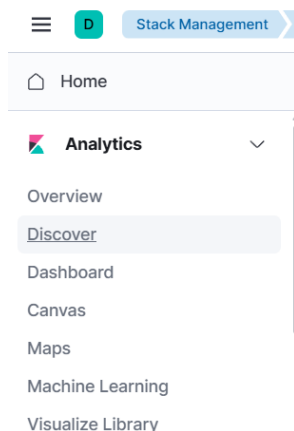
```
kubectl expose deployment nginx --port=80 --target-port=80 --
type=NodePort -n apps
```

```
minikube service nginx -n apps
```

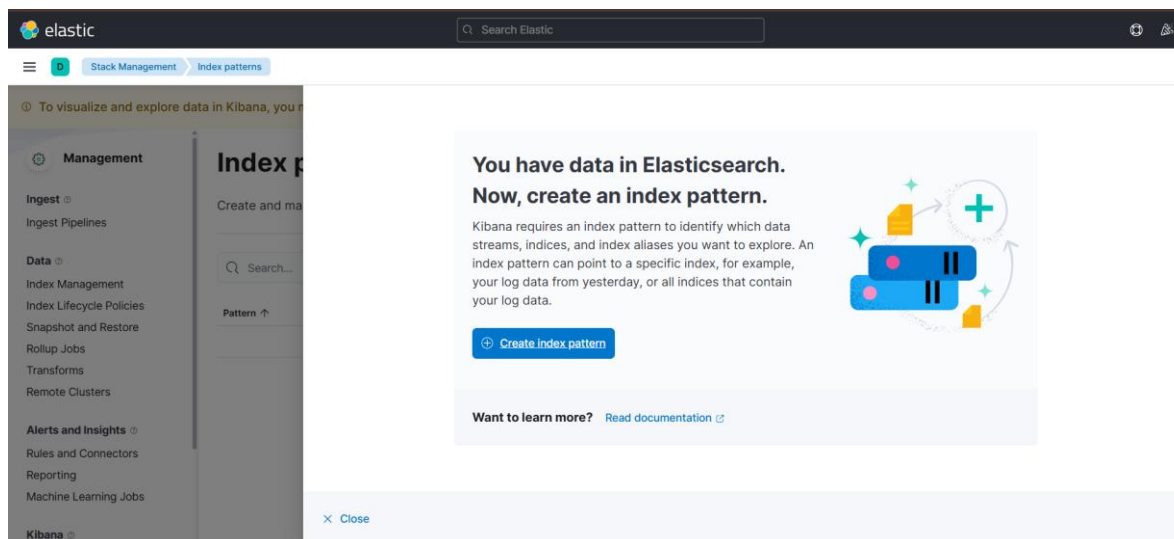
Y en otra consola hacemos:

```
minikube service kibana -n logging-stack
```

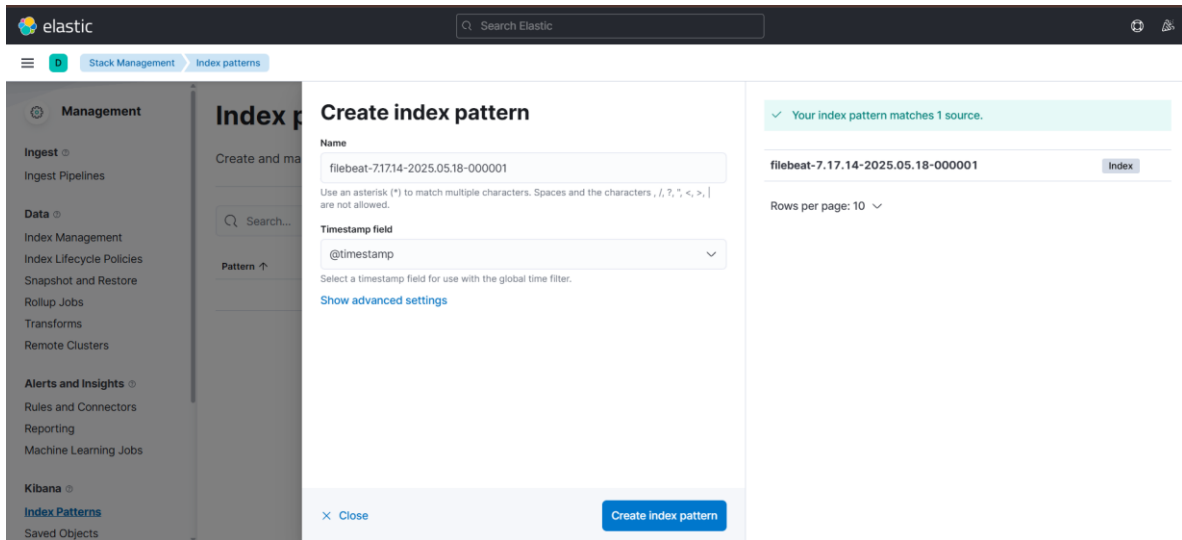
Dentro de:



Debería de salir algo parecido a:



Oprimimos **Create index pattern**:



Completamos como se (dependiendo de lo que te salga, puede cambiar el filebeat Name).

Y le damos a **Create index pattern**.

elastic

Search Elastic

Stack Management Index patterns filebeat-7.17.14-2025.05.18-000001

### filebeat-7.17.14-2025.05.18-000001

Time field: @timestamp

View and edit fields in filebeat-7.17.14-2025.05.18-000001. Field attributes, such as type and searchability, are based on [field mappings](#) in Elasticsearch.

Fields (6560) Scripted fields (0) Field filters (0)

Search

All field types Add field

Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		•	•	
_id	_id		•	•	
_index	_index		•	•	
_score					
_source	_source				
_type	_type		•	•	
activemq.caller	keyword		•	•	

elastic

Search Elastic

Dashboard Create

Inspect Download as CSV Cancel Save to library Save and return

Home

Analytics

Overview Discover Dashboard Canvas Maps Machine Learning Visualize Library

Enterprise Search

Overview App Search Workplace Search

Observability

Add integrations

Bar vertical stacked

Drop some fields here to start

Lens is a new tool for creating visualization

Make requests and give feedback

Horizontal axis

Add or drag-and-drop a field

Vertical axis

Add or drag-and-drop a field

Break down by

Add or drag-and-drop a field

Add layer

Ya configurariamos nuestro Dashboard por ejemplo, como queramos y dependiendo de nuestra app.