Università degli Studi di Camerino

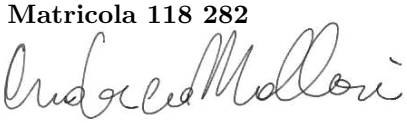
Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica per la comunicazione digitale (Classe L-31)

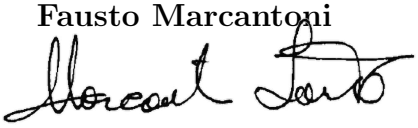
Monitoraggio remoto di macchinari CNC tramite Elasticsearch

Laureando
Andrea Malloni

Matricola 118 282



Relatore
Fausto Marcantoni



Correlatore
Matteo Malloni

A.A. 2023/2024

Indice

1 Introduzione **9**
- 1.1 Motivazioni ed obiettivi . . . 9
- 1.2 Elastic Stack . . . 10
- 1.3 Tecnologie impiegate . . . 11
  - 1.3.1 Elastic Stack . . . 11
  - 1.3.2 Web Stack . . . 11
- 1.4 Struttura della Tesi . . . 12

2 Architettura del sistema **13**
- 2.1 Perchè Elastic Stack? . . . 14
- 2.2 API Elasticsearch e Query DSL . . . 14
- 2.3 Precondizioni . . . 17
- 2.4 Architettura proposta . . . 17

3 Elasticsearch: integrazione dei macchinari **19**
- 3.1 Connessione in rete dei macchinari CNC . . . 19
- 3.2 Memorizzazione dei dati . . . 20
- 3.3 Fetching dei dati: Logstash . . . 21

4 Web app: monitoraggio remoto dei macchinari **23**
- 4.1 Precondizioni . . . 23
- 4.2 Utilizzo delle API . . . 23
- 4.3 Dashboard . . . 27
- 4.4 Monitoraggio statistico . . . 29
- 4.5 Codici . . . 30

5 Ottimizzazioni e miglioramenti **31**
- 5.1 Elasticsearch sharding . . . 32
- 5.2 Gestione dell'utenza e delle sorgenti dati . . . 33
- 5.3 Client autoprovisioning . . . 34

6 Conclusioni **37**

A Installazione ed utilizzo di Elastic stack **39**
- A.1 Installazione di Elasticsearch . . . 39

A.2 Installazione di Kibana 40
A.3 Installazione di Logstash 42
A.3.1 Ingestione di dati da database MySQL 43
A.4 Dashboard di monitoraggio 45
A.5 Configurazione dei profili utente 48

Elenco dei codici

4.1 Codice di raccolta parametrizzata dei log 24
4.2 Codice di connessione al nodo Elasticsearch 25
4.3 Codice di raccolta non parametrizzata dei log 25
4.4 Funzione di aggiornamento dello stato dei macchinari 30
4.5 Recupero dello stato dei macchinari 30
4.6 Recupero dei log di dettaglio per specifico macchinario 30

Elenco delle figure

1.1 Struttura di un tornio verticale 9
1.2 Componenti dell'Elastic Stack [Gee] 10

2.1 Schematizzazione della soluzione realizzata 13

3.1 Schematizzazione DHCP 19
3.2 Pipeline di ingestione dati tramite Logstash 22

4.1 Dashboard principale di visualizzazione dei macchinari 27
4.2 Storico dei log registrati per macchinario 28
4.3 Pagina di ricerca globale dei log registrati 28
4.4 Pagina di visualizzazione statistica 29

5.1 Schematizzazione dell'architettura ottimizzata 31
5.2 Sharding di un indice Elasticsearch [Gup22] 32

6.1 Indice risultante dall'elaborazione 37

A.1 Pagina Kibana di registrazione del token. 41
A.2 Dashboards overview in Kibana 45
A.3 Dashboard vuota 45
A.4 Schermata di personalizzazione di una visualizzazione 46
A.5 Esempio di istogramma realizzato tramite Lens 46
A.6 Esempio di dashboard con visualizzazione 47
A.7 Salvataggio di una nuova dashboard 47
A.8 Users overview in Kibana 48
A.9 Creazione di un profilo utente in Kibana 48
A.10 Esempio di utente con ruolo di visualizzazione 49

1. Introduzione

In ambito industriale, per le aziende di produzione in cui é prevista una lavorazione meccanica, sono largamente utilizzati i macchinari automatici; Stiamo parlando, in particolar modo, di macchinari a controllo numerico computerizzato (CNC), ovvero macchinari utensili, solitamente con braccia meccaniche, che vengono controllati sfruttando le capacità di computazione di un computer. L'introduzione del controllo computerizzato nei macchinari di lavorazione, avvenuto poco più di mezzo secolo fa, ha determinato un punto di svolta fondamentale per l'ambito della meccanica, in quanto grazie ad esso è stato possibile raggiungere un'altissima velocità e precisione delle lavorazioni, che assieme alla grande versatilità di cui godono questi macchinari rappresentano i motivi principali per i quali la loro diffusione ha subito una grandissima impennata a partire dagli anni '80, prima dei quali restavano una tecnologia di nicchia, impiegata solo per lavorazioni ad alta precisione. [Wika]

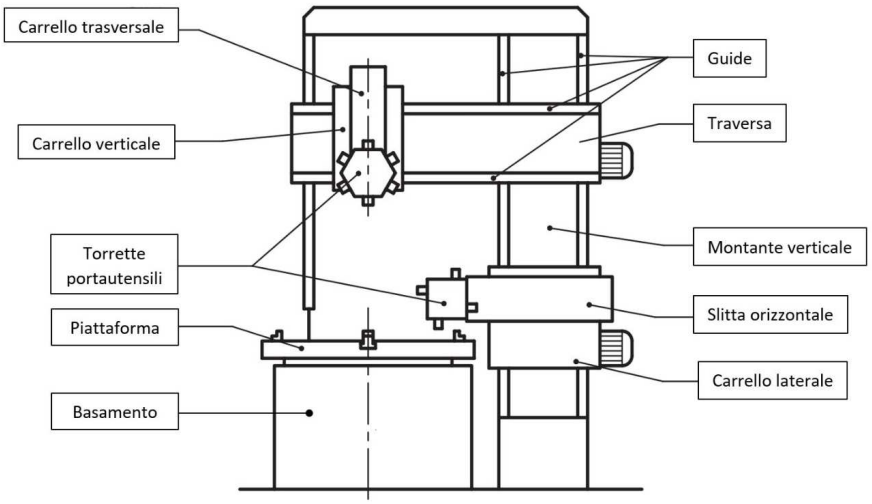


Figura 1.1: Struttura di un tornio verticale

La struttura di un tornio verticale, riportata nella figura 1.1, rappresenta uno fra i vari tipi di macchinari CNC più comunemente utilizzati. Ulteriori esempi di macchinari impiegati nelle lavorazioni sono presse piegatrici, punzonatrici, fresatrici, saldatrici e macchine da taglio.

1.1 Motivazioni ed obiettivi

Il progetto trattato è frutto dell'esperienza di tirocinio effettuata presso la New Assistent SRL, ed in particolar modo nasce dalle esigenze di un cliente committente dell'a-

zienda, le cui attività ruotano intorno alla realizzazione ed alla vendita di macchinari CNC. Le motivazioni del progetto sono da ricercare nella volontà del cliente di abbracciare la rivoluzione portata dall'industria 4.0: con questo termine, ci si riferisce alla sempre più progressiva e diffusa integrazione di tecnologie digitali intelligenti nei processi manifatturieri ed industriali [SAP], che in questo caso si concretizza attraverso la richiesta di integrazione di soluzioni di interconnessione di rete in macchinari già esistenti e collaudati.
Più precisamente, l'obiettivo del progetto è quello di realizzare un sistema di monitoraggio che consenta, ad ogni utente utilizzatore di macchinari CNC interconnessi al sistema, di visualizzare in tempo reale lo stato dei suoi macchinari e lo storico dei log registrati da essi. Tutto ciò è stato raggiunto attraverso una infrastruttura dedicata all'acquisizione dei dati dai macchinari, situati nelle varie sedi fisiche, ed un portale web che consente una visualizzazione dei dati, rapida e semplificata, all'utente finale.

1.2 Elastic Stack

Per un'organizzazione con necessità come quelle esposte dal cliente in questione, risulta indubbiamente una soluzione di alta rilevanza l'Elastic Stack. Con questo termine si intende un ecosistema completo e potente di strumenti open source progettati per la gestione, l'analisi e la visualizzazione dei dati in tempo reale. È stato sviluppato da Elastic, un'azienda leader nel settore dell'analisi dei dati e della ricerca.

Figura 1.2: Componenti dell'Elastic Stack [Gee]

A livello implementativo, lo stack si compone di tre software principali. Al cuore dell'Elastic Stack c'è Elasticsearch, un motore di ricerca distribuito e altamente scalabile basato su Apache Lucene. Elasticsearch è il pilastro su cui si fonda l'intero ecosistema, ed è progettato per l'indicizzazione e la ricerca di grandi volumi di dati in modo rapido e efficiente [Elac]. Supporta dati strutturati, semi-strutturati e non strutturati, rendendolo estremamente flessibile per una vasta gamma di casi d'uso. Con la sua architettura distribuita, Elasticsearch consente di scalare per gestire grandi volumi di dati senza compromettere le prestazioni.
Il secondo componente software è invece Logstash, uno strumento di manipolazione dei dati che permette l'ingestione, la trasformazione e l'invio di quest'ultimi, che possono anche provenire da una moltitudine di fonti [Elae]. Logstash può raccogliere log di sistema, metriche di infrastruttura, eventi di rete e altri tipi di dati da diverse sorgenti e normalizzarli per l'analisi.
In ultimo, Kibana è la terza componente chiave dell'Elastic Stack; una piattaforma di visualizzazione dei dati intuitiva e potente. Con Kibana, gli utenti possono creare dashboard interattive, grafici e mappe per esplorare e analizzare i dati archiviati in

Elasticsearch [Elad]. Questo strumento è essenziale per ottenere insights rapidi e approfonditi dai dati in tempo reale.
Recentemente, alla lista dei componenti costituenti dello stack sono stati aggiunti i Beats, agenti leggeri che consentono di raccogliere e inviare dati da sorgenti specifiche a Elasticsearch o a Logstash per l'elaborazione [Elaa]. Ci sono diverse varianti di Beats, ognuna progettata per un caso d'uso specifico, come il monitoraggio dei file di log con Filebeat o delle metriche di sistema con Metricbeat.
L'Elastic Stack non si limita solo alla gestione e all'analisi dei dati, ma include anche soluzioni per il monitoraggio delle prestazioni delle applicazioni (Elastic APM) e per la sicurezza informatica (Elastic Security). Questo rende l'Elastic Stack un'opzione completa per le organizzazioni che desiderano gestire i loro dati, monitorare le prestazioni e proteggere i loro sistemi da minacce potenziali.

1.3 Tecnologie impiegate

La realizzazione del progetto ha richiesto l'impiego di diverse tecnologie, che, per semplicità, ho preferito riportare sommariamente distinguendole in due categorie: Elastic Stack e Web Stack.

1.3.1 Elastic Stack

Nel contesto del progetto trattato, questo stack di tecnologie è impiegato per recuperare i dati provenienti dai macchinari CNC interessati, ed indicizzarli. Più precisamente, ognuna delle componenti software menzionate svolge una funzione ben distinta:

- Elasticsearch: indicizzazione dei dati dei macchinari CNC.
- Logstash: implementazione della pipeline di ingestione dati dai macchinari CNC verso Elasticsearch.
- Kibana: creazione e gestione degli indici dei log.

Anche se non propriamente parte dell'Elastic Stack, rientra nella categoria delle tecnologie di raccolta dati anche MySQL, un sistema di gestione di database relazionali open-source, che consente di archiviare dati strutturati tramite operazioni CRUD. Riporto tale tecnologia in questa sezione in quanto parte integrante della pipeline di ingestione realizzata per il progetto. Ulteriori dettagli ed approfondimenti sul ruolo di questa tecnologia sono forniti nella sezione 2.3.

1.3.2 Web Stack

A differenza dell'Elastic Stack, che è impiegato nella parte di raccolta e manipolazione dati del progetto, lo stack web è l'insieme di tutte le tecnologie adoperate per la realizzazione del portale web di monitoraggio. Fanno parte di questo insieme:

- HTML/CSS/JavaScript: i tre pilastri dello sviluppo web, rispettivamente impiegati per la definizione della struttura delle varie interfacce, per la modellazione del loro stile grafico e per l'implementazione di comportamenti dinamici relativi agli elementi in esse contenuti.

- PHP: linguaggio di scripting server-side utilizzato nella realizzazione della parte di backend della web app. Essa realizza l'accesso ai dati da visualizzare, contenuti in un'istanza Elasticsearch, sfruttando le API messe a disposizione, tramite il pacchetto Composer elasticsearch/elasticsearch[1].

- Bootstrap: framework CSS adoperato per facilitare lo sviluppo della parte estetica dell'interfaccia web, attraverso l'utilizzo delle componenti grafiche predefinite messe a disposizione. L'utilizzo di questo framework è stato specificatamente richiesto dal cliente committente nella forma di AdminLTE, un wrapper di Bootstrap che incorpora al suo interno dei temi grafici alternativi e personalizzati. Dettagli sulle motivazioni di questa scelta tecnica sono forniti nella sezione 4.1.

1.4 Struttura della Tesi

In aggiunta alla presente introduzione, verrano trattati nei capitoli successivi i seguenti argomenti:

- Capitolo 2: spiegazione ed illustrazione dell'architettura del sistema ideato per il soddisfacimento delle esigenze del cliente, discussione delle motivazioni e limitazioni tecniche che hanno portato alla sua realizzazione.

- Capitolo 3: dettagli e discussione delle scelte implementative del sistema di raccolta dei dati dei macchinari attraverso Elastic Stack. Illustrazione generale del flusso dei dati.

- Capitolo 4: dettagli implementativi riguardanti l'interfaccia web di monitoraggio ed analisi dei dati, illustrazione teorica e visuale delle feature realizzate.

- Capitolo 5: discussione su eventuali ottimizzazioni e miglioramenti apportabili al progetto in previsione di un suo possibile impiego su larga scala.

- Capitolo 6: alcune riflessioni di carattere generale sull'esito ed i risultati ottenuti dal progetto.

[1] https://packagist.org/packages/elasticsearch/elasticsearch

2. Architettura del sistema

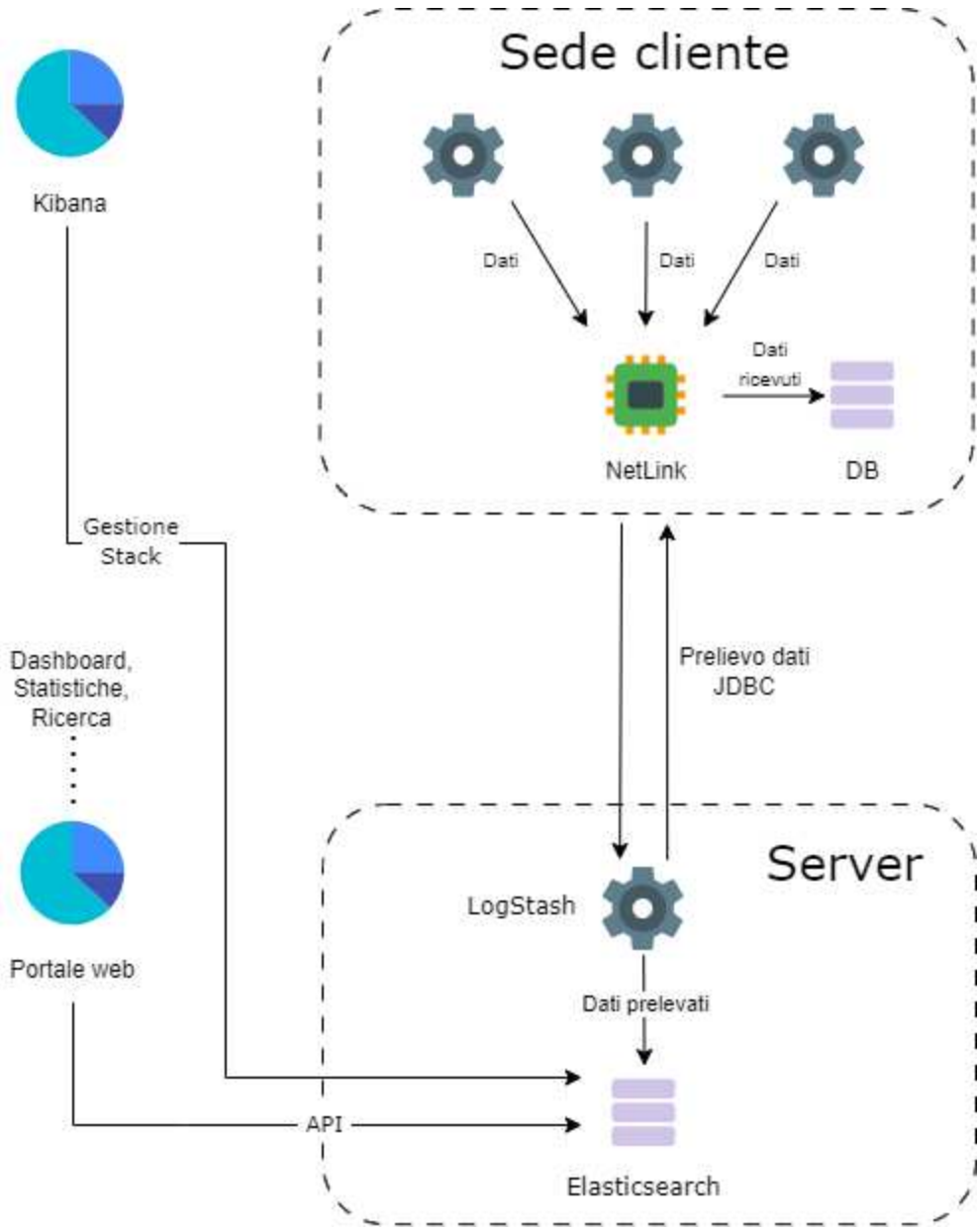


Figura 2.1: Schematizzazione della soluzione realizzata

2.1 Perchè Elastic Stack?

Date le motivazioni esposte nella sezione 1.1, per un'azienda come quella committente di questo progetto, si ha una grande mole di dati da monitorare. Ciò è dovuto al fatto che i macchinari soggetti a controllo sono numerosi, ed i log degli eventi vengono registrati molto frequentemente. In questo contesto, l'Elastic Stack si distingue come soluzione software adeguata, per numerose ragioni.

Innanzitutto, l'Elastic Stack offre una soluzione completa ed integrata per la raccolta, l'archiviazione, l'analisi e la visualizzazione dei dati. Esso è un ecosistema flessibile che può adattarsi alle esigenze specifiche di qualsiasi organizzazione, e quindi, in questo caso, facilmente adattabile alle esigenze di clienti diversi. Questo significa che non è necessario integrare diversi strumenti separati, riducendo così la complessità e i costi operativi. In secondo luogo, l'Elastic Stack è altamente scalabile e può gestire grandi volumi di dati in modo efficiente. Anche avendo centinaia o milioni di documenti da analizzare, l'Elastic Stack può adattarsi dinamicamente senza compromettere le prestazioni. Inoltre, l'Elastic Stack offre potenti funzionalità di ricerca e analisi. Utilizzando Elasticsearch, è possibile eseguire ricerche complesse e ottenere risultati istantanei su grandi dataset. Inoltre, Kibana fornisce strumenti di visualizzazione intuitivi per trasformare i dati in grafici, dashboard e mappe interattive, consentendo di ottenere insights in tempo reale.

Un altro vantaggio significativo dell'Elastic Stack è la sua capacità di gestire dati eterogenei provenienti da diverse fonti. Che si tratti di log di sistema, metriche di infrastruttura, dati di sensori IoT o testo non strutturato, l'Elastic Stack può ingegnerizzare e analizzare tutto in un'unica piattaforma unificata, che favorisce, come già detto, l'adattabilità del sistema risultante a clienti con esigenze diverse. Infine, l'Elastic Stack è open source, il che significa che è pienamente personalizzabile, privo di restrizioni di utilizzo e non necessita l'acquisto di licenze, a meno che non si voglia fare uso di funzionalità avanzate specifiche. Questa libertà offre un notevole vantaggio sia in termini di flessibilità che di controllo sui sistemi di monitoraggio.

In conclusione, l'Elastic Stack rappresenta una scelta eccellente per il monitoraggio dei dati grazie alla sua completezza, scalabilità, potenza analitica, flessibilità e natura open source.

2.2 API Elasticsearch e Query DSL

Per parlare dell'architettura e delle scelte progettuali fatte per la realizzazione del sistema è fondamentale conoscere lo strumento che rende possibile l'interazione con Elasticsearch: le API. Elastic mette a disposizione tutta una serie di API che fungono da interfaccia per l'interazione con le funzionalità offerte da Elasticsearch ed i vari altri componenti dello stack Elastic. In particolare, si distinguno diverse categorie (o gruppi) di API [Elag]:

- Elasticsearch REST API: API RESTful che consentono agli sviluppatori di eseguire operazioni come la ricerca, l'indicizzazione, l'aggiornamento e la cancellazione dei documenti, nonché il monitoraggio e la gestione del cluster ElasticSearch. Queste API sono basate su richieste HTTP e supportano JSON come formato di scambio dati.

- Kibana Query API: API di interrogazione utilizzate da Kibana che consentono agli sviluppatori di recuperare dati aggregati e di eseguire query sui dati indicizzati in ElasticSearch. Principalmente sono utilizzate per personalizzare le dashboard e creare visualizzazioni personalizzate.
- Elasticsearch SQL: linguaggio SQL-like per eseguire query sui dati indicizzati. Queste API consentono agli sviluppatori di utilizzare query SQL per interrogare ElasticSearch, semplificando l'interazione per coloro che sono già familiari con SQL rispetto alla sintassi nativa di ElasticSearch.
- Elasticsearch Query DSL: linguaggio di query nativo di Elasticsearch denominato Query DSL (Domain Specific Language), che consente agli sviluppatori di creare query complesse in formato JSON per recuperare dati in modo flessibile e potente.
- Client Libraries: librerie client disponibili per diversi linguaggi di programmazione come Java, Python, JavaScript, Ruby, PHP e molti altri.

Per il raggiungimento degli obiettivi del progetto trattato vengono impiegate in particolar modo le REST API e le Query DSL [Elaf]. Quest'ultime, sono ciò che rende possibile l'interrogazione degli indici di Elasticsearch ed essendo espresse in formato JSON assumono una struttura del tipo:

```
{
  "query": {
    "tipo_di_query": {
      "parametri_query": "valori"
    }
  }
}
```

Esistono diversi tipi di query ed ognuno di essi dispone di parametri specifici:

- `match`: utilizzata per cercare un termine specifico in uno o più campi di testo.
- `term`: utilizzata per cercare un termine esatto in un campo specifico.
- `bool`: consente di combinare più query in un'unica query logica utilizzando operatori booleani come `must`, `must_not`, `should`.
- `range`: La query range viene utilizzata per cercare valori all'interno di un intervallo specificato.
- `wildcard`: La query wildcard consente di eseguire una ricerca basata su wildcards.

Di seguito un esempio esplicativo di query DSL che mostra l'utilizzo di tutte le opzioni citate. La query mostrata restituisce tutti i documenti dell'indice con campo "titolo" corrispondente a "Elasticsearch", campo "autore" contenente la parola "John" o la parola "Doe", campo "anno_pubblicazione" compreso tra il 2010 e il 2020 campo "descrizione" contenente una parola che inizia con "test".

```
{
  "query": {
    "bool": {
      "must": [
        { "term": { "titolo": "ElasticSearch" } },
        {
          "bool": {
            "should": [
              { "match": { "autore": "John" } },
              { "match": { "autore": "Doe" } }
            ]
          }
        }
      ],
      "filter": [
        {
          "range": {
            "anno_pubblicazione": {
              "gte": "2010",
              "lte": "2020"
            }
          }
        },
        {
          "wildcard": {
            "descrizione": "test*"
          }
        }
      ]
    }
  }
}
```

2.3 Precondizioni

L'architettura proposta come soluzione alle necessità del cliente è stata sviluppata sulla base di alcune specifiche fornite da quest'ultimo; in particolare, per quanto concerne questo capitolo, stiamo parlando di specifiche relative alle funzionalità di rete dei macchinari, ed al modo in cui i log degli eventi vengono registrati. In fase di analisi è infatti emerso che i macchinari prodotti non dispongono di capacità di rete avanzate, come per esempio l'invio di dati tramite protocollo HTTP, bensì sono solamente in grado di registrare log in un database MySQL. Quest'ultimo poi può eventualmente essere situato in localhost o in un host connesso al macchinario tramite la porta di rete a disposizione. Sulla base di queste specifiche è stata ideata l'architettura di rete, ponendo particolare attenzione al modo con cui recuperare i dati dai database dei macchinari situati nelle sedi del cliente.

2.4 Architettura proposta

Nella figura 2.1 è illustrata una schematizzazione della soluzione ideata, che mostra come l'architettura si divide in due parti, lato client (la sede del cliente) e lato server (il datacenter aziendale della New Assistent). A livello di ruoli, il client si occupa semplicemente di esporre in rete i dati registrati dai macchinari durante il loro lavoro, mentre il server si occupa di tutta una serie di attività, quali la raccolta dei dati dai macchinari, l'indicizzazione di quest'ultimi e l'hosting del portale web.

Sono inoltre evidenziati nell'immagine gli elementi cruciali dell'architettura: gli storage di memorizzazione, NetLink e Logstash. Tramite la collaborazione di questi elementi viene realizzato il flusso dei dati, che vengono prelevati dalla sede ed indicizzati lato server. In particolare, facendo riferimento alle specifiche dei macchinari esposte nella sezione 2.3, i dati risiedono su un database MySQL, al quale tutti i macchinari monitorati hanno accesso. Esso è a sua volta situato nel dispositivo NetLink, che a differenza dei macchinari dispone di tutte le capacità di rete necessarie ed è perfettamente in grado di esporre i dati all'esterno. Se infatti ogni macchinario registrasse localmente i dati, essi non sarebbero accessibili da remoto ma solamente da un operatore a bordo macchina.

E' bene ricordare che l'interfaccia web richiesta dal cliente si appoggia sull'Elastic Stack per le operazioni di ricerca e analisi, non consulta in maniera diretta il database in cui i macchinari scrivono, pertanto i dati registrati nel database di NetLink necessitano di essere prelevati ed indicizzati da Elasticsearch. Entra quindi in gioco Logstash, che svolge le operazioni di fetching dei dati connettendosi al database di NetLink, e reindirizza le informazioni che trova verso l'istanza di Elasticsearch. Per quanto riguarda invece le interfacce web, lo schema ne evidenzia due:

1. Il portale di monitoraggio personalizzato per i macchinari, che mostra le informazioni indicizzate da Elasticsearch sfruttando le API di ricerca messe a disposizione ed eseguendo query DSL per l'interrogazione come mostrato nella sezione 2.2.

2. Kibana, che come già menzionato nel capitolo 1, sezione 1.3.1, è parte integrante dell'Elastic Stack e, connettendosi ad Elasticsearch, ne consente la gestione degli indici, delle pipeline di ingestione dei dati e il monitoraggio.

3. Elasticsearch: integrazione dei macchinari

3.1 Connessione in rete dei macchinari CNC

La configurazione di rete ideata per il progetto prevede che gli indirizzi dei macchinari collegati a NetLink vengano assegnati in maniera dinamica, tramite DHCP.
Per DHCP (Dynamic Host Configuration Protocol) si intende un protocollo di rete utilizzato per assegnare automaticamente indirizzi IP e altre informazioni di configurazione di rete ai dispositivi (client) su una rete IP. A tal proposito, il dispositivo NetLink è dotato di due schede di rete, una con indirizzo statico, che svolge la funzione di DHCP server, ed una con indirizzo dinamico che funge da punto di accesso in rete per NetLink.

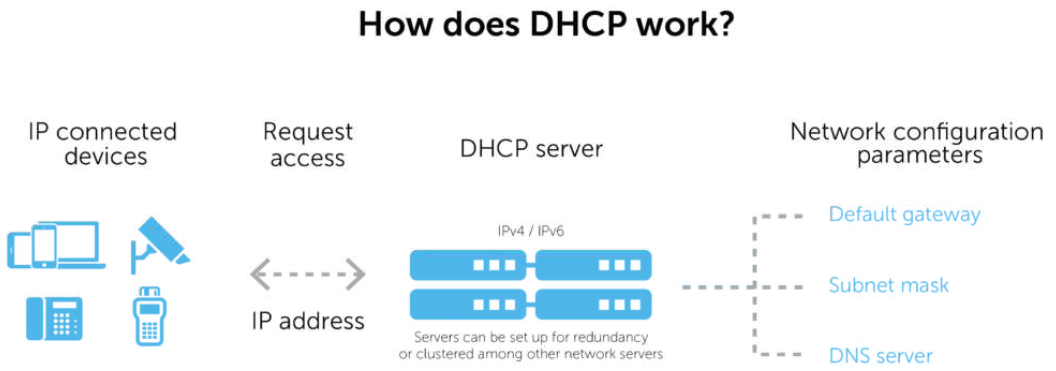


Figura 3.1: Schematizzazione DHCP

La figura 3.1 mostra una schematizzazione del processo di acquisizione dell'indirizzo dinamico da parte di un client, che in questo caso è un qualsiasi macchinario collegato. Tale processo si articola in quattro fasi principali, spesso riassunte con l'acronimo DORA [BLU]:

- Discover: Quando un dispositivo si connette alla rete per la prima volta, invia un messaggio chiamato DHCP Discover. Questo messaggio ha lo scopo di cercare un server DHCP disponibile. Il messaggio viene inviato a tutti i dispositivi della rete, chiedendo a qualsiasi server DHCP di rispondere.
- Offer: Un server DHCP risponde con un messaggio DHCP Offer, che offre un indirizzo IP al dispositivo richiedente. Questo messaggio contiene l'indirizzo IP

proposto, il tempo per cui l'indirizzo sarà valido (lease time), e altre informazioni di configurazione.

- Request: Il dispositivo, dopo aver ricevuto uno o più messaggi Offer, sceglie uno di questi e risponde con un messaggio DHCP Request. Questo messaggio informa tutti i server DHCP della sua scelta, in modo che gli altri server possano ritirare le loro offerte. In questo caso ovviamente di server ne esiste solo uno.

- Acknowledge: Il server DHCP selezionato conferma l'assegnazione dell'indirizzo IP con un messaggio DHCP Acknowledge. Questo messaggio conferma i dettagli della configurazione di rete, inclusi l'indirizzo IP e la durata del lease.

Ognuno dei macchinari fa riferimento al DHCP server all'indirizzo `192.168.250.1` per l'assegnazione del proprio indirizzo, il quale, a seguito dello svolgimento delle quattro fasi del processo di assegnazione descritte, offre ai dispositivi indirizzi in un range configurato tra `192.168.250.3` e `192.168.250.35`. Tale range di indirizzi implica un numero massimo di dispositivi collegabili per sede pari a 32, numero più che sufficiente a coprire qualsiasi scenario operativo del cliente; Quest'ultimo ha infatti specificato che al massimo dovrebbero essere collegati contemporaneamente una dozzina di macchinari.

3.2 Memorizzazione dei dati

Come già menzionato nel capitolo 2, i dati registrati dai macchinari dovrebbero essere memorizzati in un database relazionale, situato nel dispositivo NetLink, e successivamente indicizzati da Elasticsearch. A tal proposito, il cliente committente ha fornito un tracciato del database aziendale, già esistente, da utilizzare per la registrazione dei log dei macchinari, che è qui riportato.

```
+----------------------+
| DB                   |
+----------------------+
| cam                  |
| cutting              |
| cutting_utens        |
| errori_log           |
| errori_tip           |
| operatori            |
| today                |
| versione             |
+----------------------+
```

Come si può notare dall'illustrazione, la base di dati tiene traccia di diversi tipi di informazioni, come ad esempio i macchinari presenti in sede, con la tabella `cam`, gli operatori addetti alla supervisione di tali macchinari, con la tabella `operatori`, e gli errori registrati dai macchinari, con la tabella `errori_log`. In questo caso però, non tutti i dati contenuti nel database risultano utili agli scopi del progetto; Sono specificate di seguito infatti, solamente le strutture delle tabelle da tenere effettivamente in considerazione per l'implementazione del sistema.

```
// Tabella errori_log
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| ler_id       | int          | NO   | PRI | NULL    | auto_increment |
| ler_idcam    | int          | YES  |     | NULL    |                |
| ler_idope    | int          | YES  |     | NULL    |                |
| ler_idtrave  | int          | YES  |     | NULL    |                |
| ler_data     | datetime     | NO   |     | NULL    |                |
| ler_iderrtip | int          | YES  |     | NULL    |                |
| ler_desc     | varchar(255) | NO   |     | NULL    |                |
| letto        | int          | YES  |     | 0       |                |
+--------------+--------------+------+-----+---------+----------------+

// Tabella cam
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| cam_id           | int         | NO   | PRI | NULL    | auto_increment |
| cam_matricola    | varchar(21) | NO   |     | NULL    |                |
| cam_descrizione  | varchar(81) | YES  |     | NULL    |                |
| cam_tipo         | int         | YES  |     | NULL    |                |
| cam_consegna     | date        | YES  |     | NULL    |                |
| cam_soft_version | varchar(51) | YES  |     | NULL    |                |
| cam_soft_data    | date        | YES  |     | NULL    |                |
| cam_sysope       | varchar(81) | YES  |     | NULL    |                |
| cam_exec_program | int         | YES  |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

In riferimento ai tracciati riportati, i dati tenuti in considerazione per il monitoraggio sono l'ID dei log, la data di registrazione, la descrizione dell'evento, la matricola del macchinario che lo ha generato ed il suo modello (descrizione del macchinario).

Per quando riguarda invece la componente Elastic dell'implementazione, viene impiegata un'installazione di Elasticsearch lato server, tramite la quale è gestito l'indice dei log. Ovviamente, per quanto riguarda questo progetto nello specifico, viene gestito un singolo indice con impostazioni di default, ma nel caso in cui il sistema realizzato dovesse essere impiegato da più utenti sarebbe necessario gestire più indici in maniera distinta. Per ulteriori dettagli relativi a questo scenario d'uso consultare il capitolo 5. L'intero processo di installazione e configurazione delle componenti Elastic è inoltre consultabile nell'appendice A.

3.3 Fetching dei dati: Logstash

La scelta di impiegare Logstash per la realizzazione della pipeline di ingestione dati per Elasticsearch è stata ampiamente ponderata; Esistono infatti diversi altri modi per trasferire dati da una sorgente dati ad Elasticsearch, quali ad esempio i `Beats` ed i `Connectors`. I motivi per i quali queste alternative sono state scartate in favore di Logstash sono i seguenti:
Per quanto riguarda i `Beats`, come già menzionato nella sezione 1.2, essi sono componenti software che vanno installati lato client ed attivamente eseguono un push dei dati verso una destinazione. Nel contesto di questo progetto, il beat dovrebbe essere installato sul dispositivo NetLink; questa cosa richiederebbe però, in caso di multipli

clienti che usufruiscono di questo sistema, di avere diverse installazioni con configurazioni personalizzate, una per ogni cliente. D'altra parte, Logstash è invece in grado di gestire tramite una singola installazione multiple pipeline che svolgono operazioni di pull su diverse sorgenti, o anche più semplicemente gestire multiple sorgenti tramite una singola pipeline. Il tutto viene svolto operando lato server, quindi non risultano necessari interventi in sede da parte di operatori per operazioni di installazione o aggiornamento.

Per quanto concerne invece i `Connectors`, essi sono particolari componenti software, integrabili direttamente tramite l'interfaccia Kibana, che effettuano una sincronizzazione di dati tra una sorgente ed una destinazione. Naturalmente, come destinazione ci si riferisce ad una installazione Elasticsearch. Nello specifico, Elastic incentiva l'utilizzo dei connettori in casistiche in cui si fa già uso dell'Elastic Cloud, il loro servizio di cloud computing, per la gestione degli indici; Nonostante questo, è comunque possibile utilizzarli per sincronizzare un'installazione Elastisearch autonomamente gestita [Elab]. I motivi per i quali questa soluzione non risulta adeguata alla casistica di questo progetto però sono in parte i medesimi dei `Beats`, ovvero necessitano di essere configurati ed installati su ogni NetLink collegato. Inoltre, i connettori sincronizzano nell'indice elasticsearch TUTTI i dati della sorgente, in questo caso il database MySQL in cui i macchinari registrano informazioni, che però risultano necessari solamente in parte. Anzichè indicizzare dati superflui per gli scopi da realizzare, Logstash consente di effettuare il pull solamente dei dati necessari, eseguendo vere e proprie interrogazioni SQL che devono essere specificate nella configurazione della pipeline.

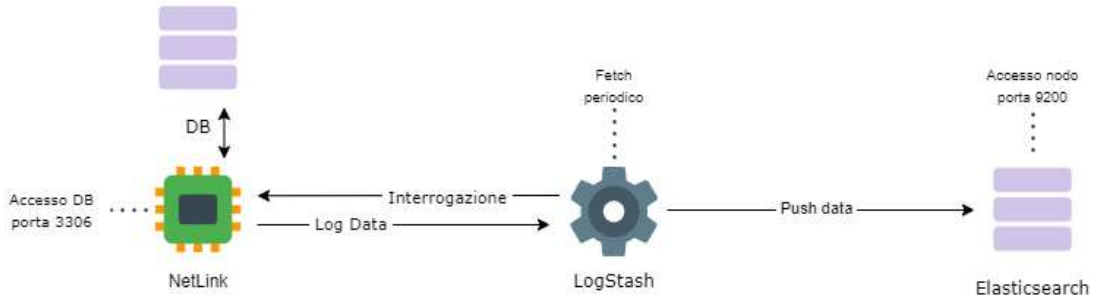


Figura 3.2: Pipeline di ingestione dati tramite Logstash

La figura 3.2 illustra in maniera schematica il funzionamento della pipeline realizzata tramite Logstash per l'ingestione dei dati in Elasticsearch. Il client NetLink ha come unico compito quello di esporre alla rete i dati contenuti nel database, allo stesso modo il nodo Elasticsearch deve semplicemente essere accessibile per ingerire dati. Logstash infine non dovrà fare altro che, periodicamente, eseguire da remoto un'interrogazione sul DB NetLink e trasferire i risultati verso Elasticsearch. All'occorrenza, Logstash consente anche di effettuare trasformazioni sui dati recuperati dalla sorgente, ma in questo caso non risultano necessarie, quindi i dati possono essere trasferiti direttamente per come si presentano.

Anche in questo caso, la configurazione di Logstash e della pipeline implementata sono consultabili nell'appendice A.3.1.

4. Web app: monitoraggio remoto dei macchinari

4.1 Precondizioni

Le specifiche fornite dal cliente committente, oltre che essere state altamente rilevanti per l'ideazione dell'architettura proposta nel capitolo 2, hanno avuto un impatto anche sulle scelte implementative del portale web richiesto. Per quanto riguarda le feature da implementare, le richieste avanzate dal cliente prevedono:

- Una dashboard che permetta la consultazione rapida dello stato attuale di tutti i macchinari connessi.
- La possibilità, per ogni macchinario, di consultare lo storico dei log registrati.
- La possibilità di effettuare ricerche a livello globale sui log, con eventualmente possibilità di filtrare i risultati per tipo di log, contenuto, matricola del macchinario e range di date.
- Una pagina che permetta la visualizzazione statistica dei dati registrati dai macchinari.

Come già accennato nella sezione 1.3.2 inoltre, il portale è stato implementato sfruttando un wrapper Bootstrap, AdminLTE, come framework grafico. Questa richiesta è motivata principalmente dalle necessità di integrazione del portale con la piattaforma web attualmente già in uso dal cliente, basata sul medesimo framework.

4.2 Utilizzo delle API

L'interazione tra il portale web ed il nodo Elasticsearch contenente le informazioni sui macchinari, alla base del funzionamento del portale, avviene tramite un backend implementato in linguaggio PHP, come già menzionato nella sezione 1.3.2, che sfrutta le API messe a disposizione da Elastic per recuperare le informazioni necessarie. Naturalmente, nel nostro caso, le API non vengono utilizzate in maniera intensiva, ma ne viene considerato solo un subset basilare per lo svolgimento delle operazioni necessarie a soddisfare le richieste del cliente. Nello specifico, quelle utilizzate per il nostro scopo sono le seguenti:

1. Stabilire una connessione con il nodo: la connessione viene stabilita tramite la funzione riportata nel blocco 4.2.

2. Recuperare log dal nodo: i blocchi 4.3 e 4.1 mostrano rispettivamente le funzioni che realizzano il recupero di log in maniera indiscriminata e tramite la specifica di parametri.

```
1
2     function get_logs($connection, $index, $params) {
3         $query = [
4             'bool' => [
5                 'must' => []
6             ]
7         ];
8
9         // Aggiungi i filtri in base ai parametri forniti
10        if (!empty($params['machine_id'])) {
11            $query['bool']['must'][] = [
12                'match' => [
13                    'machine_id' => $params['machine_id']
14                ]
15            ];
16        }
17        if (!empty($params['machine_description'])) {
18            $query['bool']['must'][] = [
19                'match' => [
20                    'machine_description' => $params['machine_description']
21                ]
22            ];
23        }
24        if (!empty($params['description'])) {
25            $query['bool']['must'][] = [
26                'match' => [
27                    'description' => $params['description']
28                ]
29            ];
30        }
31        if (!empty($params['start_date'])) {
32            $query['bool']['must'][] = [
33                'range' => [
34                    'data' => [
35                        'gte' => $params['start_date']
36                    ]
37                ]
38            ];
39        }
40        if (!empty($params['end_date'])) {
41            $query['bool']['must'][] = [
42                'range' => [
43                    'data' => [
44                        'lte' => $params['end_date']
45                    ]
46                ]
47            ];
48        }
49
50        $body = [
51            'query' => $query,
52            'sort' => [
53                'data' => ['order' => 'desc']
54            ],
55            'size' => 1000
56        ];
57
58        $searchParams = [
59            'index' => $index,
60            'body' => $body,
61        ];
62
63        return $connection->search($searchParams); }
```

Codice 4.1: Codice di raccolta parametrizzata dei log

```
1
2    function get_elastic_connection($hosts) {
3       return ClientBuilder::create()
4          ->setBasicAuthentication('elastic', 'uPxlPSaQU-XTbdGIl9QZ')
5          ->setHosts($hosts)
6          ->setSSLVerification(false)
7          ->build();
8    }
```

Codice 4.2: Codice di connessione al nodo Elasticsearch

```
1
2    function get_machines_logs($connection, $index) {
3       $params = [
4          'index' => $index,
5          'body' => [
6             'query' => [
7                'match_all' => new \stdClass(),
8             ],
9             'size' => 1000
10         ],
11      ];
12
13      return $connection->search($params);
14   }
```

Codice 4.3: Codice di raccolta non parametrizzata dei log

Entrambe le operazioni realizzate dai blocchi 4.3 e 4.1 eseguono a tutti gli effetti delle query di interrogazione verso il nodo Elasticsearch sul quale risiedono le informazioni. In particolare, nel blocco 4.1, il comportamento implementato non è altro che una manipolazione della query di ricerca da eseguire sul nodo che viene assemblata sulla base dei parametri specificati al momento della richiesta.

```
$params = [
    'index' => 'mysql_content',
    'body'  => [
        'query' => [
            'bool' => [
                'must' => [
                    'range' => [
                        'data' => [
                            'gte' => $params['start_date']
                        ]
                    ]
                ]
            ]
        ],
        'sort'  => [
            'data' => ['order' => 'desc']
        ],
        'size' => 1000
    ]
];
```

Esempio di query assemblata fornendo come parametro una data minima per la ricerca dei log. Tale query recupera i primi 1000 risultati, in ordine decrescente, che hanno al loro interno un campo date maggiore o uguale alla data minima specificata.

4.3 Dashboard

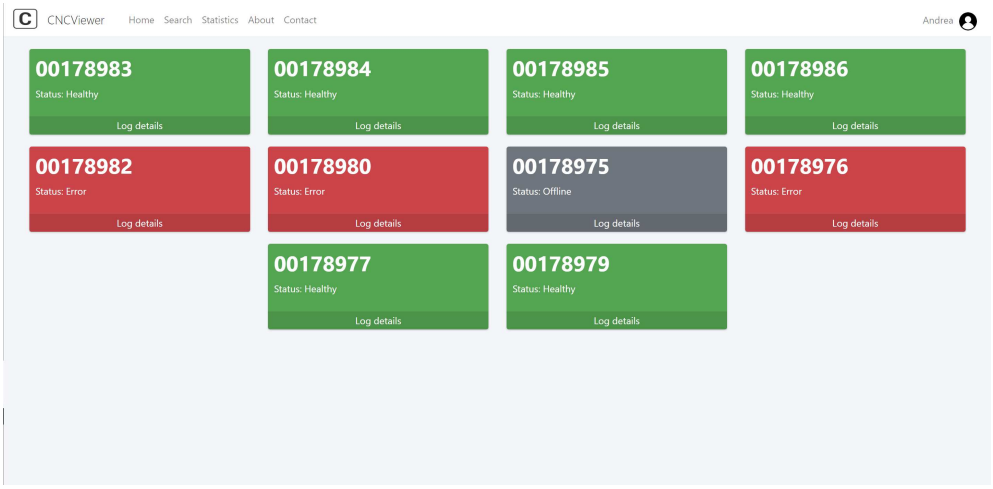


Figura 4.1: Dashboard principale di visualizzazione dei macchinari

La figura 4.1 mostra l'implementazione realizzata per la dashboard di consultazione dello stato dei macchinari, che costituisce la pagina di atterraggio una volta effettuato il login al portale. Per ognuno di essi si possono osservare uno fra quattro stati differenti:

- Healthy: il macchinario è in salute, attivo e funzionante.
- Warning: il macchinario è attivo e funzionante, ma potrebbe presentare alcuni problemi.
- Error: Il macchinario ha riscontrato un errore durante le lavorazioni ed al momento è bloccato in attesa di assistenza.
- Offline: Il macchinario non è al momento attivo. Un macchinario viene considerato offline quando non registra alcun log da almeno 60 minuti.

Ogni macchinario monitorato viene visualizzato tramite un layout tabellare, in cui per ognuno viene mostrata la matricola identificativa e lo stato attuale, attraverso colori intuitivi; Rispettivamente, verde, arancione, rosso e grigio. Gli stati mostrati inoltre, sono aggiornati in tempo reale tramite JavaScript, come si può vedere dalla porzione di codice riportata nel blocco 4.4; Essa rappresenta una funzione di controllo che viene eseguita periodicamente (sono stati concordati cinque secondi), la quale effettua, per ogni macchinario mostrato dall'interfaccia, una richiesta al backend della web app per verificare eventuali cambiamenti nello stato da visualizzare. Se vengono riscontrati cambiamenti, lo stato viene aggiornato.

Nel blocco 4.5 invece, è illustrato il codice backend di recupero dello stato per un particolare macchinario. Nello specifico, per controllare lo stato di un macchinario, si recupera l'ultimo log registrato dal macchinario con matricola corrispondente tramite le API elasticsearch, dopodichè si prende in considerazione la sua tipologia, ed, eventualmente, la sua data di registrazione; I log dei macchinari vengono registrati nel formato `<Type>: <Description>`.

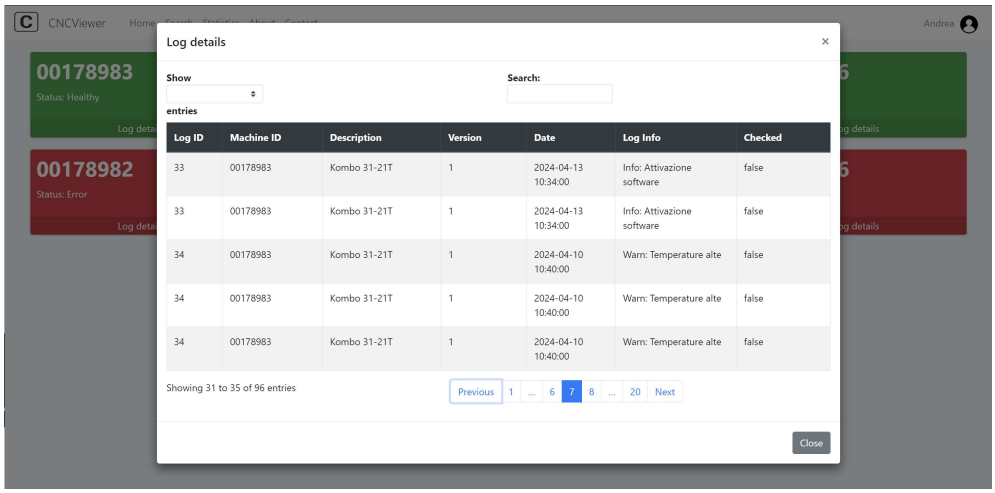


Figura 4.2: Storico dei log registrati per macchinario

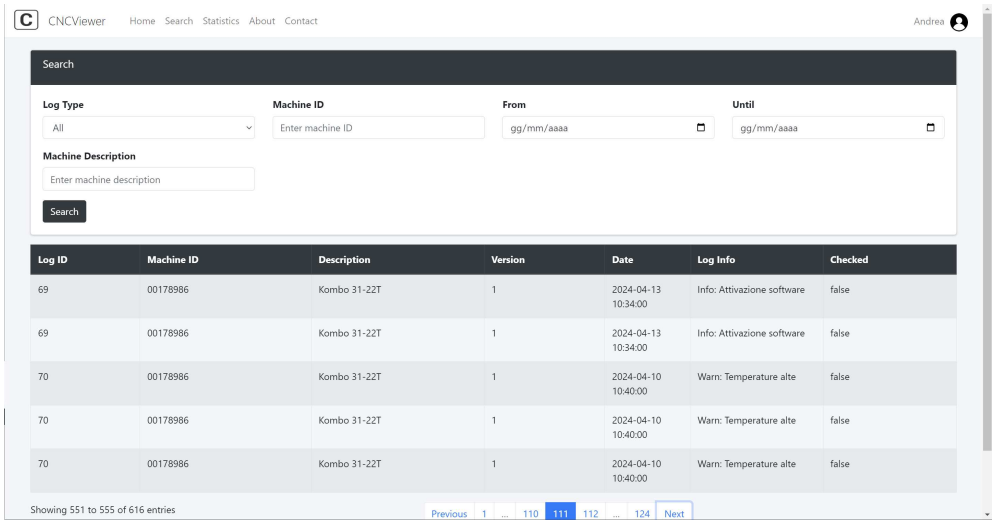


Figura 4.3: Pagina di ricerca globale dei log registrati

Tramite il medesimo layout tabellare menzionato, la dashboard permette la consultazione dello storico dei log di ogni macchinario clickando il pulsante `Log details` del macchinario interessato. Le informazioni vengono recuperate sfruttando le funzioni riportate nella sezione 4.2 e mostrate tramite una tabella paginata, per gestire grandi quantità di dati. Tale comportamento è illustrato dalla figura 4.2. La figura 4.3 invece illustra la pagina di ricerca globale dei log, che espone le informazioni tramite lo stesso layout tabellare ed è implementata seguendo la medesima logica di ricerca, in questo caso però tenendo in considerazione una parametrizzazione di quest'ultima.

4.4 Monitoraggio statistico

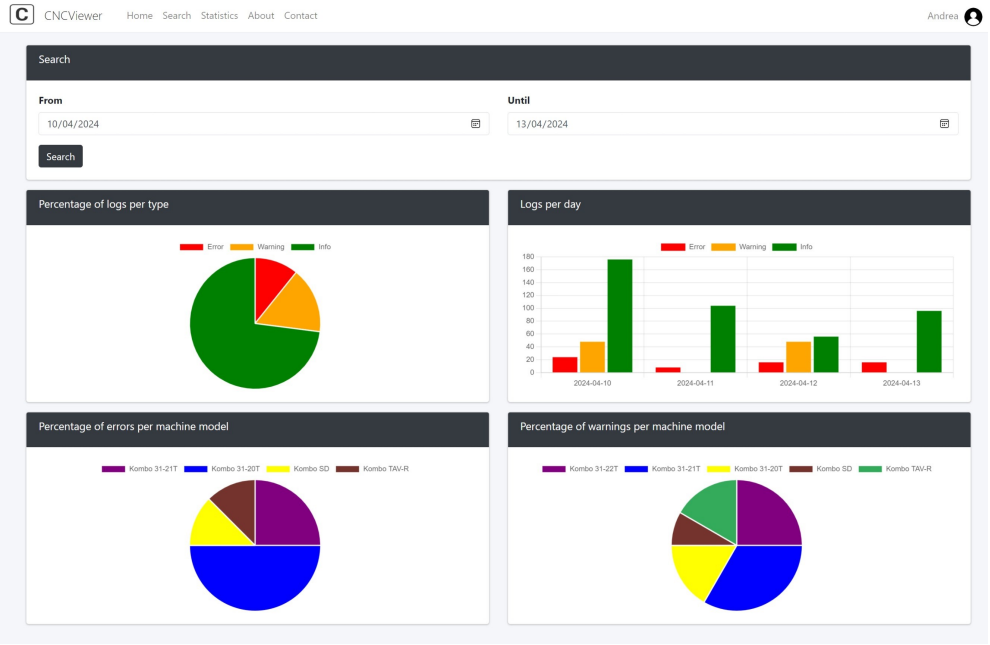


Figura 4.4: Pagina di visualizzazione statistica

L'analisi statistica delle informazioni registrate dai macchinari CNC monitorati risulta un punto molto importante tra le feature richieste dal cliente, in quanto essa può essere la base di future decisioni strategiche riguardanti la produzione. Il cliente ha specificatamente richiesto di mettere in evidenza eventuali problemi riscontrati dai macchinari, motivo per il quale i grafici implementati e visibili nella figura 4.4 mostrano informazioni percentuali riguardanti il numero di log totali registrati, il numero di log per tipo ed infine il numero di warning/errori per modello di macchinario, tramite una combinazione di diagrammi a torta ed istogrammi. E' inoltre possibile specificare l'insieme dei dati per i quali mostrare le statistiche specificando un range di date entro le quali considerare i log.
A livello implementativo, i grafici sono stati realizzati tramite il modulo `chart.js`[1], ed i dati recuperati tramite il codice mostrato dal blocco 4.1. Il filtro per data prevede semplicemente la specifica dei parametri `start_date` e `end_date` che vengono prelevati dagli input messi a disposizione all'utente dall'interfaccia.

[1] https://www.chartjs.org/

4.5 Codici

```
function check_statuses() {
   for(const element of document.getElementsByClassName("machine_box")) {
      const machineId = element.querySelector(".machine_id_heading").innerHTML;
      const currentStatus = element.querySelector(".machine_status_p").innerHTML;

      fetch_status(machineId)
         .then(function(response) {
            if(response != currentStatus) {
               update_status(element, response)
            }
         })
         .catch(function(error) {
            console.error(error);
         });
   }
}
```

Codice 4.4: Funzione di aggiornamento dello stato dei macchinari

```
$logs = get_logs(get_elastic_connection(['https://192.168.250.36:9200']),
            'mysql-content',
            $params);
$data = json_decode($logs, true);
$last_log_info = $data['hits']['hits'][0]['_source']['description'];
$last_log_date = $data['hits']['hits'][0]['_source']['data'];
$last_log_type = substr($last_log_info, 0, 4);
$status = '';

switch($last_log_type) {
   case 'Erro': $status = "Status: Error"; break;
   case 'Warn': $status = "Status: Alert"; break;
   case 'Info': $status = "Status: Healthy"; break;
   default: $status = "Status: Healthy";
}

$phpdateformat = strtotime( $last_log_date );
$last_log_date = date( 'Y-m-d H:i:s', $phpdateformat );

if ($last_log_date < date("Y-m-d H:i:s", strtotime("-1 day"))) {
   $status = 'Status: Offline';
}

echo $status;
```

Codice 4.5: Recupero dello stato dei macchinari

```
if(isset($_GET['machine_id'])) { $params['machine_id'] = $_GET['machine_id']; }
if(isset($_GET['description'])) { $params['description'] = $_GET['description']; }
if(isset($_GET['machine_description'])) { $params['machine_description'] =
$_GET['machine_description']; }
if(isset($_GET['start_date'])) { $params['start_date'] = $_GET['start_date']; }
if(isset($_GET['end_date'])) { $params['end_date'] = $_GET['end_date']; }

echo get_logs(get_elastic_connection(['https://192.168.250.36:9200']), 'mysql-
    content', $params);
```

Codice 4.6: Recupero dei log di dettaglio per specifico macchinario

5. Ottimizzazioni e miglioramenti

Il progetto svolto ed illustrato in questa tesi è basato su specifica richiesta di un singolo cliente committente. Risulta tuttavia possibile apportare alcuni cambiamenti ed ottimizzazioni in termini architetturali al sistema implementato che consentirebbero un suo impiego in un contesto con un'utenza più ampia ed un carico di dati da gestire maggiore.

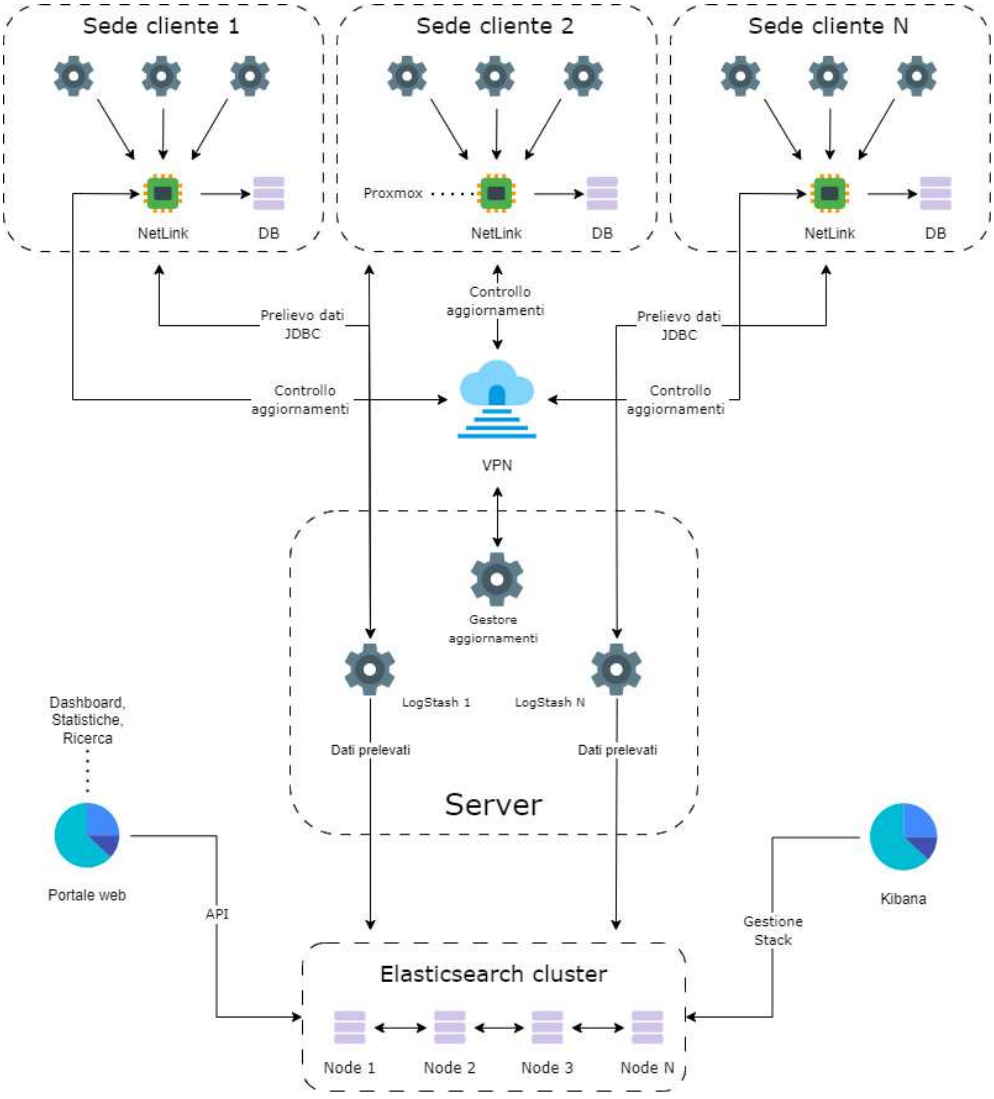


Figura 5.1: Schematizzazione dell'architettura ottimizzata

La figura 5.1 evidenzia infatti alcuni elementi che differiscono rispetto allo schema proposto nella figura 2.1:

- Elasticsearch cluster
- Multiple istanze Logstash
- Gestore aggiornamenti

5.1 Elasticsearch sharding

La realizzazione di un cluster Elasticsearch richiede la conoscenza e l'impiego del concetto di Sharding. Con sharding si intende il processo di suddivisione dei dati di un indice Elasticsearch in porzioni più piccole chiamate "shard". Ogni shard è un sottoinsieme indipendente dei dati dell'indice, che può essere distribuito in modo autonomo su nodi diversi realizzando un vero e proprio cluster Elasticsearch. Questo permette a Elasticsearch di distribuire e parallelizzare il carico di lavoro, migliorando le prestazioni e la scalabilità del sistema [SH]. I miglioramenti derivanti dalla distribuzione dei carichi di lavoro risultano particolarmente evidenti nel contesto dell'esecuzione di query, i cui tempi di esecuzione migliorano notevolmente grazie alla parallelizzazione dell'operazione su più nodi. Per gestire quantità di dati sempre maggiori infatti, basterebbe aggiungere nuovi nodi al cluster. In particolare, Elasticsearch supporta il ridimensionamento dinamico, il che significa che all'occorrenza è possibile aggiungere o rimuovere nodi, o anche redistribuire shard, senza dover interrompere il servizio. Tramite l'interfaccia Kibana, è inoltre possibile specificare il numero di shard di partenza in cui suddividere un indice al momento della sua creazione.

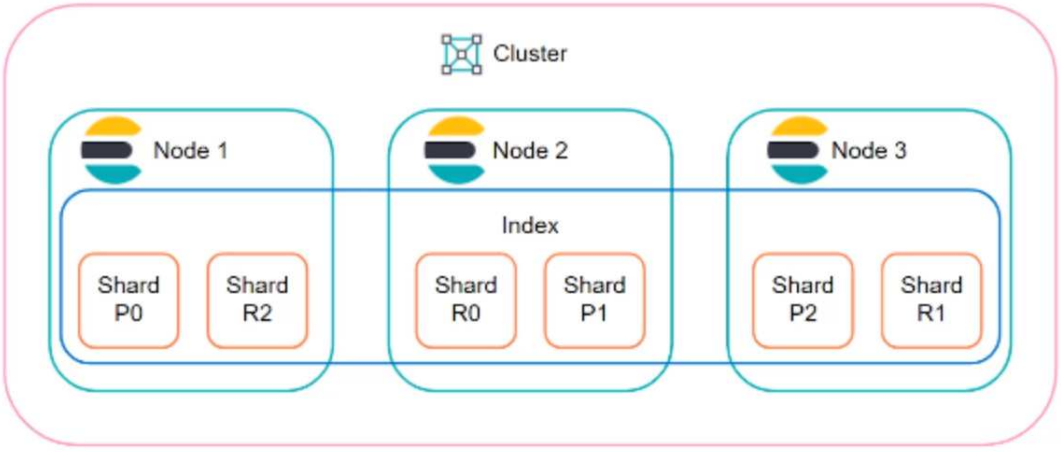


Figura 5.2: Sharding di un indice Elasticsearch [Gup22]

Lo sharding inoltre risulta una procedura estremamente importante e vantaggiosa in quanto andrebbe ad aumentare la tolleranza ai guasti del sistema: Distribuendo i dati su più nodi, ElasticSearch può tollerare la perdita di uno di essi, i quali potrebbero subire guasti o essere inaccessibili per qualche fattore esterno, senza compromettere l'integrità complessiva dei dati. Questo perchè, applicando lo sharding, i dati non solo vengono distribuiti nei nodi, ma anche replicati, come mostrato dalla figura 5.2. Se un nodo va offline quindi, i dati contenuti in esso possono essere ancora accessibili dagli

altri nodi che ospitano i corrispondenti shard replica. In generale, ogni nodo di un cluster Elasticsearch possiede uno shard primario, che viene poi replicato e distribuito a tutti gli altri nodi connessi.

5.2 Gestione dell'utenza e delle sorgenti dati

Il sistema descritto fino a questo punto gestisce i dati provenienti dai macchinari CNC monitorati tramite un singolo indice Elasticsearch; Tuttavia, in un contesto multi-utente con alti carichi di lavoro, una gestione di questo tipo risulterebbe altamente svantaggiosa. A livello di memoria e performance, a primo impatto, potrebbe non risultare un grande problema, in quanto anche gestendo grandi moli di dati con un singolo indice sarebbe possibile applicare lo sharding per parallelizzare il carico di lavoro. Processare i dati potrebbe tuttavia risultare in diverse difficoltà: non si avrebbe modo infatti di distinguere, per ognuno dei log presenti, a quale cliente appartiene il macchinario che lo ha registrato, a meno di non aggiungere ulteriori informazioni identificative nel momento in cui i dati vengono indicizzati. Si presenterebbero grandi problematiche anche in termini di sicurezza, in quanto un unico indice per tutti i clienti gestiti implicherebbe che ognuno di essi avrà indirettamente accesso ai dati degli altri, che potrebbero ad esempio essere recuperati tramite una query DSL, o in generale tramite l'uso delle API.

La soluzione ideale a questa situazione sarebbe la separazione dei dati tramite l'impiego di indici diversi, uno per ogni cliente che usufruisce del sistema. In questo modo, si avrebbe una corrispondenza uno ad uno fra i log indicizzati e chi li ha prodotti, senza dover appesantire l'indice con ulteriori informazioni, e si risolverebbe anche il problema dell'accessibilità ai dati. Sarebbe tuttavia necessario, da parte dell'admin del sistema, definire il grado di accesso che ogni cliente ha rispetto ai dati, attraverso la creazione di profili utente in Kibana. La procedura di creazione dei profili è la medesima mostrata nell'appendice A.5. I privilegi assegnati ad ogni cliente dovranno ovviamente essere congrui alle loro necessità per limitare allo stretto necessario l'accesso ad Elasticsearch.

Da questa suddivisione dei dati deriva la presenza di multiple istanze Logstash nell'immagine 5.1: Sebbene Logstash consenta tramite una singola installazione di gestire multiple pipeline contemporaneamente, quindi banalmente sarebbe in grado di prendere i dati da tutti i clienti ed inviarli nei rispettivi indici Elasticsearch, realizzare una centralizzazione di questo tipo potrebbe risultare svantaggioso sotto diversi punti di vista. In primo luogo, le prestazioni potrebbero risentirne in quanto il nodo Logstash in questione potrebbe essere messo sotto stress dal carico di lavoro da gestire, essendo i dati da indicizzare molti, frequentemente prodotti e avendo un numero di clienti potenzialmente indefinito. In secondo luogo, la centralizzazione del traffico dei dati potrebbe comportare ancora una volta rischi di sicurezza, in quanto la compromissione di un singolo nodo Logstash potrebbe portare ad un'interruzione del servizio per tutti i clienti connessi o addirittura ad una violazione di dati.

L'alternativa ideale proposta per evitare queste problematiche è appunto l'impiego di multiple installazioni Logstash, ognuna dedicata al trasporto di dati di un singolo cliente. In questo modo si otterrebbe una gestione dei dati dei singoli clienti nettamente separata, inoltre il carico di lavoro sarebbe parallelizzato su più agenti Logstash, i quali si ritroverebbero a gestire un numero ridotto di pipeline (una o più di una, in base al numero di sedi del cliente in questione). Gli agenti Logstash potrebbero inoltre risiedere su di uno stesso nodo fisico o disporre di un nodo dedicato.

5.3 Client autoprovisioning

L'ultimo aspetto in merito al quale il sistema potrebbe essere migliorabile è la sua flessibilità. Come già menzionato, il sistema di monitoraggio richiede, per ogni cliente che ne usufruisce, l'installazione in sede del dispositivo NetLink; In questo caso, ipotizziamo sia esso un mini-pc. In questo mini-pc sarà necessariamente presente un'installazione MySQL contenente il database aziendale del cliente sul quale vengono registrati i dati prodotti dai macchinari. Naturalmente, ogni cliente avrà a disposizione macchinari e database differenti; Questo rende indirettamente i dispositivi accoppiati all'utenza, in quanto, in caso di sostituzione o riutilizzo di un qualsiasi dispositivo NetLink in un contesto differente, sarebbe necessario per un tecnico riconfigurarlo manualmente.

La soluzione proposta per aumentare la flessibilità del sistema, che quindi consenta ai dispositivi di essere sostituiti o riutilizzati per clienti differenti in maniera semplice, è l'auto-provisioning. Con auto-provisioning si intende una procedura automatizzata di auto-aggiornamento dei dispositivi, che permetterebbe ad essi, al momento dell'accensione, di configurarsi in maniera autonoma per le attività da svolgere, verificando se sono disponibili aggiornamenti o cambiamenti di configurazione prima di iniziare il lavoro. In riferimento alla figura 5.1, l'impiego dell'auto-provisioning risulta nell'aggiunta di alcuni elementi allo schema:

- Gestore aggiornamenti: un agente software lato server che si dovrebbe occupare di verificare, su richiesta, la disponibilità di aggiornamenti per il NetLink richiedente, e restituendo in caso affermativo i relativi file.
- Tunnel VPN: a differenza di Elasticsearch, che sfrutta il protocollo HTTPS per tutelare gli accessi e le comunicazioni con Logstash, l'impiego di una VPN rappresenta una possibile soluzione di sicurezza per tutelare le comunicazioni tra i client NetLink e l'agente di aggiornamento, tramite la crittografia dei dati.
- Installazione Proxmox su NetLink: Proxmox è una piattaforma di virtualizzazione open-source che offre una vasta gamma di funzionalità per la creazione, gestione e monitoraggio di macchine virtuali (VM) e container. A tutti gli effetti trasforma il sistema sul quale è installato in un hypervisor bare metal [Wikb].

Per motivare la scelta di impiegare Proxmox nei dispositivi NetLink è necessario tenere a mente alcuni concetti: secondo l'implementazione illustrata nei precedenti capitoli di questa tesi, in particolare nei capitoli 3 e A, NetLink è configurato con un'installazione Debian 12, sulla quale come già detto risiede il database aziendale del cliente. Considerando l'ipotesi di avere clienti diversi con esigenze di monitoraggio diverse, si avranno necessariamente anche delle differenti basi di dati con cui lavorare, o addirittura diverse soluzioni e componenti software per il monitoraggio, ad esempio nel caso in cui per un cliente risulti più conveniente l'impiego dei Beats rispetto a Logstash per il recupero dei dati. In questo contesto, Proxmox risulta vantaggioso perchè, anzichè considerare il sistema Debian 12 come sistema principale, consente di trattarlo come una macchina virtuale. In questo modo, anzichè dover modificare i software installati o la base di dati manualmente quando si vuole riutilizzare un NetLink per un cliente differente, è possibile incapsulare le configurazioni predefinite del sistema di ogni cliente in dei file immagine (ISO), i quali rappresentano le macchine virtuali gestite dai vari NetLink tramite Proxmox. Tali immagini dovrebbero ovviamente essere memorizzate nel server, e recuperate all'occorrenza dal gestore aggiornamenti mostrato nell'immagine 5.1. In questo modo, riutilizzando i NetLink per una diversa clientela, al momento

dell'accensione viene effettuata una richiesta al server, protetta dal tunnel VPN, per verificare che l'attuale macchina virtuale gestita abbia una versione corrispondente a quella predisposta per l'attuale cliente al quale il NetLink in questione è assegnato. In caso affermativo, la macchina virtuale viene avviata normalmente, altrimenti il file immagine corretto e predisposto per l'attuale cliente viene restituito, e la macchina virtuale sostituita con una nuova costruita sulla base dell'immagine ricevuta.

6. Conclusioni

L'obiettivo posto inizialmente di realizzare un portale che sfruttasse lo stack Elastic è stato pienamente raggiunto, in particolare il portale web realizzato risulta perfettamente integrabile con quello già in uso del cliente committente. L'impiego dello stack Elastic per la realizzazione di questo progetto si è rivelata un'ottima scelta, sia per le performance ottenute durante il suo utilizzo che per la semplicità di installazione e configurazione delle componenti. La figura 6.1 mostra un estratto dei dati indicizzati tramite l'implementazione finale del sistema.

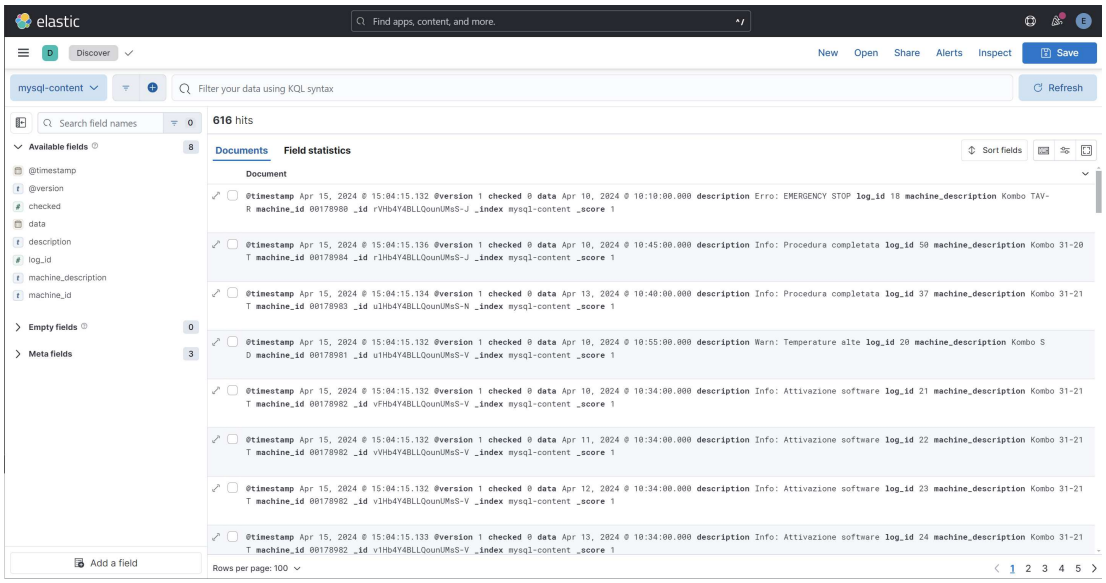


Figura 6.1: Indice risultante dall'elaborazione

Nonostante la soluzione presentata sia cucita sulle esigenze di un singolo committente, si può concludere che, apportando i miglioramenti illustrati nel capitolo 5, il sistema potrebbe risultare una valida soluzione di monitoraggio per tutti coloro che fanno uso di macchinari CNC. Questo risulta valido soprattutto se quest'ultimi non sono di recente fabbricazione, quindi come nella casistica attuale non dispongono di capacità di rete avanzate, e la loro sostituzione con modelli recenti risulerebbe troppo onerosa. Il sistema rappresenterebbe sicuramente, per la clientela descritta, un grande passo in avanti in termini di approccio all'industria 4.0.

A. Installazione ed utilizzo di Elastic stack

Questa appendice tratta il processo dettagliato di installazione dello stack Elastic in un ambiente Linux Debian 12, riportando anche comandi ed opzioni di configurazione impiegati specificatamente per il raggiungimento delle finalità del progetto. Il processo è sviluppato in tre sezioni distinte, una per ogni componente software dello stack. Sono trattate inoltre le modalità di utilizzo dell'interfaccia Kibana nel contesto della personalizzazione e della condivisione di dashboard di monitoraggio. Si tenga presente che ai fini di questo progetto ogni componente è stata installata all'interno di un unico sistema.

A.1 Installazione di Elasticsearch

Come primo passo, installare tutte le dipendenze richieste:

```
1. sudo apt-get install gnupg2
2. sudo apt-get install curl
3. wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
   sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
4. sudo apt-get install apt-transport-https
5. echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg]
   https://artifacts.elastic.co/packages/8.x/apt stable main" |
   sudo tee /etc/apt/sources.list.d/elastic-8.x.list
```

Successivamente, scaricare ed installare il pacchetto debian elasticsearch:

```
6. sudo apt-get update && sudo apt-get install elasticsearch

// Durante l'installazione, verranno generate automaticamente le
// credenziali dell'utente root di elastic
// Output:
The generated password for the elastic built-in superuser is :
uPxlPSaQU-XTbdGIl9QZ
```

A questo punto, elasticsearch dovrebbe essere perfettamente installato e funzionante. Può essere gestito come processo demone, utilizzando `systemctl`:

```
7. sudo systemctl daemon-reload
8. sudo systemctl enable elasticsearch.service
9. sudo systemctl start elasticsearch.service
```

In questo modo, elasticsearch verrà eseguito e sarà registrato come processo di avvio. Il suo corretto funzionamento può essere verificato con i seguenti comandi:

```
// Controllare la configurazione predefinita
10. grep -Ev '^#|^$' /etc/elasticsearch/elasticsearch.yml
// Controllare lo stato del processo
11. sudo systemctl status elasticsearch.service
// Controllare se la porta 9200 dedicata è aperta
12. ss -altnp | grep 9200
// Controllare se l'endpoint elasticsearch è raggiungibile
13. curl https://192.168.250.36:9200 --cacert
   /etc/elasticsearch/certs/http_ca.crt -u elastic
```

A.2 Installazione di Kibana

Le dipendenze di cui necessita Kibana, se l'installazione dei componenti dello stack viene seguita nell'ordine in cui è presentata, saranno già state installate precedentemente. Si può quindi procedere con l'installazione del pacchetto debian:

```
1. sudo apt-get update && sudo apt-get install kibana
```

Successivamente, modificare il file di configurazione di Kibana impostando l'indirizzo tramite il quale accedervi. E' possibile specificare `localhost`, ma se si vuole accedere all'interfaccia web in maniera remota l'indirizzo impostato dovrà necessariamente essere non di loopback; In questo caso è `192.168.250.36`:

```
// server.host: "192.168.250.36"
2. sudo vi /etc/kibana/kibana.yml
```

Configurare la sicurezza generando un token di collegamento per elasticsearch, da usare successivamente, e delle chiavi per la crittografia:

```
3. /usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token
   -s kibana

   // Output:
   eyJ2ZXIiOiI4LjEzLjAiLCJhZHIiOlsiMTkyLjE2OC4yNTAuMzY6OTIwMCJdLCJmZ3Ii
   OiwNGJlZmUwYWE4MDgyMzQ2MmJiYjQ3NDA4MDhjYjRhMGNlYzZhZDI5OWIwNWQ5N2YxO
   WUzYgzYzIzNDZhZWM4Iiwia2V5IjoiQnltdWhJNEJlamZhalVyUHMtU3I6WXNsRGZDeX
   VUVFM1TWF3TjZOTWlrZyJ9

4. /usr/share/kibana/bin/kibana-encryption-keys generate

   // Output:
   xpack.encryptedSavedObjects.encryptionKey:
   8ee2c221ffcc159e0c5e56d41e1f007a
   xpack.reporting.encryptionKey: 27e5b17c10db3f27a8f7f3a98cfa6b01
   xpack.security.encryptionKey: e2160ce631e378d71aa79e3d59c9de16
```

```
5. echo -e "xpack.encryptedSavedObjects.encryptionKey:
   8ee2c221ffcc159e0c5e56d41e1f007a
   xpack.reporting.encryptionKey: 27e5b17c10db3f27a8f7f3a98cfa6b01
   xpack.security.encryptionKey: e2160ce631e378d71aa79e3d59c9de16" >>
   /etc/kibana/kibana.yml
```

A questo punto, Kibana sarà pronta per essere avviata. I comandi sono i medesimi usati per elasticsearch:

```
6. sudo systemctl daemon-reload
7. sudo systemctl enable kibana.service
8. sudo systemctl start kibana.service
```

In questo caso però, sarà necessario anche controllare lo stato del processo:

```
// Go to http://192.168.250.36:5601/?code=411290 to get started.
9. sudo systemctl status kibana.service
```

Tale comando è necessario perchè essendo il primo avvio per l'interfaccia, è richiesta la registrazione della nuova installazione di Kibana attraverso il token generato inizialmente. In questo modo l'interfaccia sarà connessa e potrà accedere ai dati indicizzati da Elasticsearch. Si avrà in output un URL che conduce alla pagina di registrazione.

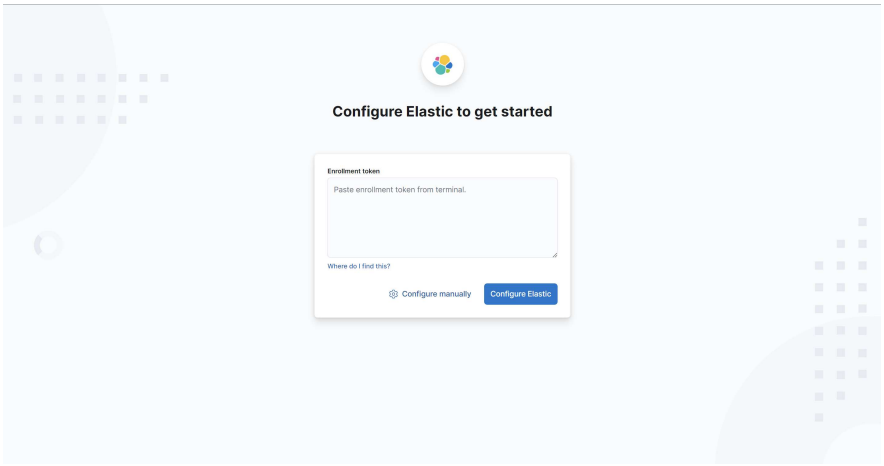


Figura A.1: Pagina Kibana di registrazione del token.

Incollando il token nell'apposito form e clickando "Configure Elastic", Kibana verrà collegata con l'istanza Elasticsearch che ha generato il token e sarà possibile effettuare il login per accedere a tutte le funzionalità messe a disposizione dall'interfaccia.

A.3 Installazione di Logstash

Anche per Logstash, in caso di una normale installazione, le dipendenze dovrebbero già essere risolte. Procedere quindi con l'installazione del pacchetto debian:

```
1. sudo apt-get update && sudo apt-get install logstash
```

Si può verificare sin da subito la correttezza dell'installazione svolta, tentando di eseguire Logstash ed utilizzando una configurazione della pipeline a riga di comando:

```
// i dati di input vengono presi dallo standard input, e vengono
// trasmessi allo standard output
2. sudo /usr/share/logstash/bin/logstash -e
   'input { stdin { } } output { stdout {} }'

/*  Fornendo degli input a terminale si dovrebbe ricevere un output
    di questo tipo:

    [INFO ] 2024-03-28 14:30:39.657 [Agent thread]
    agent - Pipelines running
    {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}

    test
    {
           "message" => "test",
          "@version" => "1",
        "@timestamp" => 2024-03-28T13:30:47.791651158Z,
             "event" => {
            "original" => "test"
        },
              "host" => {
            "hostname" => "ELK-CNC"
        }
    }
*/
```

Per eseguire correttamente Logstash ed iniziare il processo di ingestione dei dati in maniera autonoma però, è strettamente necessario specificare quale file di configurazione deve essere usato per la pipeline che si vuole realizzare. Logstash permette anche di gestire più pipeline diverse contemporaneamente, infatti di default tenta di importare un file di configurazione unico chiamato `pipelines.yml` dalla cartella `/usr/share/logstash/config/`, che è usato per specificare la posizione nel sistema delle configurazioni relative ad ognuna delle pipeline che dovrebbero essere realizzate

```
3. sudo mkdir /usr/share/logstash/config/
4. sudo vi /usr/share/logstash/config/pipelines.yml
```

Modificare il file incollando la seguente configurazione:

```
- pipeline.id: mysql-pipeline
  path.config: "/usr/share/logstash/config/<name-of-your-pipeline>.conf"
```

Il file creato specifica che la pipeline da gestire è soltanto una e che la sua configurazione si trova all'indirizzo impostato.
Successivamente, creare il file di configurazione effettivo:

```
5. sudo vi /usr/share/logstash/config/db-pipeline.conf

/* il file dovrebbe avere una struttura di questo tipo

   input {
        // custom input
   }
   filter {
        // custom filter
   }
   output {
        // custom output
   }

*/
```

Infine, conferire tutti i permessi alla cartella `/usr/share/logstash/` per la scrittura dei log e l'accesso ad i file di configurazione. A questo punto Logstash è pronto per essere eseguito. Avviare il demone con i medesimi comandi visti precedentemente (Logstash non dovrebbe essere avviato in modalità root):

```
6. sudo chmod -R 777 /usr/share/logstash/
7. sudo systemctl enable logstash
8. sudo systemctl start logstash
```

A.3.1 Ingestione di dati da database MySQL

Nel caso specifico di questo progetto, Logstash è stato configurato per recuperare dati da un database MySQL ed inviarli verso un'installazione Elasticsearch. Risulta necessario installare il plugin aggiuntivo JDBC, scaricandolo dal sito[1].

```
9. sudo dpkg -i PATH/TO/mysql-connector-j_8.3.0-1debian12_all.deb
```

Per permettere l'output dei dati verso Elasticsearch in maniera sicura sfruttando la verifica SSL, sono stati copiati localmente i certificati http generati durante l'installazione nella sezione A.1:

```
10. sudo mkdir /usr/share/logstash/certs/
11. sudo cp /etc/elasticsearch/certs/http_ca.crt
    /usr/share/logstash/certs/
```

La pipeline è stata configurata nel seguente modo, all'interno del file `db-pipeline.conf`:

```
input {
    jdbc {
     jdbc_driver_library =>
```

[1] https://dev.mysql.com/downloads/connector/j/

```
                    "/usr/share/java/mysql-connector-java-8.3.0.jar"
     jdbc_driver_class => "com.mysql.jdbc.Driver"
     jdbc_connection_string => "jdbc:mysql://127.0.0.1:3306/<db-name>"
     jdbc_user => "root"
     jdbc_password => "elkvm"
     jdbc_paging_enabled => true
     tracking_column => "data"
     use_column_value => true
     tracking_column_type => "timestamp"
     schedule => "*/5 * * * * *"
     statement => "SELECT
                     errori_log.ler_id AS log_id,
                     errori_log.ler_data AS data,
                     errori_log.ler_desc AS description,
                     errori_log.letto AS checked,
                     cam.cam_matricola AS machine_id,
                     cam.cam_descrizione AS machine_description
                     FROM errori_log
                     JOIN cam ON errori_log.ler_idcam = cam.cam_id
                     WHERE errori_log.ler_data > :sql_last_value"
   }
}
filter {
    # nessun filtro necessario
}
output {
    elasticsearch {
        hosts => [ "https://192.168.250.36:9200" ]
        index => "<index-name>"
        ssl_enabled => true
        cacert => "/usr/share/logstash/certs/http_ca.crt"
        user => "elastic"
        password => "uPxlPSaQU-XTbdGIl9QZ"
    }
}
```

A.4 Dashboard di monitoraggio

Avendo a disposizione uno Stack Elastic funzionante è possibile esplorare le diverse feature messe a disposizione da questa piattaforma, attraverso l'interfaccia web dedicata. Una fra le varie operazioni che Kibana rende possibile è la creazione di dashboard personalizzate per la visualizzazione dei dati indicizzati da Elasticsearch. Si prenda come esempio l'indice verso il quale vengono inviati i dati da Logstash, chiamato `mysql-content`. Per creare una dashboard dedicata alla visualizzazione di tali dati, partendo dalla home page di Kibana, navigare tramite il menù laterale alla sezione Analytics > Dashboards.

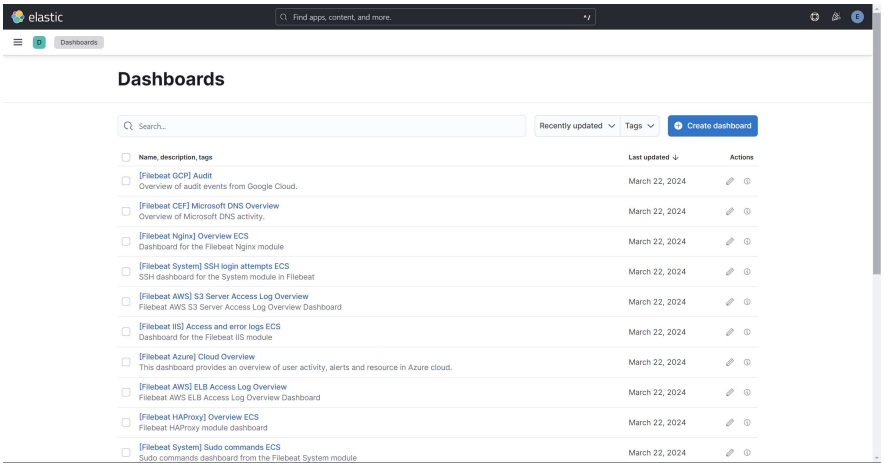


Figura A.2: Dashboards overview in Kibana

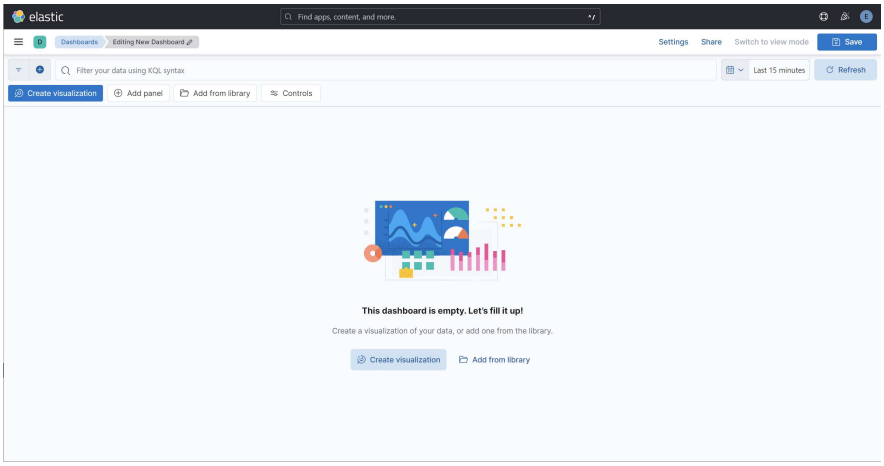


Figura A.3: Dashboard vuota

Nella pagina riportata dalla figura A.2 sono visualizzate tutte le dashboard esistenti, comprese quelle automaticamente generate. Per crearne una nuova, clickare `Create dashboard`. L'immagine A.3 rappresenta invece la nuova dashboard appena creata, la quale è momentaneamente vuota. Per aggiungere contenuti, è necessario creare delle "visualizzazioni", ovvero elementi visuali configurabili che rappresentano i dati di interesse che si vogliono, appunto, visualizzare; Una dashboard può contenere un numero indefinito di questi elementi. Per creare una nuova visualizzazione, clickare

`Create visualization`.

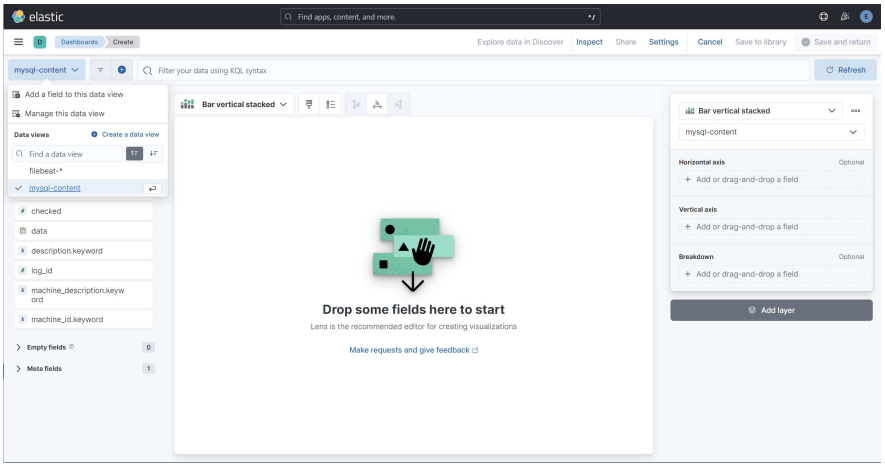


Figura A.4: Schermata di personalizzazione di una visualizzazione

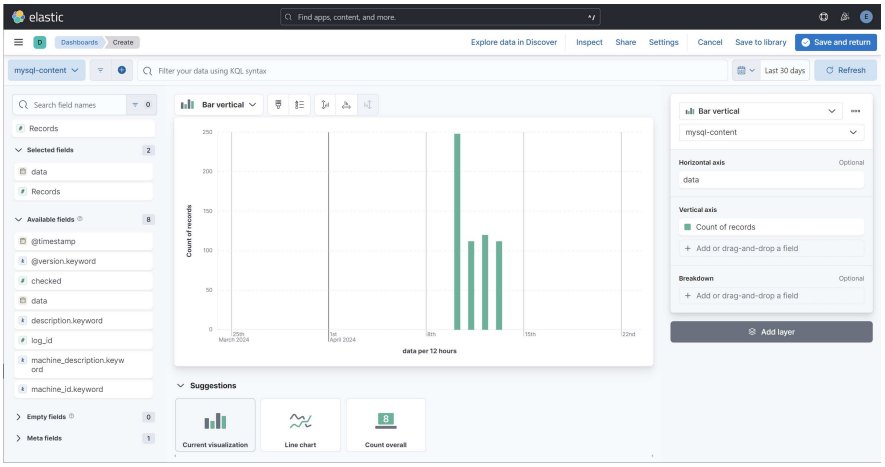


Figura A.5: Esempio di istogramma realizzato tramite Lens

Il passo successivo è la personalizzazione della visualizzazione. La figura A.4 mostra l'editor integrato in Kibana, che prende il nome di `Lens`, per l'editing dei contenuti. Esso permette una personalizzazione altamente intuitiva delle visualizzazioni, nel dettaglio:

- Nella zona nord-ovest, tramite l'apposito menù a tendina, è possibile specificare l'origine (indice), dei dati da visualizzare. Come esempio, viene preso l'indice `mysql-content`.
- Nella zona laterale sinistra, sono visualizzati tutti i campi che compongono i documenti dell'indice, e che è possibile visualizzare. Per aggiungere dei dati, basta semplicemente trascinare i campi interessati nella zona centrale della pagina, dove viene mostrata una preview della visualizzazione.
- Nella zona superiore rispetto all'area di preview è possibile specificare, sempre attraverso un menù a tendina dedicato, la tipologia di visualizzazione che si vuole applicare, che varia da una semplice metrica numerica a grafici o tabelle.

- Nella zona laterale destra infine, sono configurabili alcuni parametri relativi alla specifica tipologia di visualizzazione selezionata.

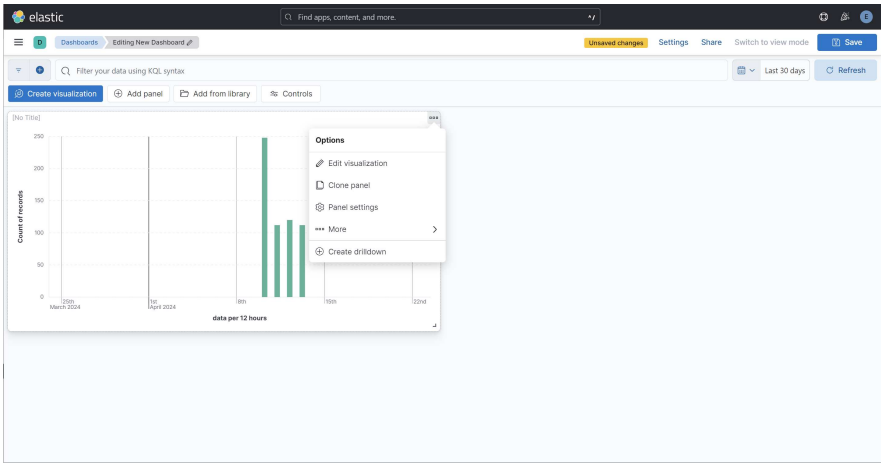


Figura A.6: Esempio di dashboard con visualizzazione

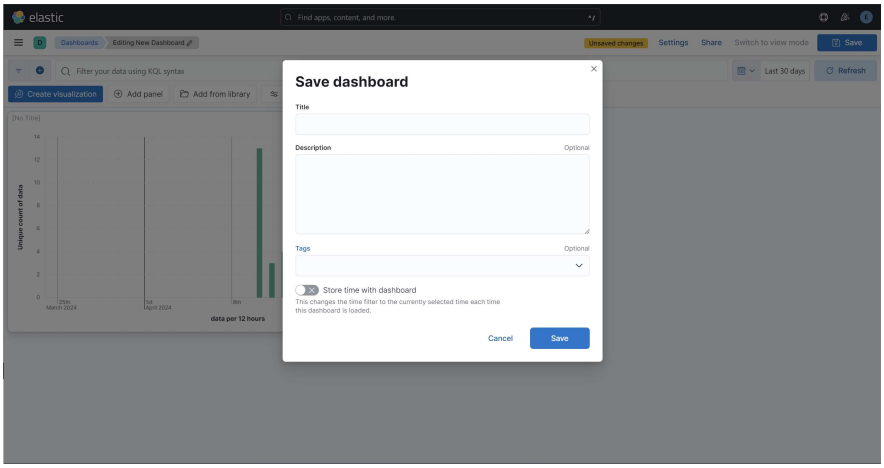


Figura A.7: Salvataggio di una nuova dashboard

La figura A.5 è un esempio molto semplice di visualizzazione realizzabile tramite `Lens`, dove è stato realizzato un istogramma verticale che mostra il numero di log registrati per data, negli ultimi 30 giorni. Per salvare una visualizzazione ed aggiungerla alla dashboard, clickare `Save and return`.
Una volta salvata la visualizzazione, questa comparirà all'interno della dashboard, come mostrato dalla figura A.6, dove sono possibili ulteriori modifiche, questa volta in termini di posizione e ridimensionamento. Per concludere la creazione della dashboard, clickare `Save` ed inserire le informazioni richieste, come mostrato nella figura A.7.
A questo punto la dashboard personalizzata è stata creata con successo, ed è consultabile, come menzionato all'inizio di questa sezione, in Analytics > Dashboards.

A.5 Configurazione dei profili utente

Un'altra importante feature che Kibana mette a disposizione è la gestione dell'utenza, ovvero la creazione di profili utente che definiscono i soggetti in grado di accedere alla piattaforma Elastic, ed i loro relativi ruoli. Di base, durante l'installazione di Elasticsearch, viene creato un singolo profilo utente, con username `elastic` e ruolo superuser. Per creare ulteriori profili, occorre navigare, tramite il menù laterale nella home page Kibana, verso Management > Stack Management > Security > Users.

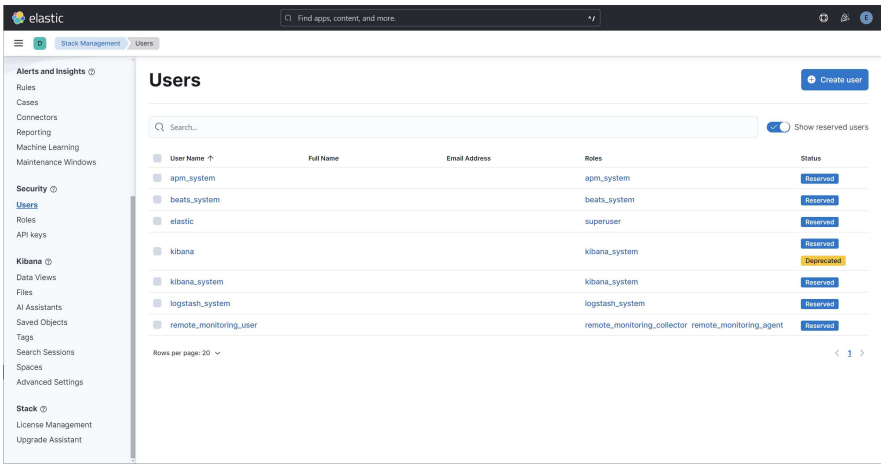


Figura A.8: Users overview in Kibana

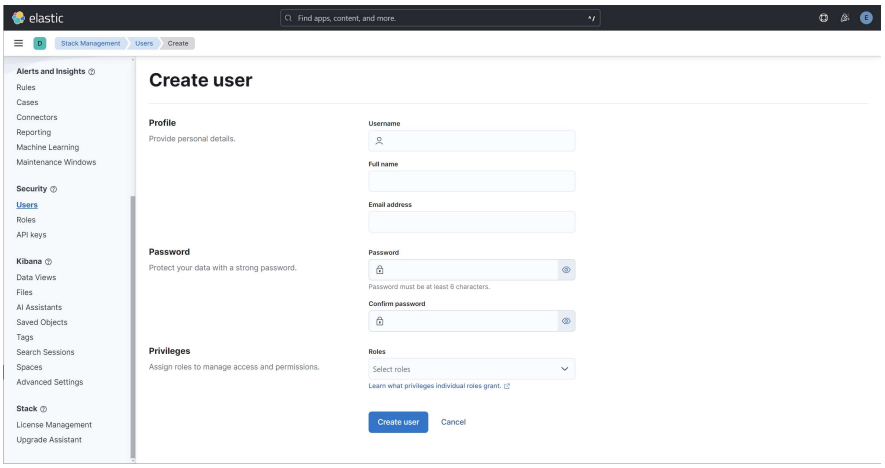


Figura A.9: Creazione di un profilo utente in Kibana

Nella pagina riportata dalla figura A.8 sono visualizzati tutti i profili utente attualmente esistenti, dei quale è possibile visualizzare i dettagli e modificare parametri semplicemente clickando lo username mostrato. Per creare invece un nuovo profilo utente, clickare `Create user`. Sarà necessario inserire tutte le informazioni richieste nel form riportato dalla figura A.9, o almeno quelle essenziali quali username, password e ruolo, e clickare nuovamente `Create user`. In particolare, per quanto riguarda quest'ultima opzione, si hanno diverse alternative, ognuna delle quali garantisce all'utente creato la possibilità di svolgere determinate operazioni all'interno della piattaforma Elastic.

Un utente può anche essere configurato per avere più di un ruolo. Se ne distinguono quattro categorie:

- User roles: ruoli progettati per gli utenti finali o gli operatori che devono interagire con i dati e le visualizzazioni all'interno di Kibana. Gli utenti a cui viene assegnata questa cateogria di ruoli possono accedere alle funzionalità specifiche di Kibana assegnate loro dall'amministratore, come, ad esempio, la visualizzazione di dashboard.
- Admin roles: ruoli destinati agli amministratori di sistema o agli utenti con responsabilità di gestione dell'ambiente Kibana ed Elasticsearch. Gli utenti con ruoli amministrativi hanno accesso completo a tutte le funzionalità di Kibana, inclusa la gestione degli utenti, dei ruoli, degli indici e delle impostazioni globali del sistema.
- Deprecated roles: ruoli identificati come obsoleti e che non vengono più raccomandati per l'uso. Si tratta di ruoli precedentemente utilizzati o implementati in versioni precedenti di Kibana che sono stati sostituiti o rimossi in versioni recenti.
- System roles: ruoli predefiniti e integrati nel sistema. Questa categoria include ruoli come "superuser" o "kibana-system", che forniscono privilegi amministrativi a livello di sistema per l'accesso e la gestione dell'ambiente Kibana ed Elasticsearch.

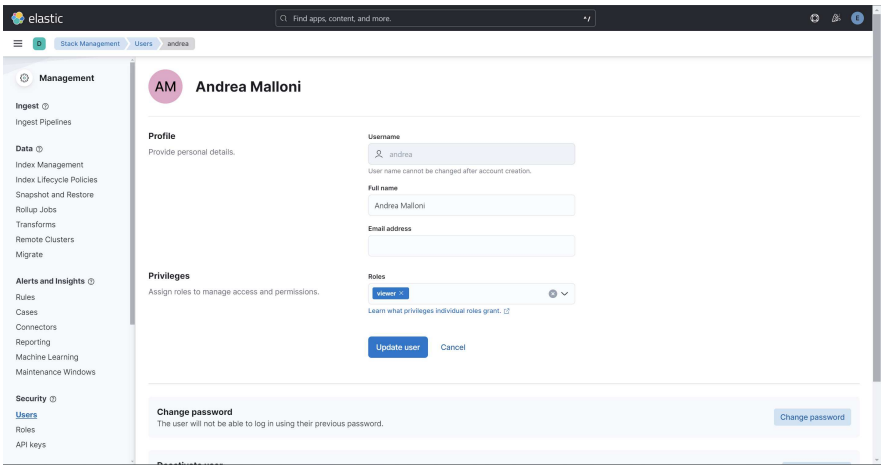


Figura A.10: Esempio di utente con ruolo di visualizzazione

La figura A.10 mostra un esempio di utente creato con privilegi `viewer`. Ciò significa che, prendendo in riferimento la figura A.6 e condividendo la dashboard con questo utente tramite l'opzione `share`, quest'ultimo sarà tranquillamente in grado di visualizzare i dati e la dashboard nel suo complesso, ma non avrà la possibilità ad esempio di modificarla, garantita invece dal ruolo `editor`.

Ringraziamenti

Nella stesura di questa tesi, è stato per me fondamentale il supporto di tante persone, senza le quali il mio percorso sarebbe stato sicuramente più difficile e che ritengo doveroso ringraziare in questo spazio conclusivo.

Ringrazio in primo luogo i miei genitori e mio fratello, che durante questo percorso universitario mi hanno costantemente sostenuto e spronato, dandomi la motivazione per rimanere sempre costante negli studi e dimostrare a me stesso ciò di cui sono realmente capace, cosa che nei momenti difficili può facilmente essere dimenticata.

Ringrazio poi tutti i miei amici e colleghi, con i quali ho condiviso momenti alti e bassi e che mi hanno accompagnato e supportato in questi anni. Sono stati fonte di confronto e di crescita sia didattica che personale, e di questo sono infinitamente grato, non sarei la persona che sono senza di loro.

Ringrazio infine il mio relatore, il professor Fausto Marcantoni, senza il quale la stesura di questa tesi non sarebbe stata possibile. Lo ringrazio per la sua infinita disponibilità e pazienza, oltre che per i consigli e le conoscenze che mi hanno accompagnato durante la stesura e che mi accompagneranno durante la vita lavorativa.

Bibliografia

[BLU] BLUECAT. *What is DHCP? It assigns addresses dynamically.* URL: https://bluecatnetworks.com/glossary/what-is-dhcp/.

[Elaa] Elastic. *Beats reference.* URL: https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html.

[Elab] Elastic. *Elastic connectors.* URL: https://www.elastic.co/guide/en/enterprise-search/current/connectors.html.

[Elac] Elastic. *Elasticsearch introduction.* URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.

[Elad] Elastic. *Kibana introduction.* URL: https://www.elastic.co/guide/en/kibana/current/introduction.html.

[Elae] Elastic. *Logstash introduction.* URL: https://www.elastic.co/guide/en/logstash/current/introduction.html.

[Elaf] Elastic. *Query DSL reference.* URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html.

[Elag] Elastic. *REST API.* URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html.

[Gee] GeeksForGeeks. *What is Elastic Stack?* URL: https://www.geeksforgeeks.org/what-is-elastic-stack-and-elasticsearch/.

[Gup22] Nidhi Gupta. *Primary shards and Replica shards in Elasticsearch.* 2022. URL: https://nidhig631.medium.com/primary-shards-replica-shards-in-elasticsearch-269343324f86.

[SAP] SAP. *Industria 4.0.* URL: https://www.sap.com/italy/products/scm/industry-4-0/what-is-industry-4-0.html.

[SH] Thiago dos Santos Hora. *Shards and Replicas in Elasticsearch.* URL: https://www.baeldung.com/java-shards-replicas-elasticsearch.

[Wika] Wikipedia. *Macchina a controllo numerico.* URL: https://it.wikipedia.org/wiki/Macchina_a_controllo_numerico.

[Wikb] Wikipedia. *Proxmox Virtual Environment.* URL: https://en.wikipedia.org/wiki/Proxmox_Virtual_Environment.