Duración: 24 horas

ELASTICSEARCH ADM.

BIENVENIDA

Organización

Temario

Objetivos

Presentaciones

Iván Osuna Ayuste

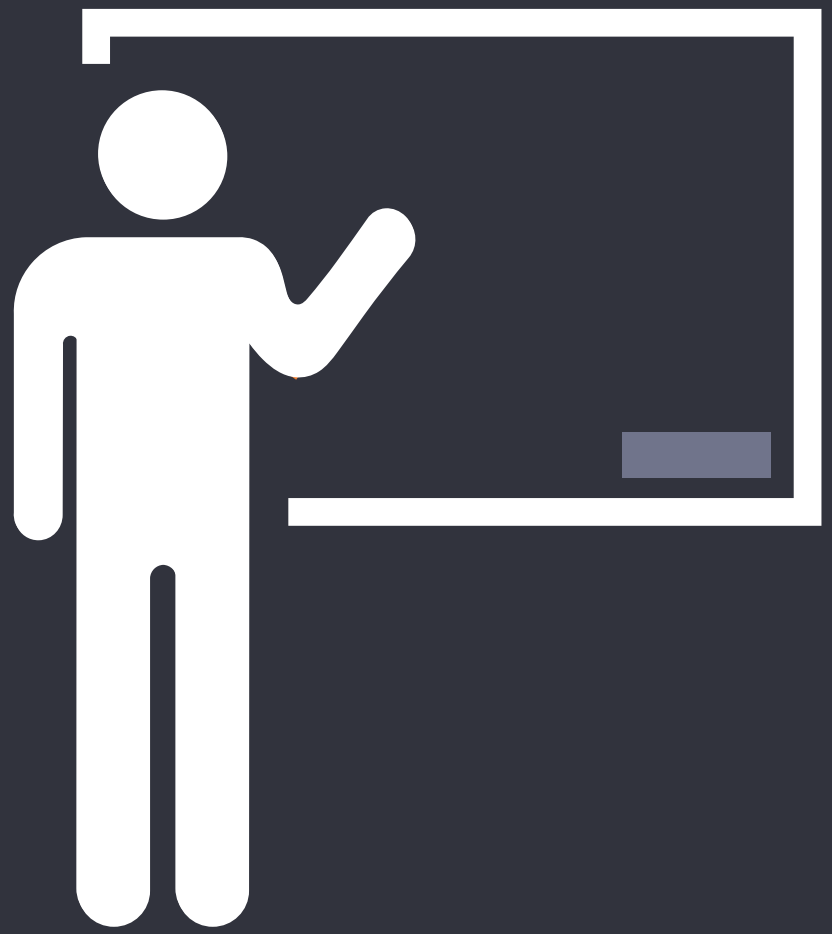
FORMADOR

Conozco cosillas de desarrollo, data analytics y BigData

Desarrollando desde los 8 años. Lenguajes de programación: C, C++, C#, Java, Python, ...

Diseño de arquitecturas y procesos Cloud y BigData: Hadoop, Spark, Hive, Casandra, Kafka, Flume, ...

Devops: Sistemas CI/CD, docker, kuberetes, openshift, openstack, ansible



ORGANIZACION

Metodología formativa

- 50% del tiempo para conceptos teóricos
- 50% del tiempo para prácticas

Material y actividades extra

- Plataforma de aprendizaje
- Grabación de las clases

Desarrollo de las clases

- 5 días de 5 horas con 2 descansos
- Resolución de preguntas

Financiado a través de FUNDAE

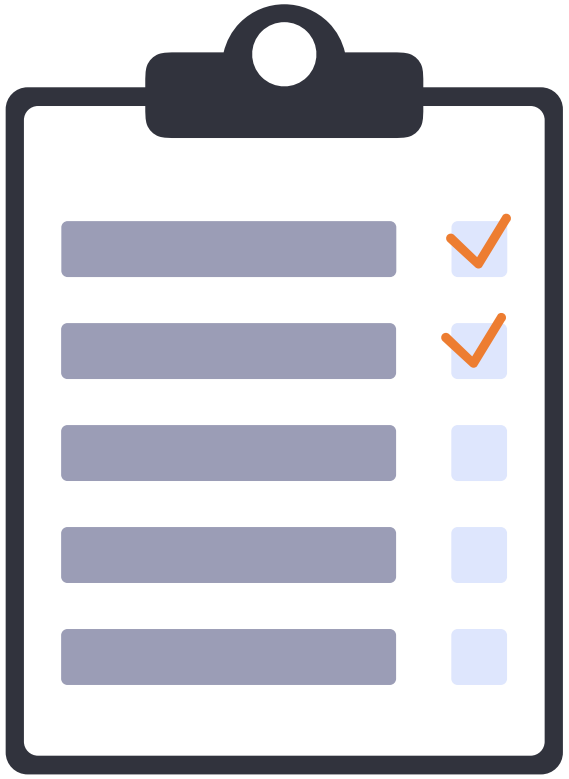
- Test de evaluación final

OBJETIVOS

- Conocer las características de ElasticSearch
- Conocer la arquitectura y los conceptos básicos que se manejan en ElasticSearch
- Conocer las herramientas del ecosistema ElasticSearch
- Aprender a realizar una buena configuración de un cluster ElasticSearch
- Aprender a realizar una buena monitorización de un cluster ElasticSearch
- Aprender a identificar y resolver problemas en un cluster ElasticSearch

NO SON OBJETIVOS

- Conocer en detalle cómo utilizar ElasticSearch como backend de una aplicación
- Conocer el funcionamiento de Kibana relacionado con el análisis y explotación de datos



01

Elastic Search

ElasticSearch, Lucene, Indices Invertidos, Índices invertidos, ...

02

Términos

node, cluster, index,document ...

03

Arquitectura y despliegue

Arquitectura del cluster, tipos de nodos, …

04

Configuración avanzada

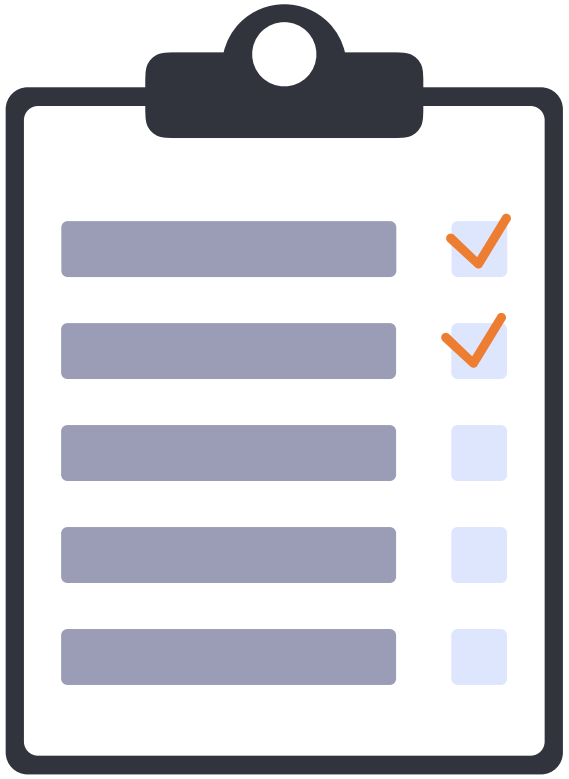
Parámetros importantes, seguridad, configuraciones para entornos de producción, routing…

05

Mantenimiento de nodos e indices

Optimización de nodos, índices y shards, alternativas de configuración

TEMARIO



06

Monitorización y resolución de problemas

Configurar la monitorización, principales problemáticas, optimización de nodos, índices y shards…

TEMARIO

ELASTIC SEARCH

ElasticSearch, Componentes, Características, Arquitectura

ELASTICSEARCH

Es un motor de búsqueda con las siguientes características:

- Permite búsquedas de texto completo (full text)
- Distribuido
- Escalable
- Tenencia múltiple
- Interfaz web RESTful
- Gestiona documentos JSON.

ELASTICSEARCH

Elasticsearch está desarrollado en Java y es un sistema OpenSource

- bajo las condiciones de la licencia Apache.

Está basado en Lucene.

Primera versión en 2010.

LUCENE

Apache Lucene es un motor de búsqueda, con las siguientes características:

- Permite búsquedas de texto completo (full text)
- Escalable
- Alto rendimiento

Lucene está desarrollado en Java y es un sistema OpenSource

- bajo las condiciones de la licencia Apache.

Basado en índices invertidos.

INDICES INVERTIDOS

Índice Directo

1	que es esto
2	esto es un texto
3	este texto es otro texto

Índice Invertido

que	1
es	1, 2, 3
esto	1, 2
un	2
texto	2, 3
este	3
otro	3

que	(1,1)
es	(1,2) (2,2) (3,3)
esto	(1,3) (2,1)
un	(2,3)
texto	(2,4) (3,2) (3,5)
este	(3,1)
otro	(3,4)

ÍNDICES INVERTIDOS

Ventajas:

- Velocidad de búsqueda
- Facilidad para aplicar algoritmos de relevancia y analizadores de texto

Inconvenientes:

- Velocidad de indexación

LUCENE VS ELASTICSEARCH

Lucene

- Generación de índices invertidos
- Búsquedas sobre índices invertidos
- Analizadores de texto
- Algoritmos de relevancia
- Resaltado de coincidencias
- Corrector ortográfico

ElasticSearch

- Interfaz REST
- Distribución
- Alta disponibilidad
- Tenencia múltiple

TÉRMINOS

Nodo, cluster, índice, shard, ...

NODE - NODO

Un nodo es una instancia en ejecución de Elasticsearch que pertenece a un clúster.

Se pueden iniciar varios nodos en un solo servidor para fines de prueba, pero generalmente suele haber un nodo por servidor.

Al inicio, un nodo usará unicast para descubrir un clúster existente con el mismo nombre e intentará unirse a ese clúster.

CLUSTER - CLÚSTER

Un clúster consta de uno o más nodos que comparten el mismo nombre de clúster.

Cada clúster tiene un único nodo maestro que el clúster elige automáticamente y que puede reemplazarse si falla el nodo maestro actual.

INDEX - ÍNDICE

Un índice es como una tabla en una base de datos relacional.

En él se almacenan documentos.

Un índice es un espacio de nombres lógico que se asigna a:

- uno o más fragmentos (shards) primarios
- cero o más fragmentos de réplica

Lleva asociado un mapping, que contiene un tipo de documento (que a su vez contiene los campos que se almacenarán en el índice).

INDEX ALIAS - ALIAS DE ÍNDICE

Un alias de índice es un nombre secundario utilizado para referirse a uno o más índices existentes.

La mayoría de las API de Elasticsearch aceptan un alias de índice en lugar de un nombre de índice.

DOCUMENT - DOCUMENTO

Documento JSON que se almacena en Elasticsearch.

Es como una fila en una tabla en una base de datos relacional.

Cada documento se almacena en un índice y tiene un tipo y un id.

Un documento es un objeto JSON que contiene cero o más campos (pares clave-valor).

El documento JSON original que se indexa se almacenará en el campo _source, y es devuelto de forma predeterminada al obtener o buscar un documento.

DOCUMENT TYPE - TIPO DE DOCUMENTO

Representa un tipo de documento, por ejemplo:

- Email
- Usuario
- Tweet
- Entrada de un log

Los tipos están en desuso y están en proceso de eliminación.

SOURCE FIELD - CAMPO FUENTE

De manera predeterminada, el documento JSON que indexa se almacenará en el campo _source y será devuelto por todas las solicitudes de búsqueda.

Esto permite acceder al objeto original directamente desde los resultados de búsqueda, en lugar de requerir un segundo paso para recuperar el documento a partir del Id.

ID - IDENTIFICADOR

Identifica un documento.

El id un documento debe ser único.

Si no se proporciona ningún id, se genera uno automáticamente.

FIELD - CAMPO

Un documento contiene una lista de campos o pares clave-valor.

Un campo es similar a una columna en una tabla en una base de datos relacional.

El valor puede ser:

- un valor simple (escalar) (por ejemplo, una cadena, un entero, una fecha) - una estructura anidada como una matriz o un objeto.

FIELD - CAMPO

El mapeo define para cada campo su tipo (que no debe confundirse con el tipo de documento), que indica el tipo de datos que se pueden almacenar en ese campo, por ejemplo integer, string, object.

El mapeo también permite definir cómo se debe analizar el valor de un campo.

TEXT FIELD - CAMPO DE TEXTO

Campo de texto no estructurado ordinario, como este párrafo.

Los campos de texto deben analizarse en el momento de la indexación para poder hacer búsquedas sobre el texto.

Por defecto, el texto se analiza en términos, que es lo que realmente se almacena en el índice.

Las palabras clave en las consultas full text deben analizarse en el momento de la búsqueda para producir (y buscar) los mismos términos que se generaron en el momento la indexación.

MAPPING - MAPEO

Un mapeo es como una definición de esquema en una base de datos relacional.

Cada índice tiene un mapeo, que define un tipo, más una serie de configuraciones de todo el índice.

Un mapeo se puede definir explícitamente o se generará automáticamente cuando se indexe un documento.

SHARD - FRAGMENTO

Un fragmento es una instancia de Lucene.

Es un "trabajador" de bajo nivel que Elasticsearch administra automáticamente.

Un índice es un espacio de nombres lógico que apunta a fragmentos primarios y de réplica.

Aparte de definir el número de fragmentos primarios y réplica que debe tener un índice, nunca necesita referirse a los fragmentos directamente.

SHARD - FRAGMENTO

El código debe tratar solo con los índices.

Elasticsearch distribuye fragmentos entre todos los nodos del clúster y puede mover fragmentos automáticamente de un nodo a otro en caso de falla del nodo o la adición de nuevos nodos.

PRIMARY SHARD

Cada documento se almacena en un único fragmento primario.

Cuando indexa un documento, se indexa primero en el fragmento primario, luego en todas las réplicas del fragmento primario.

Por defecto, un índice tiene un único fragmento primario.

PRIMARY SHARD

Pueden especificarse más fragmentos primarios para escalar el número de documentos que su índice puede manejar.

No pueden cambiarse el número de fragmentos primarios en un índice, una vez que se crea el índice. Sin embargo, un índice se puede convertir en un nuevo índice con más fragmentos utilizando la API Split.

REPLICA SHARD

Cada fragmento primario puede tener cero o más réplicas. Una réplica es una copia del fragmento primario y tiene dos propósitos:

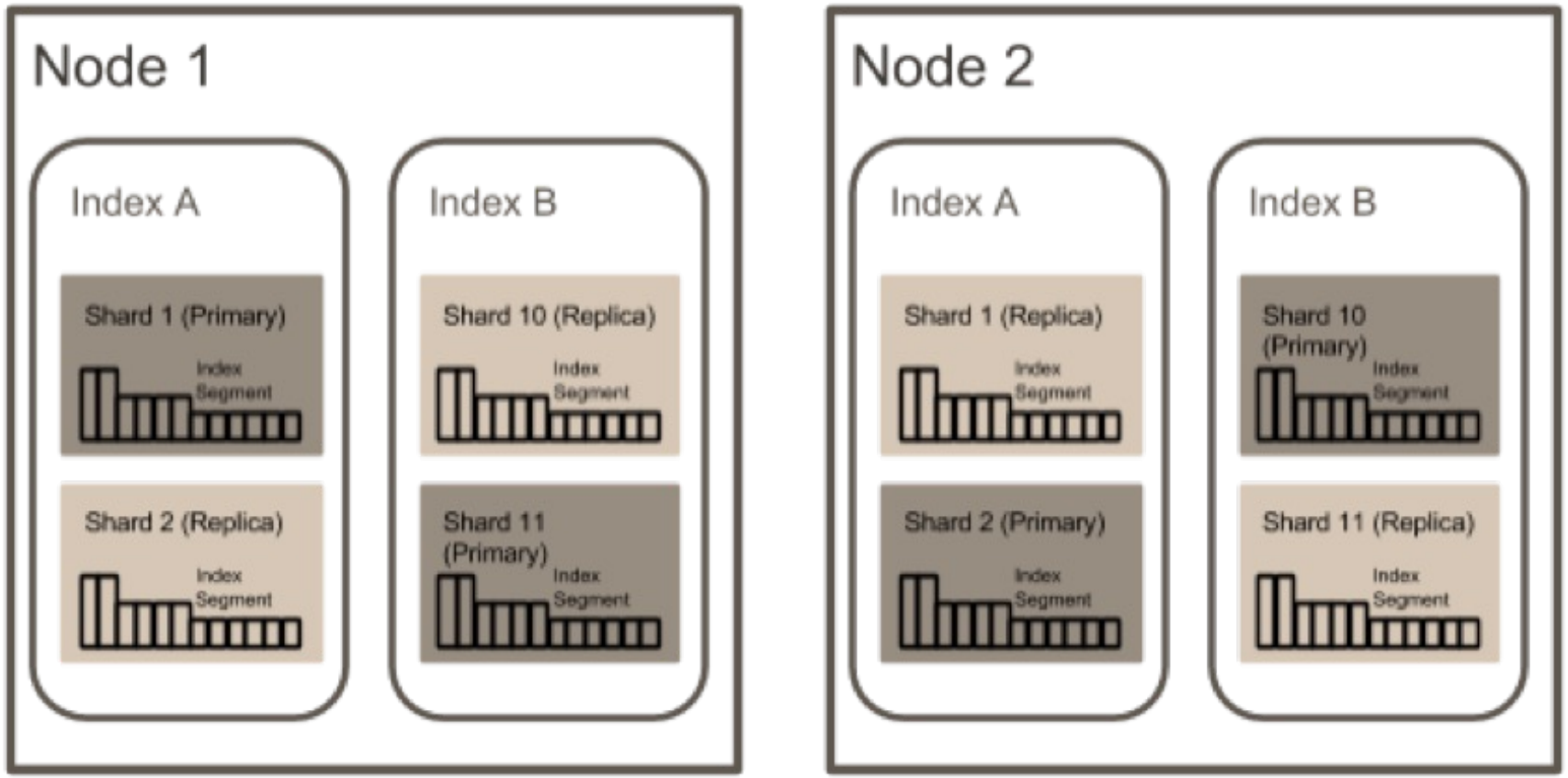
- Mejorar la tolerancia a errores: un fragmento de réplica puede promoverse a un fragmento primario si el primario falla
- Mejorar el rendimiento: las solicitudes de obtención y búsqueda pueden ser manejadas por fragmentos primarios o por réplicas.

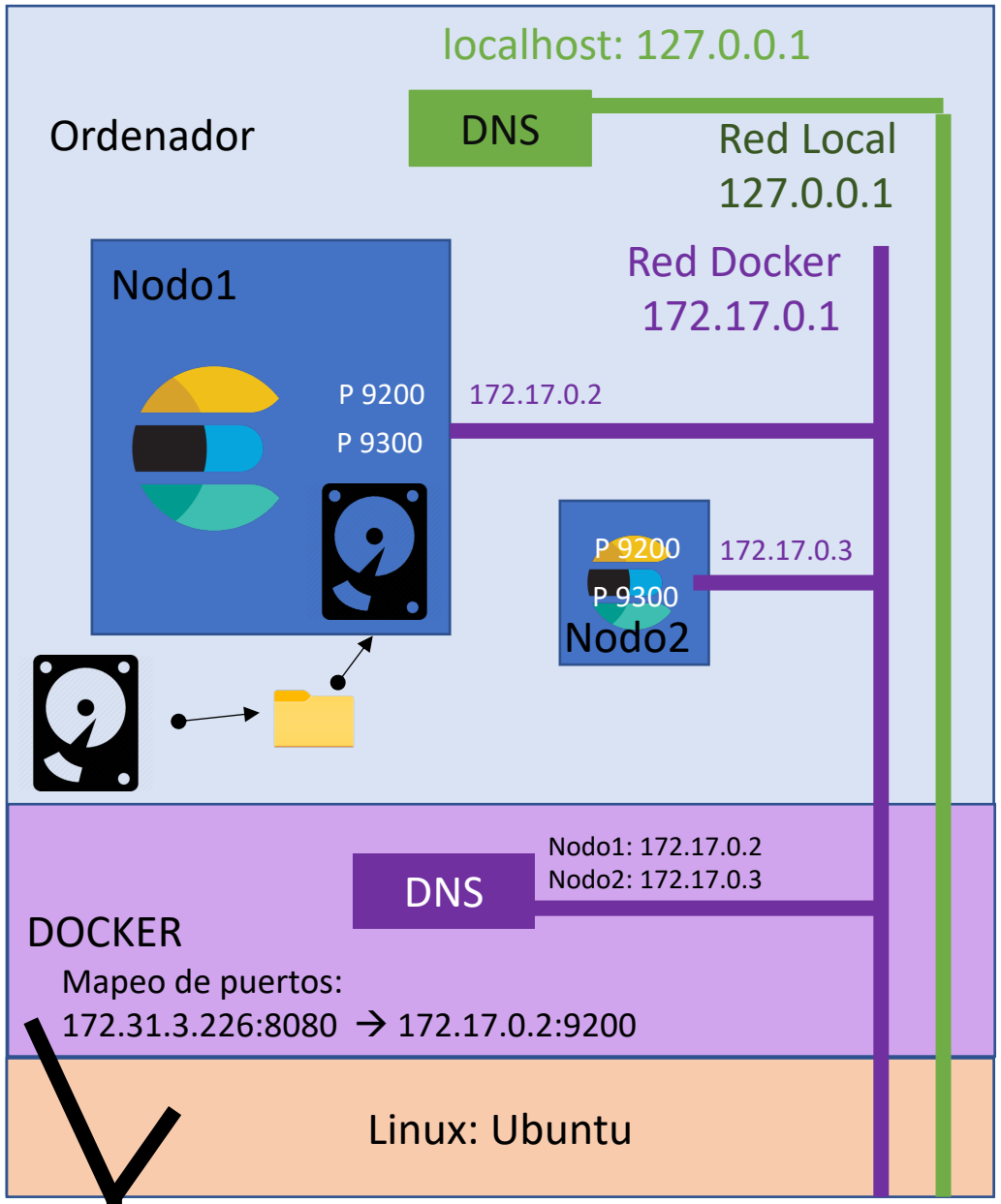
REPLICA SHARD

De manera predeterminada, cada fragmento primario tiene una réplica, pero la cantidad de réplicas se puede cambiar dinámicamente en un índice existente.

Un fragmento de réplica nunca se iniciará en el mismo nodo que su fragmento primario.

NODOS, ÍNDICES Y SHARDS





P: 9200-9299, puerto para que los clientes se comuniquen con ES
P: 9300-9399, puerto para comunicación entre nodos del cluster

Ayer usamos el 8080: Ya que AMAZON por defecto abre los puerto 8080, 8081, 8082

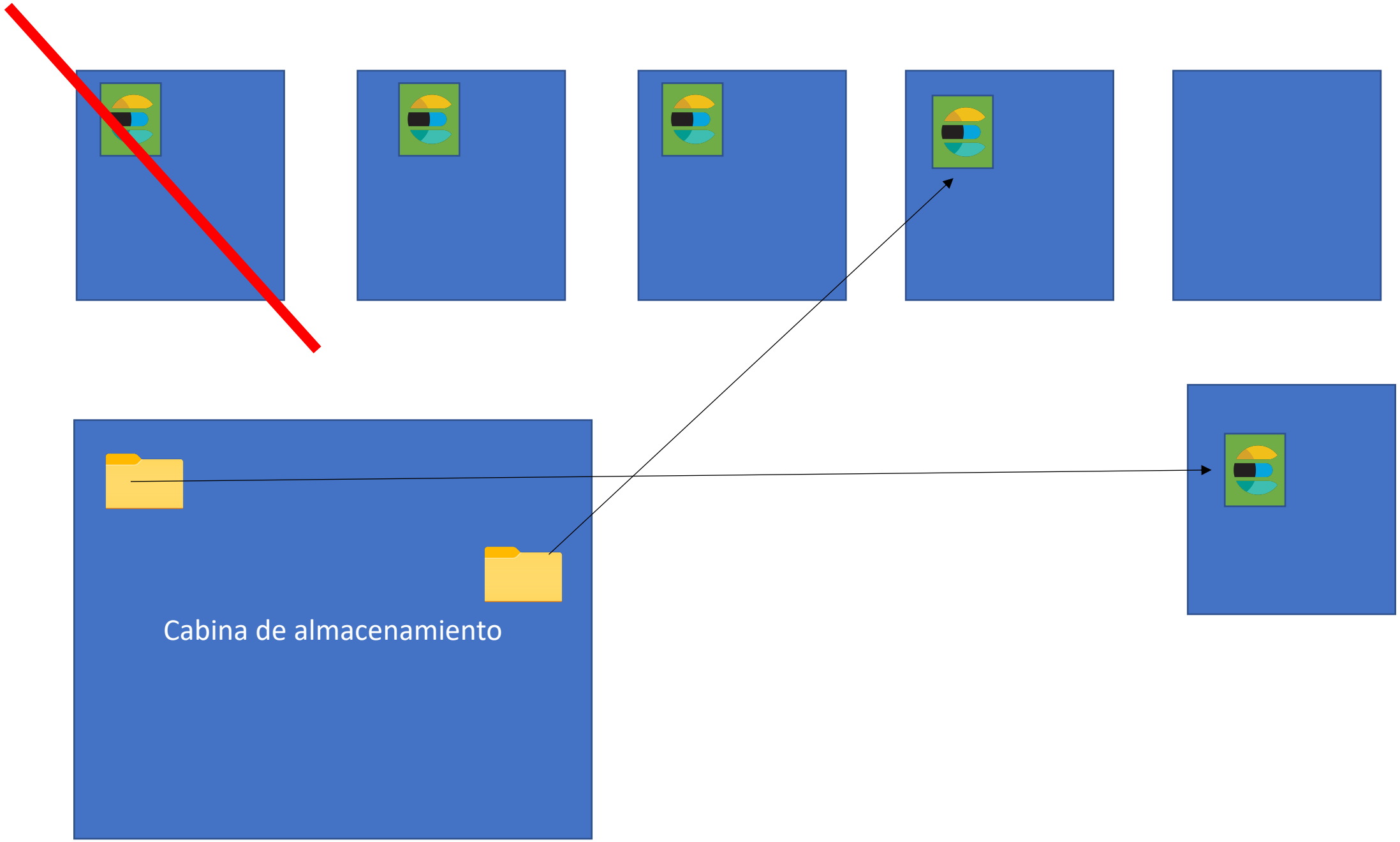
172.17.0.2:9200

Carmen
Iván

Ethernet AWS

DNS

FIRE
WALL


Cabina de almacenamiento

ANALYSIS - ANÁLISIS

El análisis es el proceso de convertir un campo de texto a términos.
En función del analizador utilizado, estas frases:

- FOO BAR - Foo-Bar - foo, bar

probablemente todas resulten en los términos foo y bar.

Estos términos son los que realmente se almacenan en el índice.

ANALYSIS - ANÁLISIS

Una consulta full-text del texto FoO:bAR también será analizada dando lugar a los términos foo y bar, y por lo tanto coincidirá con los términos almacenados en el índice.

Es este proceso de análisis (tanto en tiempo de índice como en tiempo de búsqueda) lo que permite a Elasticsearch realizar consultas full-text.

TEXT ANALYZER

Por defecto, Elasticsearch usa el analizador standard para todos los análisis de texto, que está pensado para usar con la mayoría de los lenguajes naturales y casos de uso.

Si el analizador estándar no se ajusta a las necesidades del proyecto, Elasticsearch trae otros analizadores integrados.

Los analizadores integrados disponen de algunas opciones para ajustar su comportamiento. Por ejemplo, es posible definir una lista de palabras personalizadas para su eliminación.

TEXT ANALYZER

ElasticSearch también permite la utilización de analizadores personalizados, lo que le brinda un mayor control sobre el proceso.

Funcionan en fases: Lowercase, charfilters, tokenizers, stopwords, stemmers...

TERM - TÉRMINO

Un término es un valor exacto que se indexa en Elasticsearch.

Los términos foo, Foo, FOO no son equivalentes.

Puede que los textos foo, Foo, FOO si lo sean en función del analizador utilizado.

Los términos (es decir, valores exactos) se pueden buscar mediante consultas de términos.

ROUTING - ENRUTAMIENTO

Cuando se indexa un documento, se almacena en un único fragmento primario.

Ese fragmento se elige haciendo un hash del valor routing.

De manera predeterminada, el valor routing se deriva del ID del documento o, si el documento tiene un documento primario asociado, del ID del documento primario (para garantizar que los documentos secundarios y primarios se almacenen en el mismo fragmento).

ROUTING - ENRUTAMIENTO

Este comportamiento se puede cambiar especificando un valor para routing en el momento de la indexación o un campo routing en el mapping.

LEADER INDEX

Son los índices origen para realizar replicación entre clústeres.

Son los índices que existen en los clústeres remotos y que se replican en el clúster local como follower index.

FOLLOWER INDEX

Los índices seguidores son los índices de destino para la replicación entre clústeres.

Existen en su clúster local y replican índices líderes.

QUERY - CONSULTA

Una solicitud de información en Elasticsearch.

Es como una pregunta, escrita de una manera que Elasticsearch la entienda.

Una búsqueda consta de una o más consultas combinadas.

Hay dos tipos de consultas:

- consultas de puntuación
- filtros

FILTER - FILTRO

Un filtro es una consulta no puntuable, lo que significa que no puntúa los documentos.

Solo se preocupa por responder a la pregunta: "¿Coincide este documento?".

La respuesta es siempre un simple, binario, sí o no.

Este tipo de consultas se realiza en contextos de filtro.

Los filtros son verificaciones simples para establecer la inclusión o exclusión.

En la mayoría de los casos, el objetivo del filtrado es reducir la cantidad de documentos que deben examinarse.

REINDEX - REINDEXAR

Recorrer algunos o todos los documentos en uno o más índices, volviendo a escribirlos en el mismo índice o en uno nuevo, en un clúster local o remoto.

Esto se hace más comúnmente tras actualizar los mappings o para actualizar Elasticsearch entre dos versiones de índice incompatibles.

RECOVERY - RECUPERACIÓN

La recuperación de fragmentos es el proceso de sincronizar un fragmento de réplica desde un fragmento primario.

Al finalizar, el fragmento de réplica está disponible para búsquedas.

RECOVERY - RECUPERACIÓN

La recuperación se produce automáticamente durante los siguientes procesos:

- Inicio de un nodo o tras una falla. Este tipo de recuperación se llama local store recovery.
- Replicación de un fragmento primario.
- Reubicación de un fragmento en un nodo diferente en el mismo clúster.
- Restauración de instantáneas (snapshots).

CROSS-CLUSTER REPLICATION (CCR)

La función de replicación de clúster cruzado permite replicar índices en clústeres remotos al clúster local.

CROSS-CLUSTER SEARCH (CCS)

La función de búsqueda entre clústeres cruzados permite que cualquier nodo actúe como un cliente federado en varios clústeres

FORMACION DE UN CLUSTER

Cuando se inicia un cluster, los nodos comienzan a ponerse en comunicación (unicast) y se van presentando unos a otros.

Los nodos habilitados para ello, se encargan de elegir un nodo maestro por votación.

Elastic.conf

```
cluster.name=cluster-2m-1i-2d-2c
discovery.seed_hosts=master2,ingest1,data1,data2,coordinator1,coordinator2
cluster.initial_master_nodes=master1,master2
cluster.join.timeout=180s
cluster.publish.timeout=180s
```

master*+ 0 0 Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

ROLES DE NODOS EN UN CLUSTER

Elegible para maestro

De datos

De ingesta

De coordinación

ELEGIBLE PARA MAESTRO

El nodo maestro es responsable de ciertas acciones en el clúster:

- crear o eliminar un índice
- rastrear qué nodos son parte del clúster
- decidir qué fragmentos asignar a qué nodos.

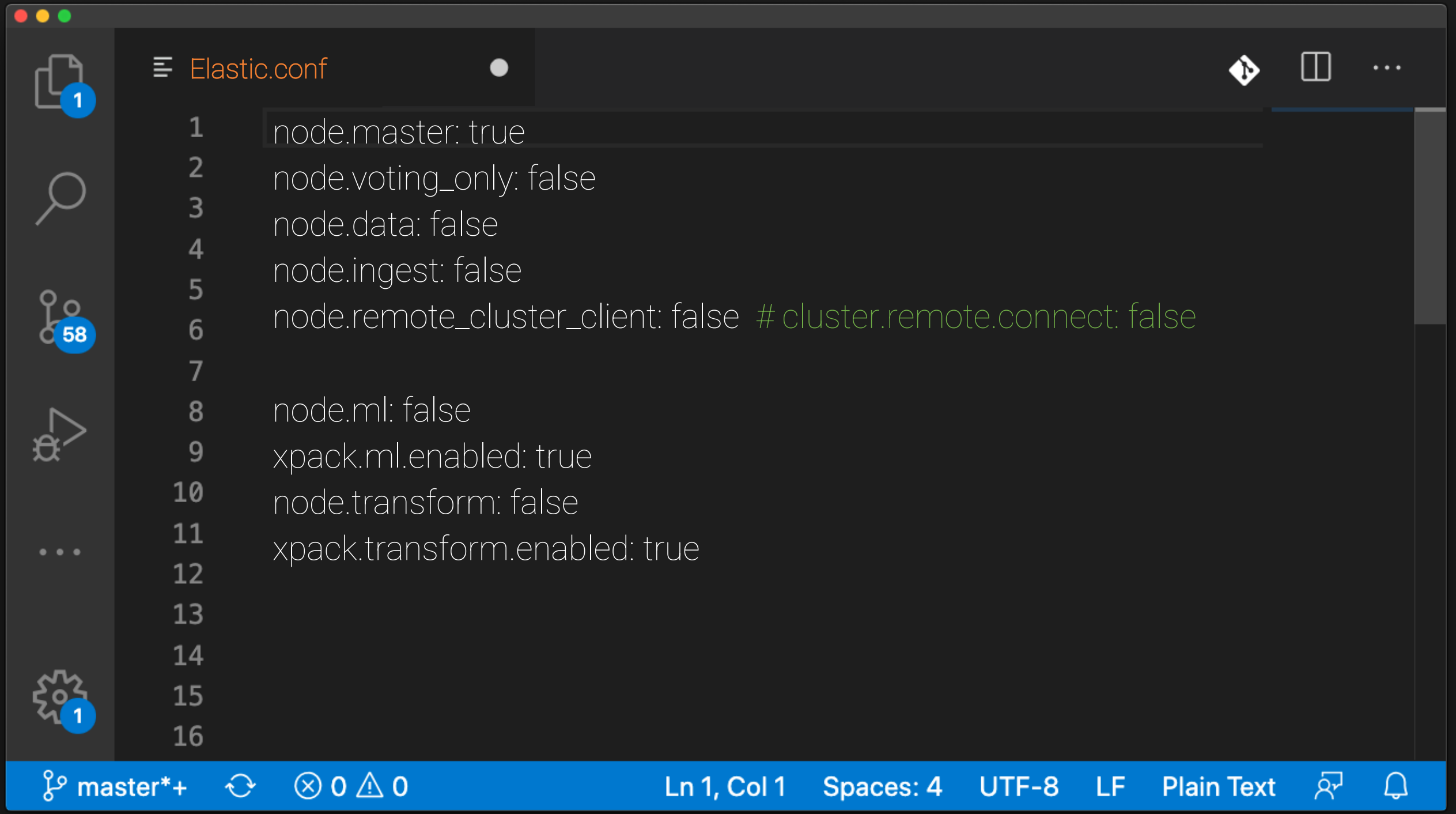
Cualquier nodo maestro elegible que no sea un nodo de solo votación puede ser elegido para convertirse en el nodo maestro mediante el proceso de elección maestra .

ELEGIBLE PARA MAESTRO

El nodo maestro es responsable de ciertas acciones en el clúster:

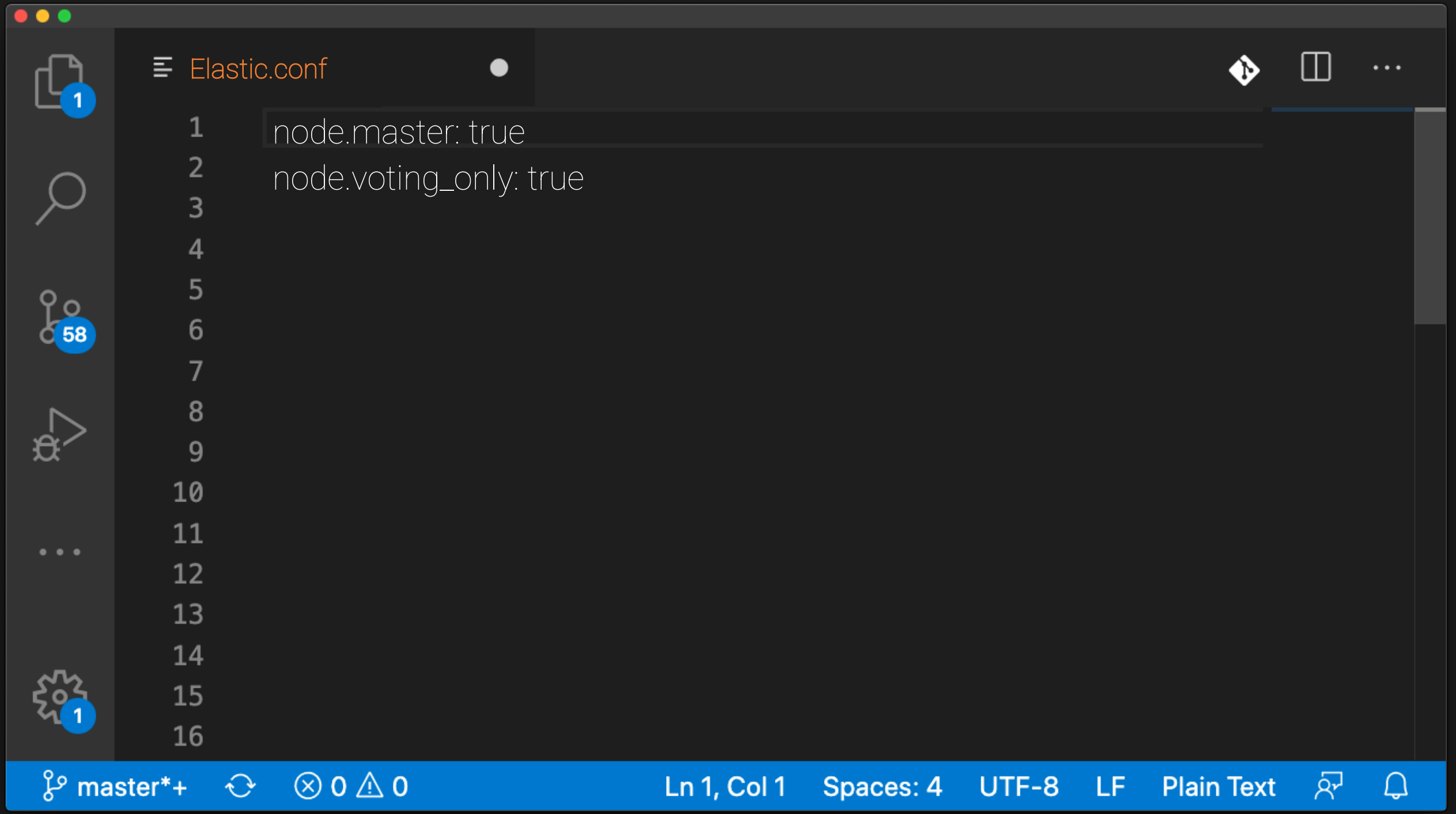
- crear o eliminar un índice
- rastrear qué nodos son parte del clúster
- decidir qué fragmentos asignar a qué nodos.

Cualquier nodo maestro elegible que no sea un nodo de solo votación puede ser elegido para convertirse en el nodo maestro mediante el proceso de elección maestra .


Elastic.conf
node.master: true
node.voting_only: false
node.data: false
node.ingest: false
node.remote_cluster_client: false # cluster.remote.connect: false
node.ml: false
xpack.ml.enabled: true
node.transform: false
xpack.transform.enabled: true
master*+
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text

VOTACION DE MAESTRO

Un nodo principal elegible solo para votación es un nodo que participa en elecciones principales pero que no actuará como el nodo principal elegido del clúster. En particular, un nodo de solo votación puede servir como desempate en las elecciones.

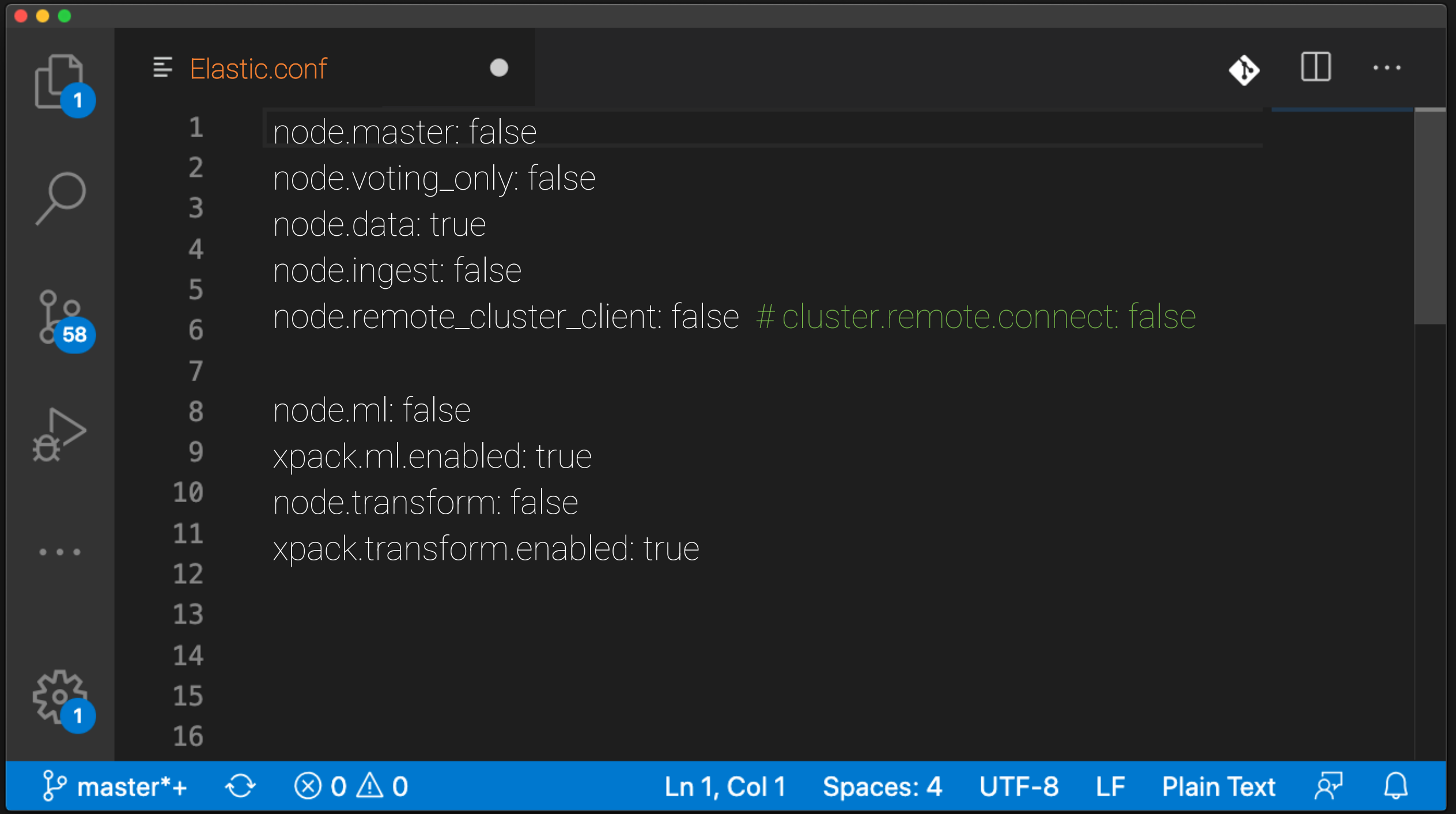

Elastic.conf
node.master: true
node.voting_only: true
master*+
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text

NODO DE DATOS

Los nodos de datos contienen los fragmentos con los documentos que se han indexado.

Los nodos de datos manejan operaciones relacionadas con datos como indexación, búsqueda y agregaciones.

Estas operaciones son intensivas en E / S, memoria y CPU. Es importante monitorear estos recursos y agregar más nodos de datos si están sobrecargados.


Elastic.conf
node.master: false
node.voting_only: false
node.data: true
node.ingest: false
node.remote_cluster_client: false # cluster.remote.connect: false
node.ml: false
xpack.ml.enabled: true
node.transform: false
xpack.transform.enabled: true
master*+
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text

NODO DE INGESTA

Los nodos de ingesta pueden ejecutar tareas de preprocesamiento.

Dependiendo del tipo de operaciones realizadas por los procesadores de ingesta y los recursos requeridos, puede tener sentido tener nodos de ingesta dedicados, que solo realicen esta tarea específica.

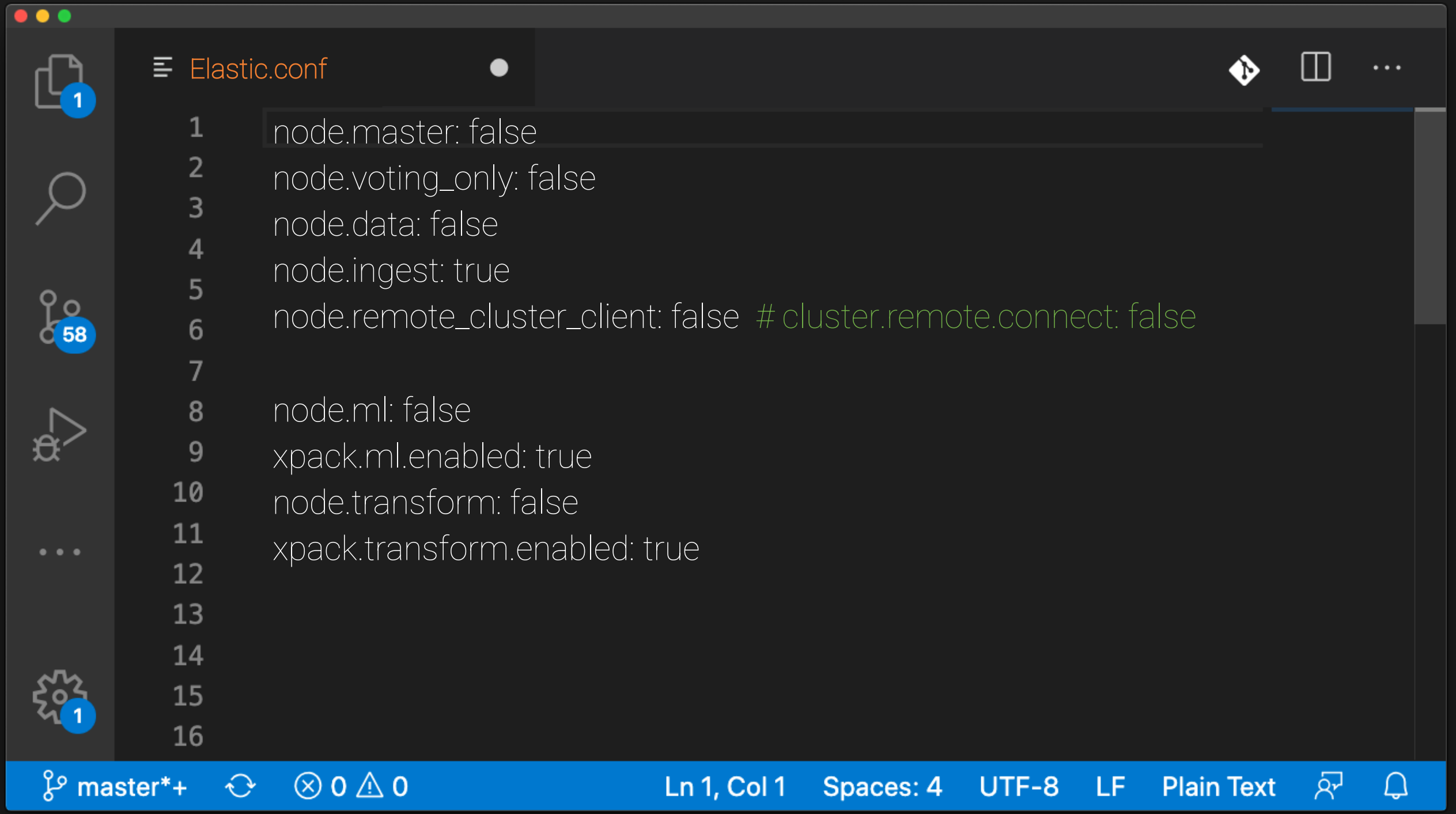
NODO DE INGESTA

Si se eliminan las capacidades de poder :

- manejar tareas maestras
- gestionar datos
- preprocesar documentos

queda un nodo de *coordinación* que solo puede:

- enrutar solicitudes
- manejar la fase de reducción de búsqueda
- distribuir indexación masiva.

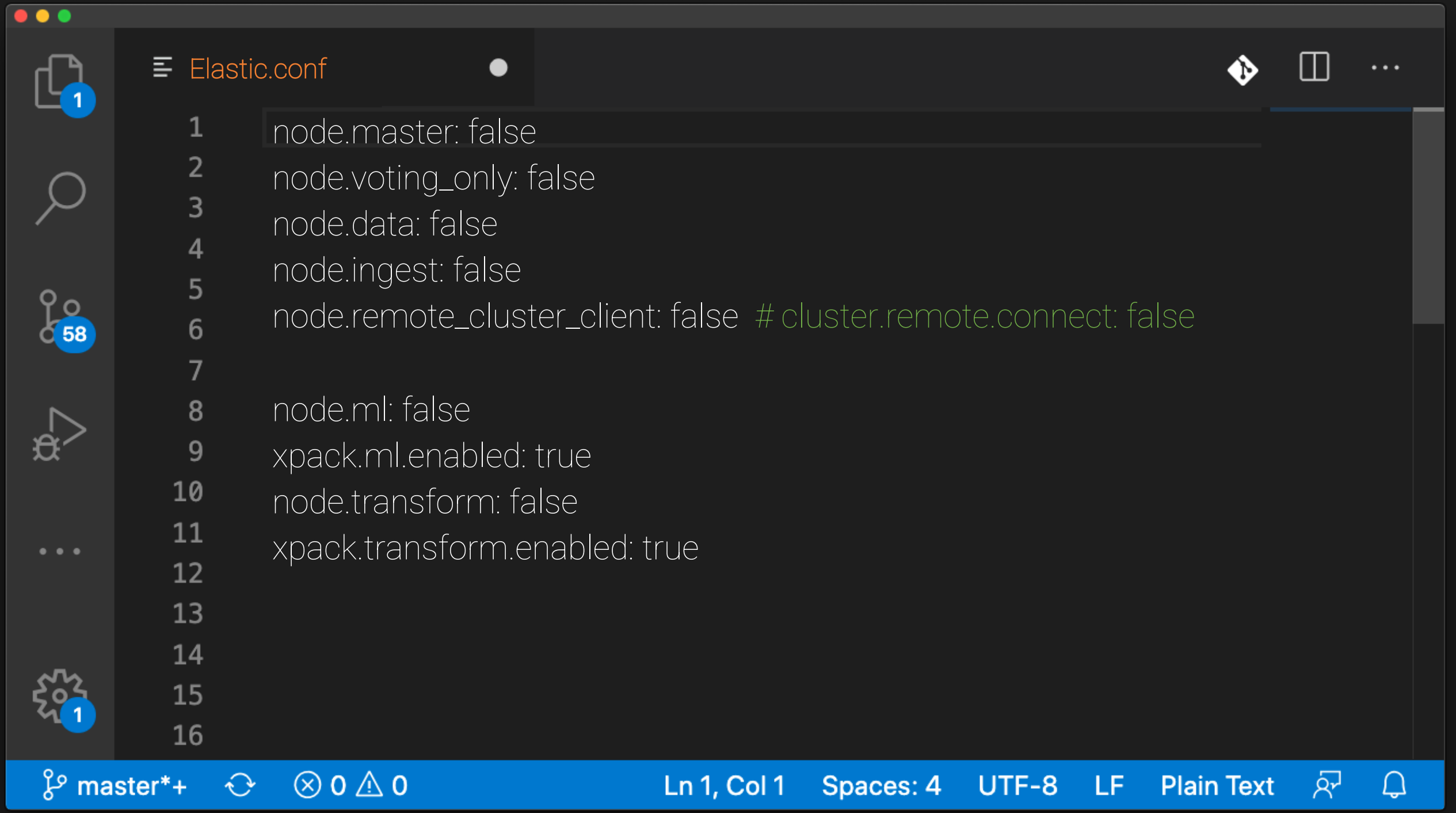

Elastic.conf
node.master: false
node.voting_only: false
node.data: false
node.ingest: true
node.remote_cluster_client: false # cluster.remote.connect: false
node.ml: false
xpack.ml.enabled: true
node.transform: false
xpack.transform.enabled: true
master*+
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text

NODO DE COORDINACION

Los nodos de coordinación se comportan como equilibradores de carga inteligentes.

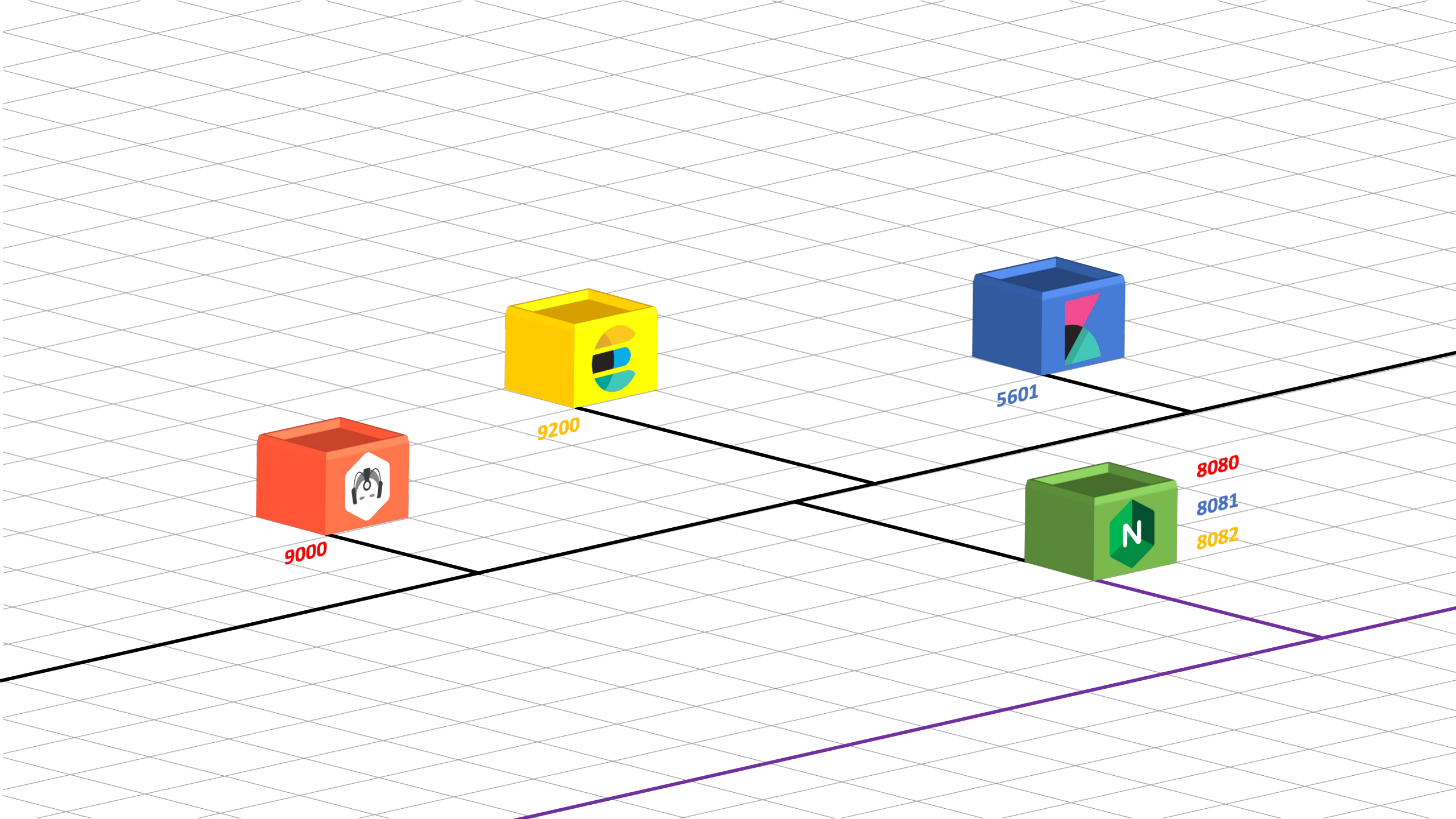
Los nodos de coordinación solamente pueden beneficiar a los clústeres grandes al quitar trabajo a los nodos de datos y maestros.

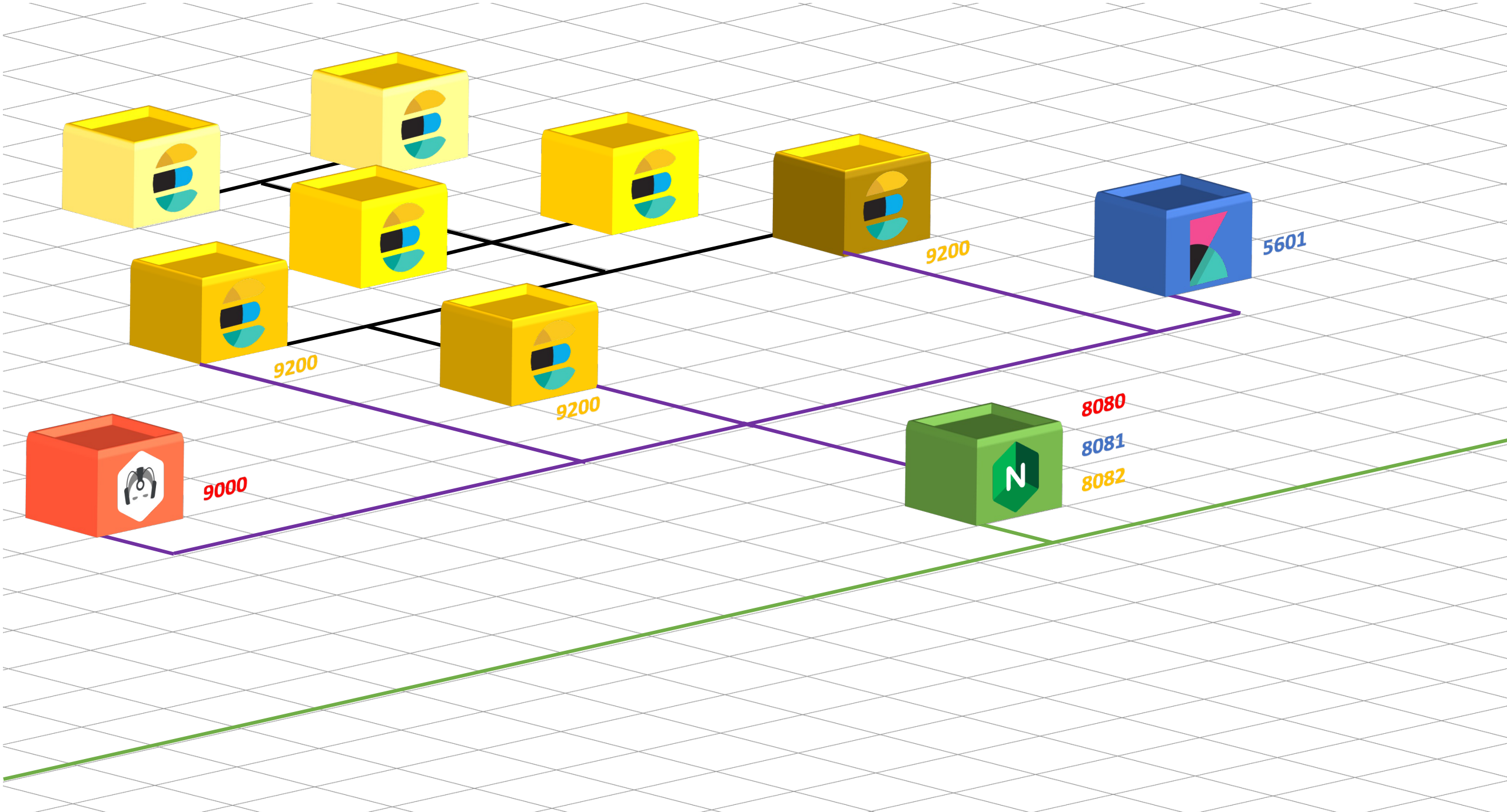
Se unen al clúster y reciben el estado completo del clúster , como cualquier otro nodo, y usan el estado del clúster para enrutar las solicitudes directamente a los lugares apropiados.


Elastic.conf
node.master: false
node.voting_only: false
node.data: false
node.ingest: false
node.remote_cluster_client: false # cluster.remote.connect: false
node.ml: false
xpack.ml.enabled: true
node.transform: false
xpack.transform.enabled: true
master*+
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Plain Text

Ejemplos de configuraciones

A continuación se muestran 2 de las configuraciones que de elasticsearch que se efectuarán durante la formación.


9200
5601
9000
8080
8081
8082


9200
5601
9200
9200
9000
8080
8081
8082

CONFIGURACIONES DE ÍNDICE

Se pueden establecer configuraciones a nivel de cada uno de los índices.

Las propiedades de configuración puede ser:

- Estáticas: Solo se pueden establecer en el momento de creación del índice.
- Dinámicas: Se pueden cambiar posteriormente utilizando la API update-index-settings.

CONFIGURACIONES ESTÁTICAS

index.number_of_shards

El número de fragmentos primarios que debe tener un índice.

El valor predeterminado es 1.

Esta configuración solo se puede establecer en el momento de creación del índice.

CONFIGURACIONES ESTÁTICAS

index.shard.check_on_startup

Si los fragmentos pueden verificarse para ver si están corruptos antes de abrirlos.

Cuando se detecta corrupción, evitará que se abra el fragmento.

Valores:

- false: (predeterminado) No compruebe si hay corrupción al abrir un fragmento.
- checksum: Verifica la corrupción física.
- true: Verifique la corrupción tanto física como lógica. Esto es mucho más costoso en términos de uso de CPU y memoria.

Verificar fragmentos puede llevar mucho tiempo en índices grandes.

CONFIGURACIONES ESTÁTICAS

index.codec

Por defecto, los datos almacenados se comprimen con compresión LZ4.

El valor best_compression utiliza el algoritmo DEFLATE, que ofrece una relación de compresión más alta, a expensas de un rendimiento más lento de los campos almacenados.

Si está actualizando el tipo de compresión, el nuevo se aplicará después de fusionar los segmentos.

La fusión de segmentos se puede forzar usando la fusión forzada.

CONFIGURACIONES ESTÁTICAS

index.routing_partition_size

El número de fragmentos a los que puede ir un valor de enrutamiento personalizado.

El valor predeterminado es 1 y solo se puede establecer en el momento de creación del índice.

Este valor debe ser menor que el index.number_of_shards (salvo si number_of_shards vale 1, que podrá tomar el valor 1).

CONFIGURACIONES ESTÁTICAS

index.hidden

Indica si el índice debe estar oculto de forma predeterminada.

Los índices ocultos no se devuelven de manera predeterminada cuando se usa una expresión comodín.

Los valores posibles son true y false (predeterminado).

CONFIGURACIONES DINÁMICAS

index.number_of_replicas

El número de réplicas que tiene cada fragmento primario.

Por defecto es 1.

CONFIGURACIONES DINÁMICAS

index.auto_expand_replicas

Expande automáticamente el número de réplicas en función del número de nodos de datos en el clúster.

Establecer en un guión delimitado límite inferior y superior (por ejemplo 0-5) o utilizar all para el límite superior (por ejemplo 0-all).

El valor predeterminado es false (es decir, deshabilitado).

CONFIGURACIONES DINÁMICAS

index.refresh_interval

Con qué frecuencia realizar una operación de actualización, que hace que los cambios recientes en el índice sean visibles para la búsqueda.

Por defecto es 1s.

Se puede configurar -1 para deshabilitar la actualización automática.

CONFIGURACIONES DINÁMICAS

index.max_result_window

Por defecto es 10000.

CONFIGURACIONES DINÁMICAS

index.max_inner_result_window

Por defecto es 100.

CONFIGURACIONES DINÁMICAS

index.blocks.read_only

El valor true hace que el índice y los metadatos del índice sean de solo lectura.

False permite cambios en los metadatos.

CONFIGURACIONES DINÁMICAS

index.blocks.read

El valor true deshabilita las operaciones de lectura en el índice.

CONFIGURACIONES DINÁMICAS

index.blocks.write

El valor true deshabilita las operaciones de escritura de datos en el índice.

CONFIGURACIONES DINÁMICAS

index.blocks.metadata

El valor true deshabilita las lecturas y escrituras de metadatos de índice.

UBICACIÓN DE SHARDS

Existen varias configuraciones para controlar cómo se asignan los shards a los nodos del cluster:

- Limitación de fragmentos totales por nodo
- Retraso en la reubicación de fragmentos
- Reglas de filtrado para asignación de fragmentos

FRAGMENTOS TOTALES POR NODO

index.routing.allocation.total_shards_per_node

El número máximo de fragmentos (réplicas y primarias) que se asignarán a un solo nodo, para un determinado índice.

El valor predeterminado es ilimitado (-1).

RETRASAR LAS REUBICACIONES

index.unassigned.node_left.delayed_timeout

El tiempo que elasticsearch espera desde que pierde la comunicación con un nodo, antes de prodecer a la reubicación de sus fragmentos.

Por defecto es 1m (1 minuto).

REGLAS PARA ASIGNACIONES

Paso 1: Crear atributos (tags) asociados a cada nodo:

node.attr.tamano: medio

Paso 2: Agregar filtros de asignación para cada índica que se quiera controlar:

PUT mi_indice/_settings {
"index.routing.allocation.[include|require|exclude].tamano" : "mediano" }

Nota: Se pueden usar caracteres comodín

- PUT test / _settings { "index.routing.allocation.include._ip" : "192.168.2. *" }

REGLAS PARA ASIGNACIONES

Nota: Atributos integrados en elasticsearch

- _name: Hacer coincidir nodos por nombre de nodo
- _host_ip: Haga coincidir los nodos por la dirección IP del host (IP asociada con el nombre del host)
- _publish_ip: Hacer coincidir nodos por dirección IP de publicación
- _ip: Match ya sea _host_ip o _publish_ip
- _host: Hacer coincidir nodos por nombre de host
- _id: Hacer coincidir nodos por ID de nodo

CONFIGURACIONES DINÁMICAS

index.routing.allocation.enable

Permite establecer a que fragmentos del índice en el cluster se le aplican los anteriores filtros.

Se puede establecer en:

- all (predeterminado): permite filtrar la ubicación de todos los fragmentos del índice.
- primaries - Permite filtrar la ubicación de fragmentos primarios.
- new_primaries - Permite filtrar la ubicación de fragmentos primarios recién creados.
- none - No se aplican filtros a la ubicación de fragmentos.

Congelar Indices

Los índices de Elasticsearch mantienen algunas estructuras de datos en la memoria para aumentar su eficiencia.

Si hay muchos índices, la memoria requerida. Puede ser muy grande

Para los índices que se usan con frecuencia, es mejor mantener estas estructuras en la memoria porque lleva tiempo reconstruirlas.

Para los índices que se usan muy poco, es preferible liberar la memoria correspondiente y reconstruir estas estructuras de datos en cada búsqueda.

Congelar Indices

Al congelar un índice, se construyen las estructuras de datos transitorias de cada fragmento cada vez que se busca en ese fragmento, y se descartan estas estructuras al completar la búsqueda.

Los índices congelados consumen muchos menos recursos que los índices normales.

Los índices congelados son de solo lectura.

Los índices congelados no están destinados a una alta carga de búsqueda.

Cerrado de Indices

En un índice cerrado no se puede ni buscar ni indexar.

No consumen recursos, más que espacio en disco.

Hay operaciones que requieren realizarse sobre índices cerrados.

Un índice se puede cerrar y luego abrir en cualquier momento posterior.