RdS
RICERCA DI SISTEMA

ENEA
AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE, L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

MINISTERO DELL'AMBIENTE E DELLA SICUREZZA ENERGETICA

Ricerca di Sistema elettrico

Realizzazione e implementazione su un caso d'uso di una piattaforma di stream analytics e implementazione delle logiche di training e inference dei modelli di ML e Artificial Intelligence su acceleratori di varia natura (FPGA/GPU)

Angelo Mariano, Serena D'Onofrio

ENEA

RdS_PTR 22-24_PR 2.1_LA3.7_141

Realizzazione e implementazione su un caso d'uso di una piattaforma di stream analytics e implementazione delle logiche di training e inference dei modelli di ML e Artificial Intelligence su acceleratori di varia natura (FPGA/GPU)

A. Mariano (ENEA), S. D'Onofrio (ENEA)

Dicembre 2024

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dell'Ambiente e della Sicurezza Energetica -ENEA Piano Triennale di Realizzazione 2022-2024

Obiettivo 2: Digitalizzazione ed evoluzione delle reti

Progetto 2.1: Cybersecurity dei Sistemi Energetici

Linea di attività: 3.7

Responsabile del Progetto: Maria Valenti (ENEA)

Responsabile Linea di Attività: A. Mariano (ENEA)

Mese inizio previsto: luglio 2023

Mese inizio effettivo: luglio 2023

Mese fine previsto: dicembre 2024

Mese fine effettivo: dicembre 2024

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione: Si ringrazia per la collaborazione alle attività svolte

Indice

Indice delle figure

1 Risultati attesi

Il problema della cybersecurity delle reti elettriche si riferisce alle misure adottate per proteggere le infrastrutture elettriche, come le reti di trasmissione e di distribuzione, dalle minacce informatiche. Si tratta di un problema critico in quanto le reti elettriche sono infrastrutture strategiche di una nazione, e una loro interruzione o compromissione potrebbe avere gravi conseguenze tecniche ed economiche.

Lo scopo di questa attività è quello di ridurre la vulnerabilità ai cyber attacchi nei contesti considerati avvalendosi delle funzionalità della piattaforma progettata nella LA3.3.

Tra i risultati attesi ci sono:

- Realizzazione di una piattaforma che utilizzi componenti HW/SW acquistati nel progetto ed integrati con sistemi già esistenti
- Realizzazione di una dashboard di gestione facile ed intuitiva, che consenta di monitorare lo stato dei component della piattaforma e di monitorare ed inserire nuovi task
- Realizzazione di una piattaforma di stream analytics che consenta di estrarre i dati da sorgenti certificate e di applicare sui dati una serie di algoritmi di ML specializzati per la rilevazione di anomalie e di comportamenti inusuali che possono comportare rischi per la sicurezza.

Per lo svolgimento della LA, era inizialmente prevista l'acquisizione delle competenze di una società esperta in programmazione avanzata di acceleratori hardware e dei loro ambienti di gestione per l'acquisizione dati per la realizzazione di componenti ad hoc del sistema integrato.

2 Risultati ottenuti

Nell'ambito dell'attività in oggetto sono stati perseguiti e raggiunti i seguenti risultati:

- Realizzazione di un sistema IT integrato basato su container e sulla virtualizzazione delle risorse.
- Realizzazione di una piattaforma per lo stream processing dei messaggi provenienti da diverse sorgenti di dati di comunicazione in un'ottica producer-consumer.
- Realizzazione di una piattaforma per l'analisi dei dati con algoritmi di machine learning e intelligenza artificiale.
- Realizzazione di una dashboard per il monitoraggio e la gestione dei componenti del sistema integrato di stream analytics.

2.1 Avanzamento della ricerca rispetto allo stato dell'arte internazionale con riferimento ai risultati ottenuti

Questo ambito di ricerca è innovativo rispetto allo stato dell'arte internazionale, in quanto i sistemi applicati in ambito di cybersicurezza non hanno una letteratura molto diffusa. Inoltre, tutti gli algoritmi pubblicati fino ad ora sono applicazioni di alcuni modelli a dei dataset di

esempio costruiti ad hoc, che quindi non riflettono le esigenze di un sistema reale, in grado di rispondere in quasi real-time. Pertanto, l'attività di ricerca svolta ha un valore di innovazione in quanto applica differenti approcci di elaborazione a dati reali.

3 Prodotti attesi

Realizzazione di una piattaforma di stream analytics applicata alle trasmissioni di messaggi su reti dati, integrata con strumenti di data analytics e applicabile ad almeno un caso d'uso di algoritmo di machine learning e intelligenza artificiale.

4 Prodotti ottenuti

La piattaforma ottenuta nel suo complesso può essere concepita come un sistema integrato di più componenti: la componente di stream processing; la componente di data analytics (applicata a due casi d'uso diversi); la dashboard di monitoraggio e gestione.

Il sistema integrato si appoggia su un cluster Kubernetes[1]. Kubernetes è un sistema open source di orchestrazione di container, ovvero il processo che permette di automatizzare il deployment, la gestione, la scalabilità e il networking dei container. Kubernetes funziona tramite pod e nodi. Il pod è la risorsa che descrive l'unità elementare eseguibile su un nodo del cluster, gestisce uno o più container ed è molto efficace dal punto di vista della scalabilità orizzontale soprattutto alle applicazioni orientate ai microservizi. I nodi eseguono i pod e sono in genere raggruppati in un cluster Kubernetes, astraendo le risorse hardware fisiche sottostanti.

L'infrastruttura Hw/Sw nel suo complesso, associata alle tecniche di machine learning elaborate, consente di fornire un valore aggiunto in termini di sicurezza rispetto a possibili vulnerabilità delle reti smartgrid.

4.1 Componente di stream processing

La componente di stream processing è un sistema IT basato su una soluzione Apache Kafka[2]. Apache Kafka è una piattaforma distribuita open source di stream processing, le cui tre funzionalità principali sono: consentire alle applicazioni di pubblicare o sottoscrivere flussi di dati o eventi, archiviare i record accuratamente in modo tollerante agli errori e durevole, elaborare i record in tempo reale. Inoltre, Kafka ha una bassa latenza e un'alta velocità di gestione, e permette di analizzare in modo affidabile e persistente grandi volumi di dati che provengono da diverse sorgenti. Nell'implementazione del componente è stata realizzato un sistema scalabile composto da tre componenti denominati enea-kafka-controller-[0,1,2] che fungono da broker dei messaggi recepiti da diverse fonti. Questa componente è interrogabile

[1] https://kubernetes.io/it/

[2] https://kafka.apache.org/

da client diversi in modalità producer consumer, con i codici disponibili nel repository dedicato Gitlab[3] ed installati su uno dei pod del cluster Kubernetes.

4.2 Componente di data analytics

La componente di data analytics è composta da due tipi di pod all'interno del sistema integrato: uno basato su Code-server[4] e uno basato su container Docker[5] python.

Code-server è un progetto open source che realizza una versione web-based di VS Code[6] che permette di eseguire un workspace di esecuzione di modelli e di algoritmi su qualsiasi host e di accedervi tramite browser. Quindi, tramite il progetto Code-server, l'interfaccia utente di VS Code è trasformata in un servizio web, consentendo di sviluppare senza dover installare dei componenti in locale ed inoltre il codice e gli algoritmi sono eseguiti direttamente sul server remoto. L'accessibilità via web di questo strumento software risulta particolarmente utile per progetti che richiedono risorse computazionali elevate, come l'apprendimento automatico o la compilazione di grandi progetti. Code-server supporta una vasta gamma di estensioni disponibili per VS Code e quindi permette di personalizzare lo spazio di lavoro installando i pacchetti più utili per data analytics, come quelli sviluppati con R[7].

Tramite Code-server è inoltre possibile collaborare in maniera simultanea con altri data scientist sullo stesso progetto, visualizzando le modifiche effettuate e chattando nell'editor di codice. Per le finalità del progetto, tutte le componenti di elaborazione che richiedono l'esecuzione di script di R sono state installate nell'ambiente di sviluppo di Code-server (come si può vedere nella Figura 1).

[3] https://gitlab.brindisi.enea.it/bigdataai/cyber/kafka-stream-processing

[4] https://github.com/coder/code-server

[5] https://www.docker.com/

[6] https://github.com/Microsoft/vscode

[7] https://www.r-project.org/

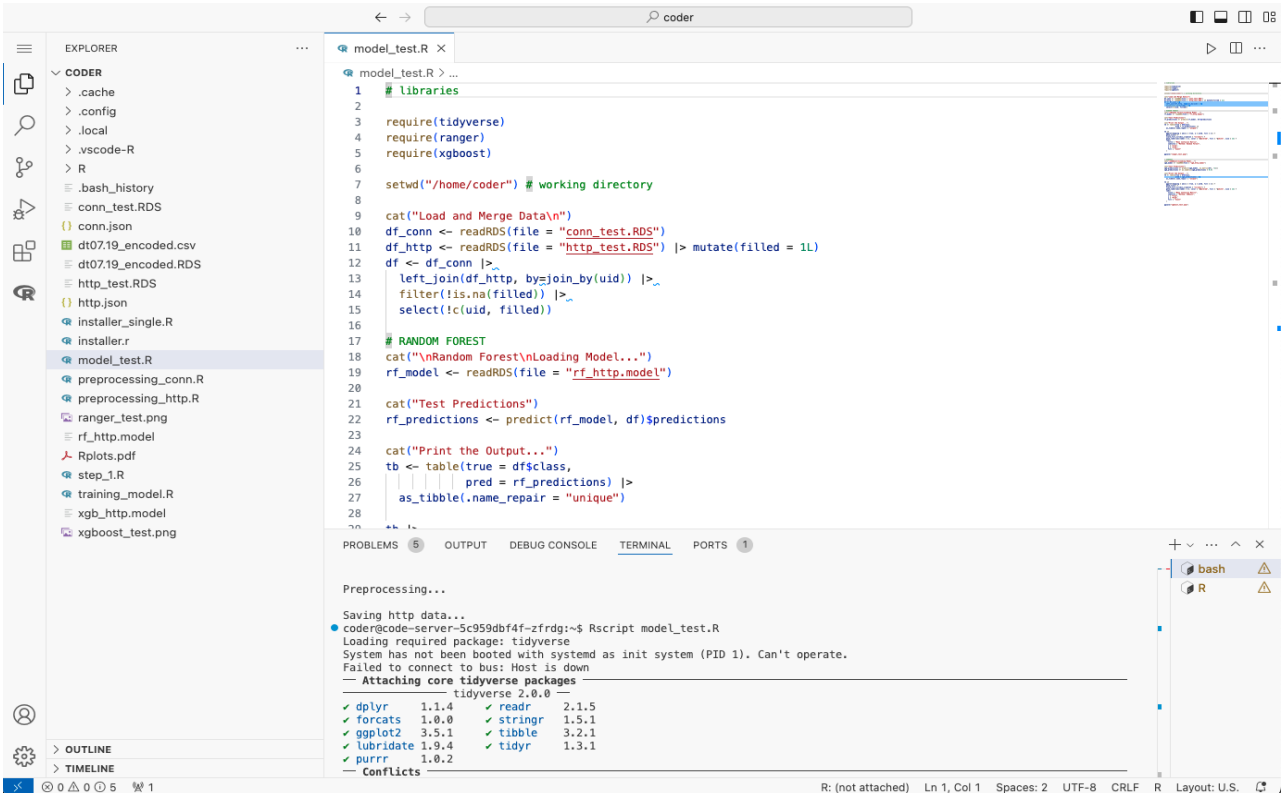

Figura 1 - Schermata dell'ambiente di data analytics basata su Code-server, contenente i codici di esecuzione di R

Docker è una piattaforma open source che permette agli sviluppatori di creare, implementare, eseguire, gestire e aggiornare i container. Il container è una componente standardizzata che può essere eseguita su una varietà di sistemi di elaborazione. Al suo interno ci sono tutte le dipendenze necessarie per farlo funzionare correttamente, come librerie, configurazioni e strumenti. L'uso di questa tecnologia consente di ottenere la portabilità del codice, per cui indipendentemente dal sistema operativo, il codice può essere eseguito senza che sia necessario installare le librerie e configurare opportunamente l'ambiente. Un altro vantaggio di Docker è l'isolamento delle librerie e dell'esecuzione, il singolo container può richiedere qualsiasi configurazione e può incontrare qualsiasi errore di esecuzione senza influire in alcun modo sul sistema operativo sottostante. Inoltre, Docker consente un'elevata scalabilità, permettendo di creare facilmente più copie dello stesso container per gestire un aumento del carico computazionale. Il formato Docker è estremamente veloce nell'esecuzione e può essere facilmente integrato nei sistemi di orchestrazione di container come Kubernetes.

Per le finalità del progetto gli algoritmi di elaborazione basati su Python sono stati integrati in container Docker appositi, poi caricati come pod all'interno di Kubernetes (vedi Figura 2: Schermata del pod con integrato il Docker che contiene il codice in Python).

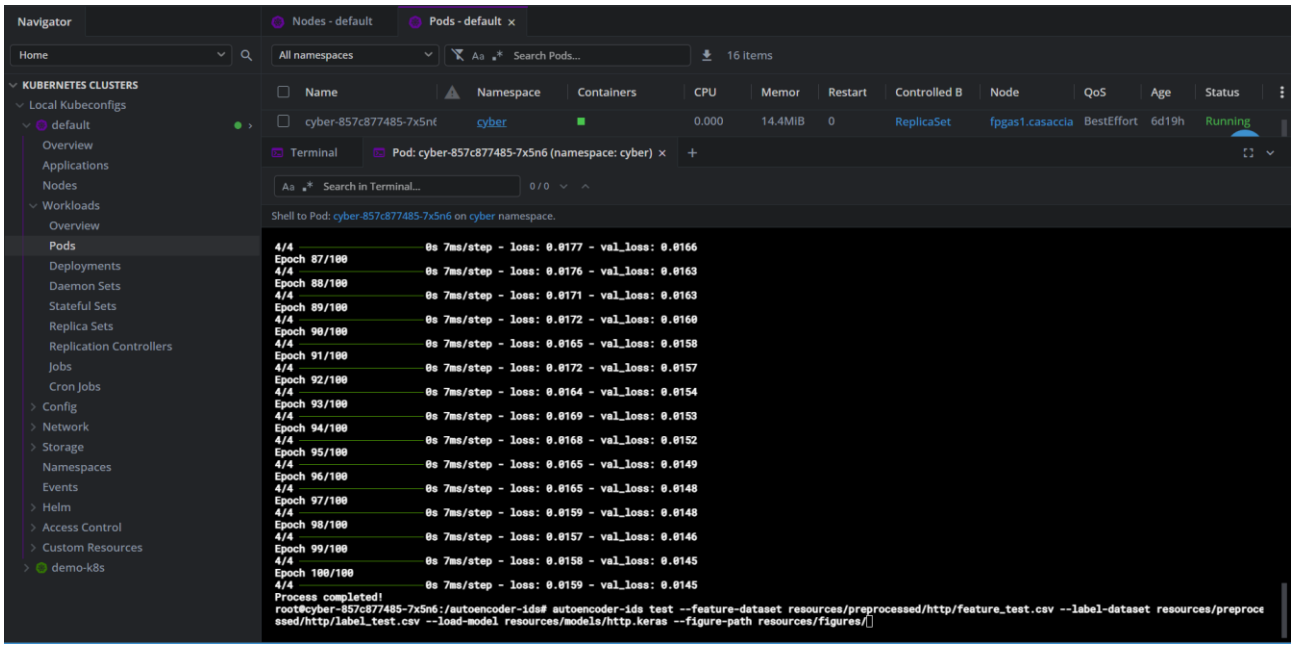

Figura 2: Schermata del pod con integrato il Docker che contiene il codice in Python

4.3 Dashboard di monitoraggio e gestione

La dashboard di monitoraggio e gestione dell'intero sistema basato su Kubernetes è stata implementata come client di connessione al cluster. Tramite questa dashboard è possibile controllare lo stato di ogni singolo pod, accedere alle singole istanze ed eventualmente caricare nuovi pod e ambienti di esecuzione del codice per la data analytics.

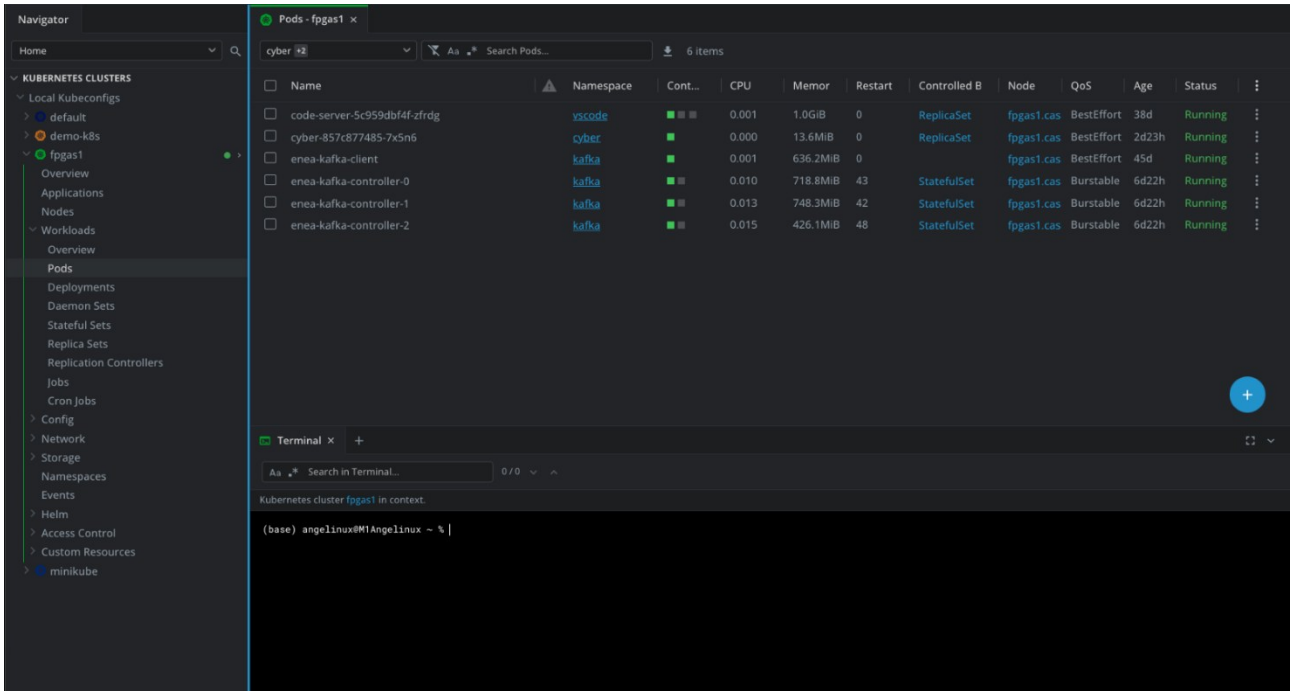

Figura 3: Dashboard dei servizi attivi sul cluster Kubernetes, suddivisi nel namespace kafka, vscode e cyber

I pod in esecuzione nel sistema integrato sono stati suddivisi in diversi namespaces a seconda della loro funzione: kafka per la parte di stream processing, vscode per la parte di data analytics basata su codici R, cyber per la parte di data analytics basata su codici Python (vedi Figura 3).

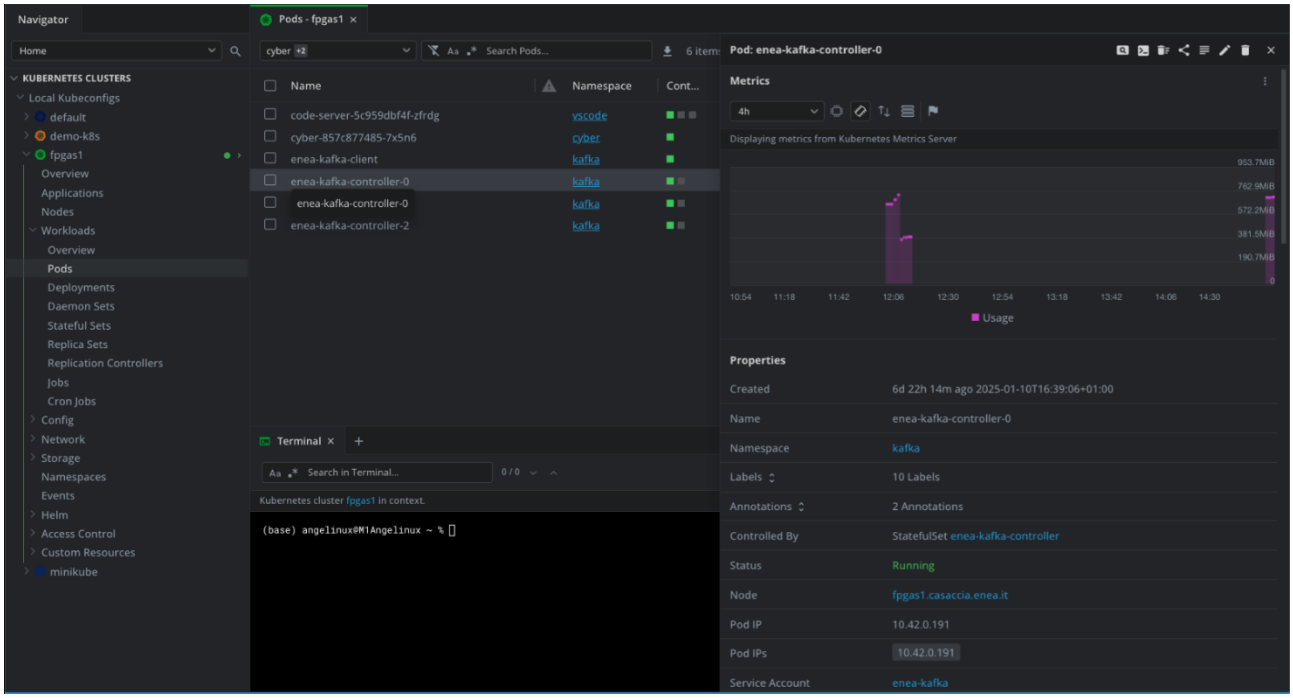

Figura 4: Dashboard con il dettaglio del pod per il primo controller kafka della componente di stream processing (namespace kafka)

Nella dashboard, per ogni pod è possibile verificare lo stato e l'occupazione delle risorse computazionali come la CPU e la RAM. In Figura 4 e Figura 5 è possibile vedere gli esempi dei pod per la componente di stream processing e quella di data analytics.

Per ogni singolo pod è inoltre possibile verificare lo stato, accedere alla riga di comando e ovviamente operare scelte di cancellazione di pod esistenti o di creazione di nuovi.

Infine, per ogni singolo pod è possibile gestire quali risorse di storage utilizzare e questa opzione è esercitata tramite i persistent volume claims (vedi Figura 6).

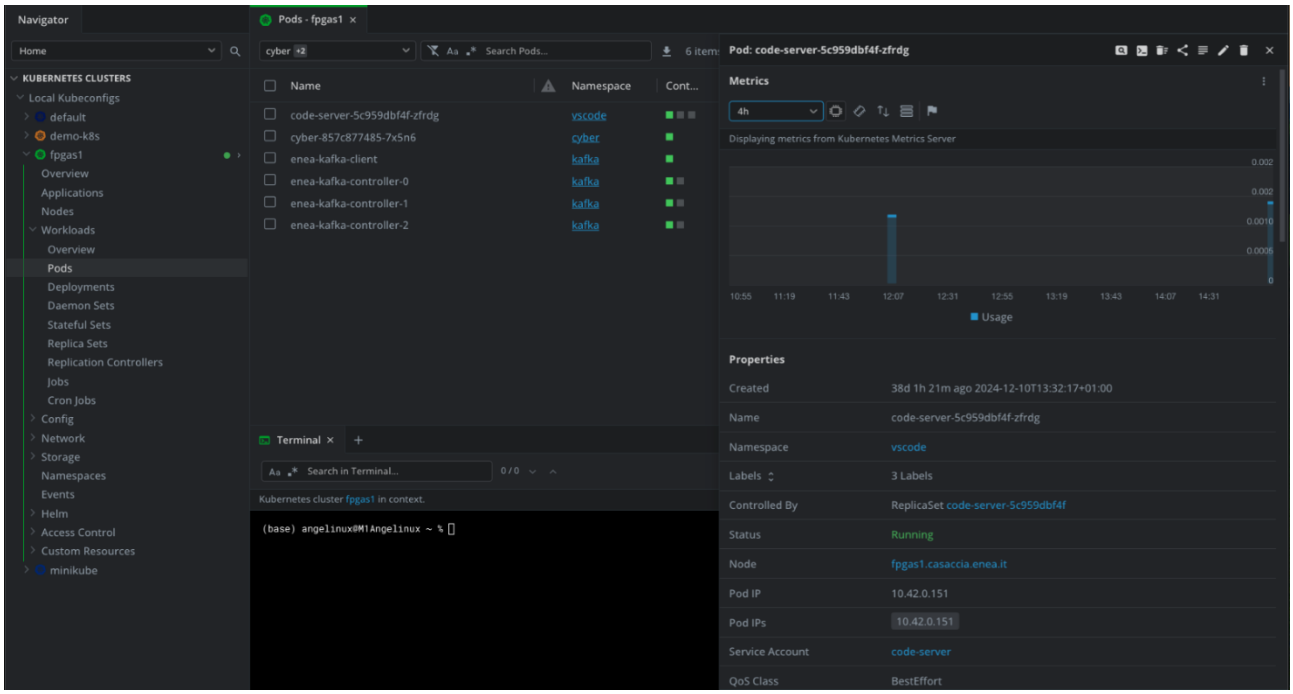

Figura 5: Dashboard con il dettaglio del pod per il componente di data analytics basato su Code-server (namespace vscode)

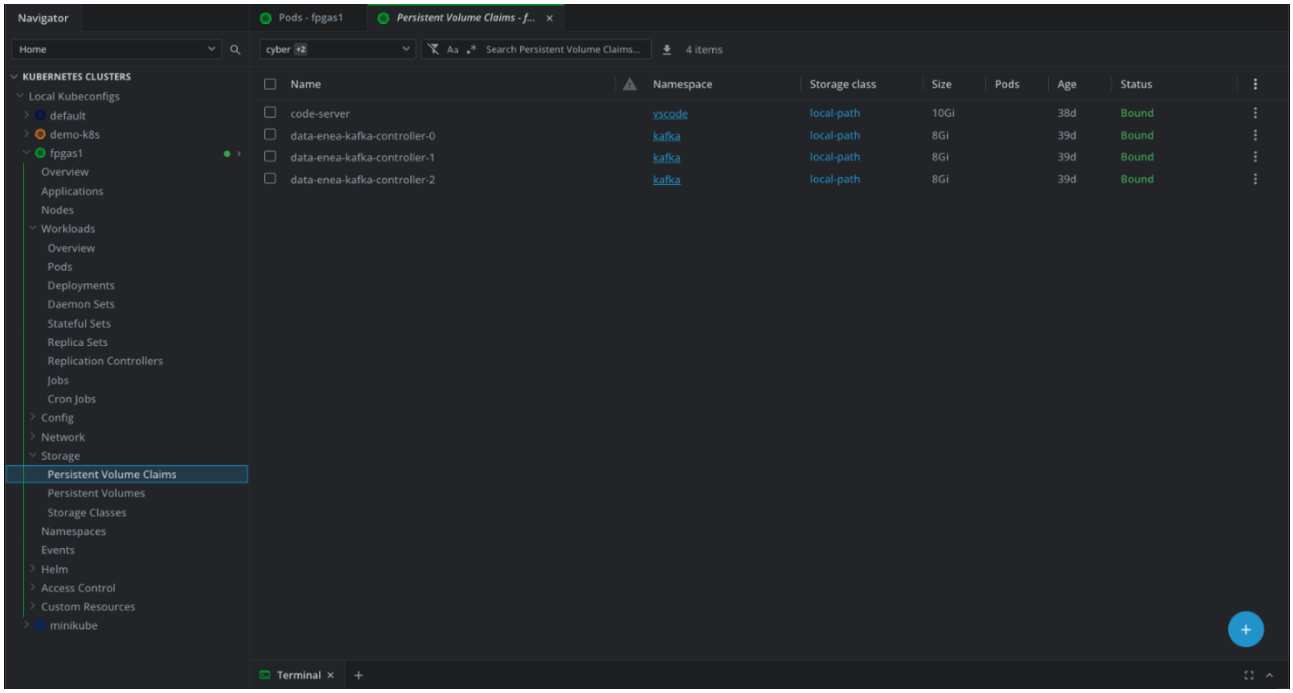

Figura 6: Dashboard con il dettaglio dei persistent volume claims

5 Analisi degli scostamenti su attività e risultati

In generale durante lo svolgimento delle attività sono stati riscontrati lievi scostamenti. Come evidenziato nel piano di rischio, la mancata attivazione della consulenza che a priori

avrebbe potuto comportare un estremo ritardo sulla consegna dei prodotti finali, è stata compensata tramite l'adozione di azioni di mitigazione descritte nel seguito.

5.1 Scostamenti tecnici

Nella fase di realizzazione del progetto ci sono stati alcuni scostamenti tecnici in considerazione del fatto che non è stato più possibile, per problemi organizzativi e a causa delle tempistiche ristrette di progetto, procedere alla formalizzazione di una consulenza esterna per la realizzazione della dashboard. Questo ha comportato la riorganizzazione del lavoro con l'utilizzo preponderante di software open source per la realizzazione dei componenti e l'installazione di un orchestratore di container per facilitare la comunicazione tra di essi. La soluzione realizzata, con estrema semplicità consente di ottenere un sistema IT scalabile e replicabile, in grado di analizzare i dati provenienti dalle sorgenti di rete.

Un altro scostamento tecnico è la scelta di non utilizzare acceleratori hardware specializzati per il training dei modelli. Infatti, durante tutti i test di esecuzione si è rilevato che hardware GPU/FPGA non presentava un netto vantaggio di elaborazione sugli algoritmi presi in considerazione. Pertanto, al momento si è deciso di non includere nel sistema IT questo tipo di risorse hardware, sebbene tutto ciò sia possibile, consentendo quindi una maggiore facilità di replica e di scalabilità degli algoritmi stessi.

5.2 Scostamenti economici

Nella fase di realizzazione del progetto il principale scostamento economico è dovuto alla mancata attivazione, per problemi organizzativi e a causa delle tempistiche ristrette di progetto, del contratto di consulenza inizialmente previsto. Questo ha comportato un maggiore impegno di personale interno per utilizzare e gestire il software open-source e implementare una soluzione alternativa adeguata agli obiettivi del progetto. Come descritto nel paragrafo precedente, questo ha portato ad alcuni scostamenti tecnici che comunque sono stati opportunamente mitigati.

6 Sintesi delle attività svolte

Le attività svolte per la realizzazione dei prodotti definiti nei paragrafi precedenti hanno riguardato in sintesi:

1. la creazione di un sistema IT integrato di componenti containerizzate in cloud, in grado di essere orchestrate in maniera facile ed efficiente;
2. la creazione del componente di stream processing basato su Apache Kafka in grado di processare sotto forma di log gli eventi collegati al traffico della rete dati;
3. la creazione di diversi componenti di data analytics capaci di elaborare questo flusso di eventi utilizzando algoritmi di machine learning e di intelligenza artificiale sviluppati dai partner di progetto per alcuni casi d'uso;
4. l'individuazione di una dashboard con un'interfaccia di gestione facile ed intuitiva per la gestione dei singoli componenti e la loro scalabilità e replicabilità.

7 Dettaglio delle attività svolte

7.1 Sistema IT basato su Kubernetes

Nell'ambito del progetto sono state studiate le possibili implementazioni in grado di consentire la realizzazione di un sistema IT integrato capace di scalare sia orizzontalmente, ottenendo ambienti replicabili in caso di aumento dei carichi di lavoro, che verticalmente, aggiungendo risorse specializzate ad alte prestazioni in caso di necessità.

Si è scelto Kubernetes come sistema in grado di soddisfare queste richieste ed in particolare è stata utilizzata la versione K3s[8], che risulta essere una versione Kubernetes essenziale, con tutti i componenti fondamentali, di facile installazione e manutenzione anche in ambito di produzione.

Il cluster è stato installato sul nodo fpgas1.casaccia.enea.it per consentire di utilizzare all'occorrenza le risorse HW specializzate del nodo.

7.2 Componente di stream processing e test di esecuzione

La componente di stream processing è stata implementata usando chart Helm[9] per Apache Kafka[10]. **Helm** è uno strumento open-source che semplifica notevolmente la gestione e il deployment di applicazioni su cluster Kubernetes, ampiamente utilizzato come gestore dei pacchetti per Kubernetes.

In particolare, questa componente è stata installata nel namespace kafka, con tre controller paritari con funzione di broker Kafka denominati enea-kafka-controller-[0,1,2] e con un client che ospita i codici per le logiche producer-consumer di Kafka.

7.3 Componente di data analytics e test di esecuzione

La componente di data analytics è stata implementata tramite dei pod di Code-server personalizzati per eseguire codici in R (nel namespace vscode), e dei pod standard basati su Docker per eseguire codici Python (nel namespace cyber).

I pod Code-server sono stati installati mediante chart Helm specifiche del progetto open source, mentre i pod basati su Docker sono stati installati utilizzando DockerHub e il comando di gestione di Kubernetes kubectl.

[8] https://k3s.io/

[9] https://helm.sh/

[10] https://github.com/bitnami/charts/blob/main/bitnami/kafka/README.md

7.4 Dashboard di gestione di Kubernetes

La dashboard di monitoraggio e gestione è un componente grafico per la visualizzazione del cluster Kubernetes che semplifica le operazioni che sarebbero comunque eseguibili da riga di comando.

Come strumento di monitoraggio e di gestione del cluster Kubernetes è stato scelto Lens[11], un'interfaccia grafica (GUI) open-source, progettata specificamente per visualizzare in modo intuitivo tutte le risorse, le configurazioni e lo stato delle applicazioni attive sul cluster.

Tramite questa interfaccia si ottiene una dashboard per esplorare il cluster, visualizzare i dettagli dei pod, configurare le applicazioni che sono eseguite nei pod, risolvere eventuali problemi di esecuzione dei pod, monitorare le prestazioni dei singoli componenti.

8 Contributo delle eventuali consulenze alle attività sopra descritte

Come specificato in precedenza, non è stato possibile accedere a consulenze alle attività sopra descritte.

9 Pubblicazioni scientifiche

Le pubblicazioni scientifiche sono quelle congiunte con i partner di progetto, relative allo sviluppo e alla valutazione di algoritmi di machine learning e intelligenza artificiale eseguiti sul sistema IT integrato. Si riporta la lista nel deliverable della LA3.8.

10 Eventi di disseminazione

La partecipazione all'evento conclusivo di presentazione dei risultati del Progetto Integrato Cybersecurity dei sistemi energetici (Workshop-Progetto-Cybersecurity.pdf), presso l'Auditorium GSE in viale Maresciallo Pilsudski 92, Roma, il 6 Dicembre ha dato l'opportunità di condividere con un'ampia platea di partecipanti i risultati delle attività di progetto.

[11] https://github.com/lensapp/lens

11 Descrizione dei risultati ottenuti

11.1 Architettura del sistema

La progettazione della piattaforma di stream analytics è basata su un'architettura distribuita e scalabile, progettata su un cluster Kubernetes ed utilizza Apache Kafka come backbone per l'elaborazione dei dati in tempo reale.

Kubernetes è un sistema open source di orchestrazione di container, ovvero il processo che permette di automatizzare il deployment, la gestione, la scalabilità e il networking dei container. Kubernetes funziona tramite pod e nodi. Il pod è la risorsa che descrive l'unità elementare eseguibile su un nodo del cluster, gestisce uno o più container ed è molto efficace dal punto di vista della scalabilità orizzontale soprattutto alle applicazioni orientate ai microservizi. I nodi eseguono i pod e sono in genere raggruppati in un cluster Kubernetes, astraendo le risorse hardware fisiche sottostanti.

Apache Kafka è una piattaforma distribuita open source di stream processing. Le tre funzionalità principali sono: consentire alle applicazioni di pubblicare o sottoscrivere flussi di dati o eventi, archiviare i record accuratamente in modo tollerante agli errori e durevole, elaborare i record in tempo reale. Inoltre, Kafka ha una bassa latenza e un'alta velocità di gestione, e permette di analizzare in modo affidabile e persistente grandi volumi di dati che provengono da diverse sorgenti.

Una piattaforma a microservizi basata sull'utilizzo combinato di Kubernetes e Kafka permette di avere un sistema scalabile, flessibile e resiliente, perfetto per gestire carichi di lavoro dinamici e per elaborare dati in tempo reale.

11.2 Prestazioni del sistema

La piattaforma di stream analytics introdotta risulta essere ad elevate prestazioni benché il deployment utilizzato sia semplice.

Infatti, l'elaborazione in tempo reale è molto efficace: per quanto riguarda gli eventi/processi al secondo, la piattaforma è in grado di elaborare più di 50 pacchetti trasmessi al secondo, garantendo una risposta rapida e in tempo reale alle variazioni dei dati in ingresso. La latenza media di elaborazione di un singolo evento è di circa 150 millisecondi, assicurando una bassa latenza e una reattività ottimale, come si evince dal test di carico effettuato durante stress su dieci messaggi consecutivi (vedi Figura 7).

```
Message consumed with latency: 0.1685 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114300.146349}'}
Message consumed with latency: 0.1709 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114302.6280499}'}
Message consumed with latency: 0.1660 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114304.848991}'}
Message consumed with latency: 0.1680 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114307.117176}'}
Message consumed with latency: 0.0644 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114309.991528}'}
Message consumed with latency: 0.1696 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114313.317048}'}
Message consumed with latency: 0.1671 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114316.819328}'}
Message consumed with latency: 0.1677 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114319.237053}'}
Message consumed with latency: 0.1665 seconds
Received: {'value': '{"key": "value", "timestamp": 1737114323.167892}'}
Message consumed with latency: 0.0641 seconds
```

Figura 7: Test di carico su dieci messaggi consecutivi

Inoltre, anche la scalabilità del sistema ha ottime performance, poiché sottoponendo la piattaforma a test di carico crescenti, si è dimostrata la capacità di scalare linearmente con l'aumento del volume di dati e del numero di dispositivi connessi. La piattaforma è stata interessata dal processamento di quantità crescenti di elementi di traffico di rete mantenendo costante il rapporto tra numero di eventi processati e tempo di processamento.

Il sistema è stato progettato per essere altamente scalabile, grazie all'utilizzo di tecnologie basate su kubernetes, e quindi sulla realizzazione di microservizi containerizzati gestiti su un sistema cloud. In particolare, i singoli pod che compongono il sistema di stream analytics possono essere scalati sia orizzontalmente, aggiungendo nuovi pod per l'elaborazione dei dati, sia verticalmente andando a modificare a runtime le configurazioni di storage e risorse di calcolo da utilizzare.

Per quanto riguarda le prestazioni della dashboard in termini di tempi di risposta, essi sono variabili in base a diversi fattori, quali la dimensione del cluster Kubernetes, il tipo di operazione che si vuole svolgere sul cluster, il carico del cluster, la latenza di connessione con il cluster e non ultimo la potenza di calcolo del computer su cui la dashboard è in esecuzione. Tenute presenti queste considerazioni, in una configurazione come quella di progetto con un cluster leggero, anche geograficamente distante dalla dashboard e con l'uso di hardware di base senza particolari esigenze i tempi di esecuzione dei comandi della dashboard si mantengono sempre inferiori al secondo nelle fasi di monitoraggio.

Per quanto riguarda le prestazioni della dashboard in termini di scalabilità, per la sua natura, si tratta di un software che può essere installato su qualsiasi piattaforma e non necessita di repliche per essere efficiente e monitorare e orchestrare i container sul cluster Kubernetes, ma si affida alle API di Kubernetes per operare.

Per quanto riguarda le prestazioni della dashboard in termini di usabilità, emerge chiaramente, sulla base delle figure presentate e dell'uso quotidiano, che lo strumento utilizza una interfaccia grafica intuitiva, presenta una notevole facilità d'uso, consente l'accesso alla gamma completa di funzionalità del cluster Kubernetes, fornisce la possibilità di integrarsi con altri strumenti dell'ecosistema Kubernetes. Di conseguenza lo strumento si presta in maniera eccellente alle funzionalità di monitoraggio e di gestione dei componenti software descritti in precedenza sia per utenti non esperti che in condizione in cui la complessità del cluster sottostante può aumentare.

11.3 Utilizzo delle risorse

Analizziamo l'utilizzo delle risorse di memoria, CPU, GPU/FPGA del sistema.

Durante l'esecuzione delle logiche di stream analytics e dei modelli di machine learning, la CPU è utilizzata in media al 30% della sua capacità massima, con alcuni picchi vicini al 40% della capacità come si evince dalla Figura 8, in cui i picchi di utilizzo della CPU si verificano principalmente nelle fasi in cui c'è un traffico più intenso sulla rete dati che si sta processando, ma in ogni caso rimangono nei limiti di tollerabilità del sistema.

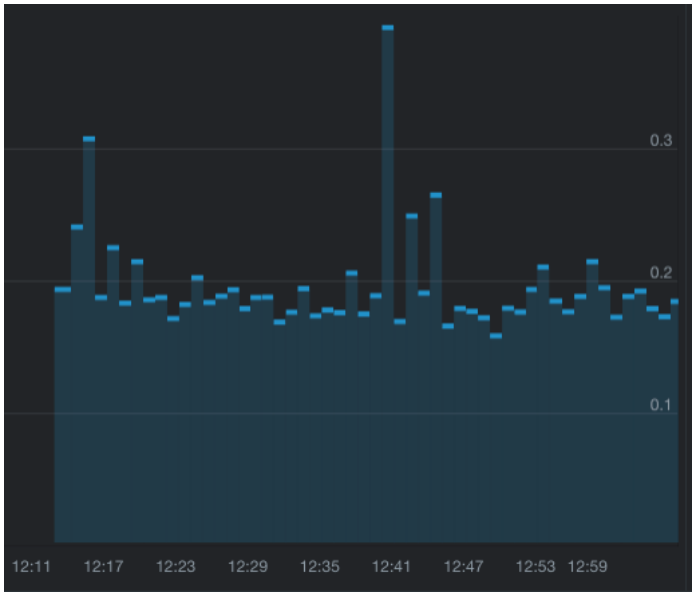

Figura 8: Schermata relativa all'uso della CPU da parte dei processi di stream analytics

Per quanto riguarda la quantità di memoria utilizzata per l'elaborazione dei dati e l'esecuzione dei modelli, questa è di poco superiore ai 9 GB di utilizzo della RAM (vedi Figura 8), in cui la maggior parte della memoria è utilizzata per il caching dei dati durante la fase di stream processing e si mantiene tendenzialmente stabile lungo tutta la durante dei test di esecuzione.

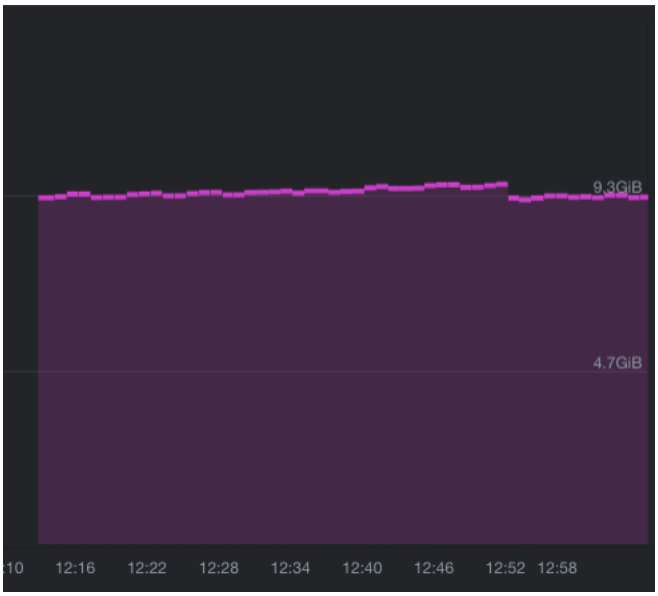

Figura 9: Schermata relativo all'uso della RAM da parte dei processi di stream analytics

In questa specifica applicazione non si è rivelato di marcato beneficio l'utilizzo di architetture speciali per il calcolo quali quelle basate su GPU e FPGA. Infatti, tutti i modelli utilizzati lavorano con la massima precisione e affidabilità già sulla CPU standard della piattaforma. Di conseguenza i modelli elaborati hanno ampia affidabilità e basso costo rispetto alle richieste hardware/software e quindi possono essere messi in produzione senza particolari attenzioni.

11.4 Precisione e accuratezza dei modelli

I modelli di machine learning implementati sulla piattaforma sono tre: un Autoencoder impilato sviluppato dall'Università RomaTre (per i dettagli rimandiamo al report della Linea di Attività 3.9), ed una Random Forest e un Extreme Gradient Boosting sviluppati dall'Università di Bari (per ulteriori dettagli vedere il report della Linea di Attività 3.10).

11.4.1 Autoencoder

Il primo modello implementato sulla piattaforma è un Autoencoder impilato (stacked) non supervisionato, che identifica attività dannose del traffico di rete apprendendo i modelli di comportamento normale, e segnalando variazioni significative come potenziali attacchi.

Il modello ha ottenuto, su un dataset di 1.288.025 record i seguenti risultati in termini di:

- Precision: 98%
- Recall: 97%
- F1-score: 97%
- Accuratezza: 100%

11.4.2 Random Forest

Il secondo modello testato sulla piattaforma è una Random Forest addestrata per identificare minacce di tipo DoS.

Il modello ha prodotto su un dataset di 640.144 osservabili e 43 variabili, i seguenti risultati in termini di:

- Accuratezza: 99.987%
- Recall: 99.965%
- F1: 99.974%

11.4.3 Extreme Gradient Boosting

Il terzo modello è un Extreme Gradient Boosting (XGBoost), ovvero una versione altamente ottimizzata dell'algoritmo di gradient boosting, trainato per riconoscere attacchi di tipo DoS.

Il modello ha prodotto su un dataset di 640.144 osservabili e 43 variabili, i seguenti risultati in termini di:

- Accuratezza: 99.986%
- Recall: 99.979%
- F1: 99.977%

11.5 Conclusioni

Il sistema integrato di stream analytics sviluppato si è dimostrato essere una soluzione efficace per implementare logiche di addestramento ed esecuzione di modelli di machine learning e di intelligenza artificiale applicati all'analisi di flussi di dati reali in ambito cybersecurity. Le prestazioni del sistema, l'utilizzo efficiente delle risorse e l'accuratezza dei modelli di machine learning confermano la validità della soluzione adottata.