

MiniSOC

Scalable and Reliable Services

Caterina Leonelli
Leonardo Bambini
Patrick Di Fazio

2 giugno 2024

Indice

1	Abstract	3
2	Introduzione	4
2.1	Cosa è un SOC	4
2.2	Diagramma Applicativo	4
2.3	Disegno Architettuale	5
3	Struttura del Progetto	6
3.1	Servizio SOC	6
3.1.1	Dashboard	6
3.1.2	Security	7
3.1.3	Statistics	9
3.2	Servizio TOOL	10
3.2.1	Network Analyzer	10
3.2.2	Malware Analyzer	11
3.3	Servizio APP	11
3.4	Screenshots Applicazione	11
4	Tecnologie Utilizzate	15
4.1	AWS	15
4.2	Terraform	16
4.3	Kubernetes	17
4.4	Helm	17
5	Infrastruttura	18
5.1	AWS	18
6	Scalabilità	20
6.1	Scalabilità EKS	20
6.2	Scalabilità Volumi	20
6.3	Scalabilità Kubernetes	21
6.4	Test di Scalabilità	22

<i>INDICE</i>	2
7 Affidabilità	23
7.1 Affidabilità Kubernetes	23
7.2 Affidabilità AWS	23
8 Dashboard e Frontend	24
Bibliografia	26

Capitolo 1

Abstract

Il progetto implementa un Security Operation Center che monitora e testa le risorse dell'utente, effettua analisi di rete, scan di malware e permette operazioni di difesa come lo spegnimento di risorse in caso di attacco.

In questo scenario il SOC è implementato come un microservizio autoscalabile installato su rete privata che protegge altri servizi esposti su rete pubblica. Nel caso di MiniSOC l'applicazione esposta è un semplice server NGINX.

Il progetto è basato su microservizi (pod) creati tramite Kubernetes e prevede un deploy sia su rete locale sia su cloud quali AWS ed Azure. La facilità di deploy è data da singoli file YAML di risorse Kubernetes per creare l'applicazione e singoli file YAML Terraform per la creazione dell'infrastruttura in ambiente cloud. Il monitoraggio dell'applicazione è basato su tool open source quali Grafana e Prometheus installati tramite Helm charts.

Capitolo 2

Introduzione

2.1 Cosa è un SOC

Un Security Operation Center (SOC) è un team specializzato nella sicurezza IT, interno o esterno all'azienda, che monitora e protegge l'infrastruttura IT 24/7, rilevando e rispondendo agli eventi di sicurezza in tempo reale. Gestisce le tecnologie di sicurezza e analizza continuamente le minacce per migliorare la protezione dell'organizzazione. Il SOC prepara e pianifica la sicurezza gestendo l'inventario degli asset, eseguendo aggiornamenti e backup, sviluppando piani di risposta agli incidenti e testando regolarmente le vulnerabilità. Monitora costantemente l'infrastruttura IT, analizza i dati di log per rilevare anomalie e risponde rapidamente alle minacce. Dopo un incidente, ripristina gli asset colpiti, perfeziona le misure di sicurezza e garantisce la conformità alle normative. Il SOC migliora le misure preventive, accelera il rilevamento delle minacce e ottimizza la risposta agli incidenti, aumentando la fiducia dei clienti e semplificando la conformità normativa.

2.2 Diagramma Applicativo

Nella figura sottostante sono rappresentate le chiamate che il SOC esegue verso il resto dell'applicazione. Possiamo notare come gestisce, esegue e fornisce richieste verso gli altri microservizi.

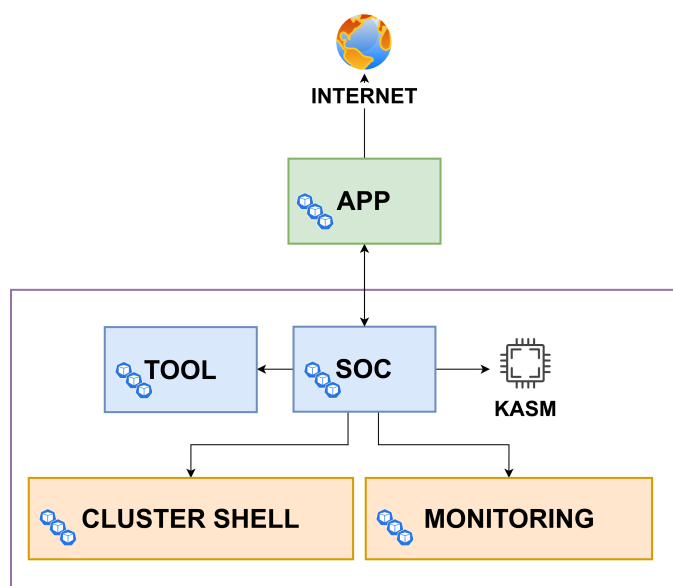


Figura 2.1: Diagramma Applicativo

2.3 Disegno Architetture

Nel disegno architetture sono evidenziati i due nodi di Kubernetes, le differenti reti, i volumi condivisi e la parte di applicazione gestita dagli autoscaler. La comunicazione verso internet durante la fase di testing è stata gestita tramite un LoadBalancer **MetallB** [1]. Per fare il deploy dell'applicazione in ambiente non-cloud si è usato **K3S** [2]. **KASM** [3] è qui gestito come servizio esterno installato su macchina virtuale.

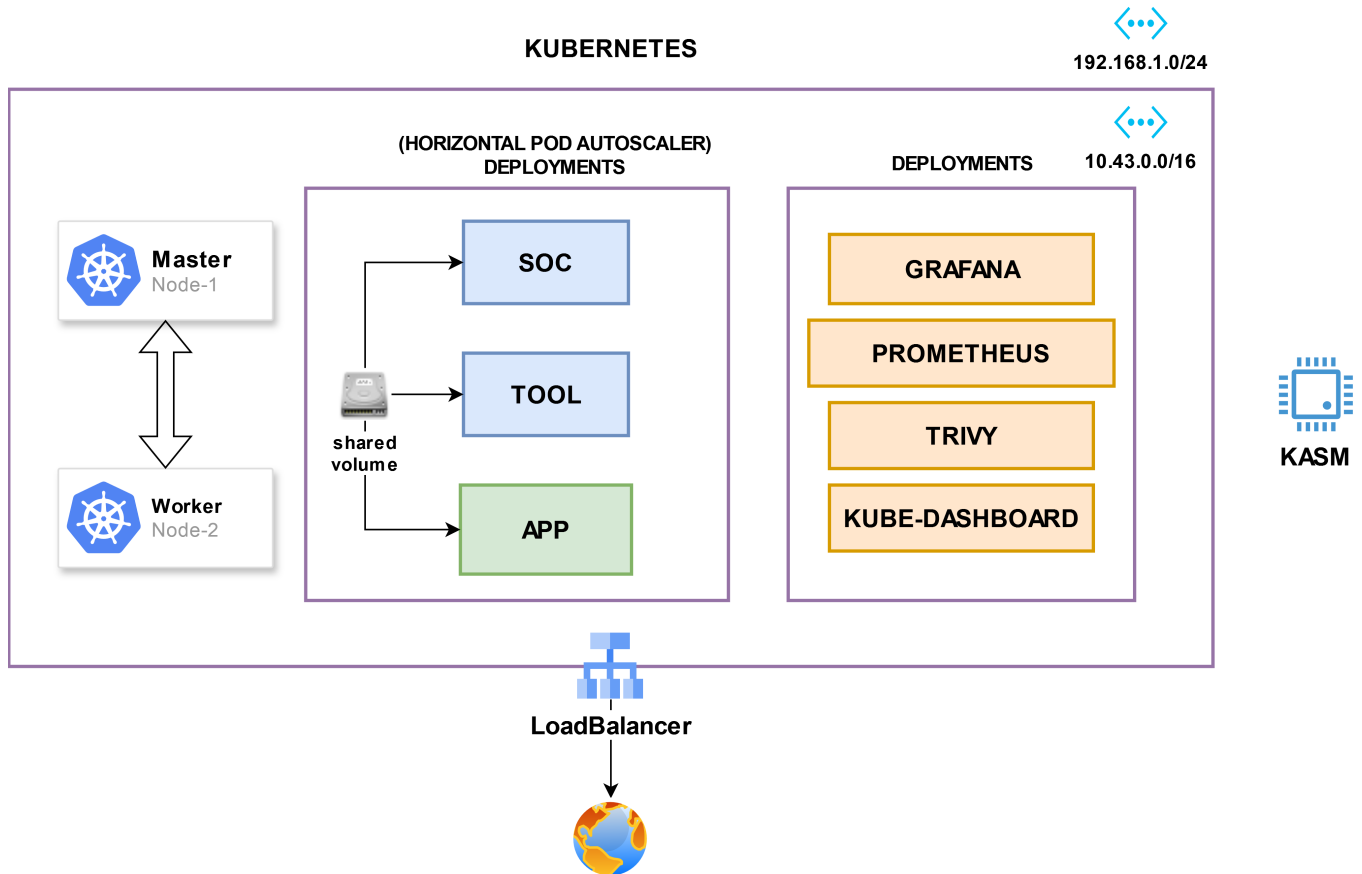


Figura 2.2: Diagramma Infrastrutturale

Capitolo 3

Struttura del Progetto

3.1 Servizio SOC

SOC è il microservizio che si occupa di comunicare direttamente con l'utente finale e di inviare richieste ad altri microservizi per attuare delle operazioni di sicurezza.

In particolare, SOC presenta una parte di autenticazione tramite username e password, salvate direttamente su un DB SQL. Ogni operazione eseguita all'interno di SOC deve essere autenticata.

Le operazioni del SOC si suddividono in 3 categorie:

3.1.1 Dashboard

Cluster

Punto di partenza dell'applicazione, nonché landing page dopo aver effettuato l'applicazione, l'unica operazione di questa sezione si chiama "Cluster" e serve a mostrare una dashboard che permette di gestire sia il cluster Kubernetes che le applicazioni in esecuzione su di esso. A esempio si possono trovare informazioni sulle risorse (utilizzo di CPU, memoria), sui pod (numero di pods allocati), sulle Replica, etc.

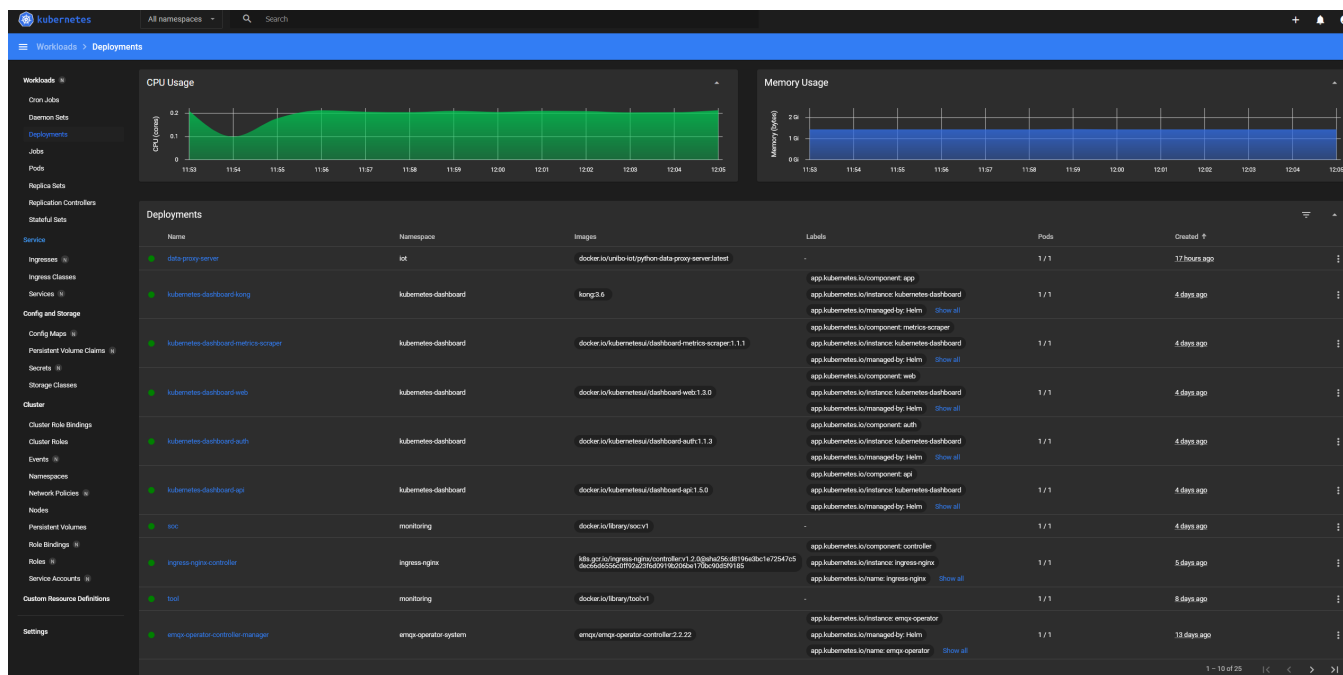


Figura 3.1: Kubernetes Dashboard

3.1.2 Security

Queste operazioni di sicurezza forniscono strumenti avanzati per monitorare e proteggere l’infrastruttura, consentendo di rilevare e rispondere tempestivamente alle minacce e migliorando la sicurezza complessiva dell’organizzazione.

Network Analyzer

Questa sezione consiste nell’analisi del traffico di rete. L’utente può fare l’upload di uno o più file di tipo **.pcap** o **.pcapng** per poi visualizzarli in una tabella che apparirà in output. Sono incluse anche operazioni di filtraggio di protocolli comuni quali **SSH**, **HTTP**, **DNS**, **ICMP**, **SSL** e **TCP**. L’analisi di questi pacchetti di rete viene effettuata tramite una chiamata HTTP al microservizio **TOOL**. Si veda la Sezione 3.2 per maggiori informazioni su come viene effettuata l’analisi.

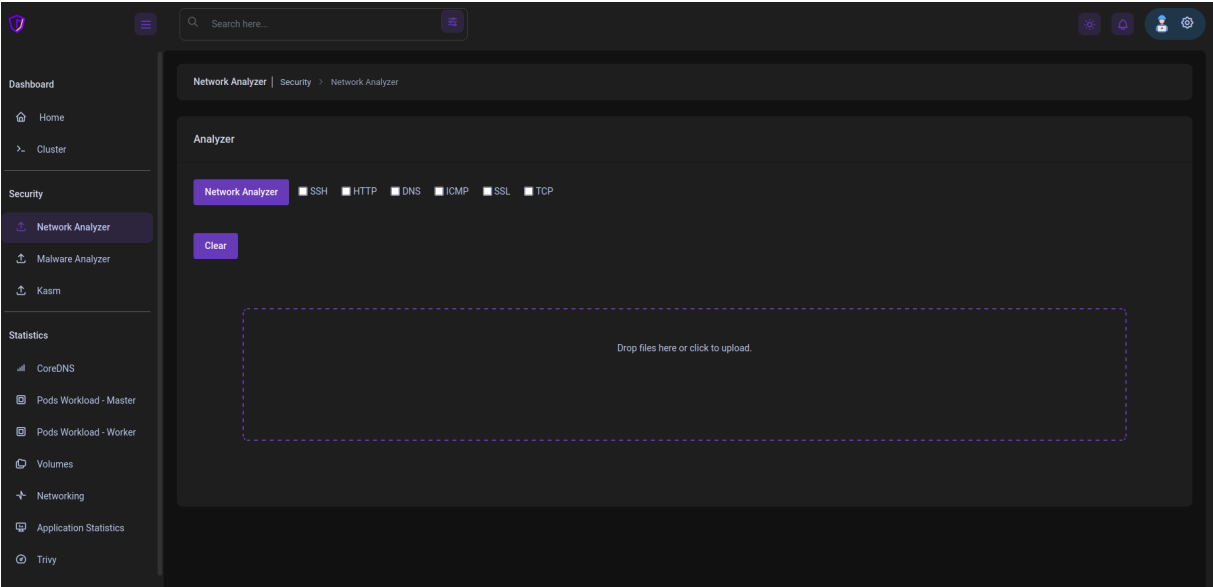


Figura 3.2: Pagina del Network Analyzer

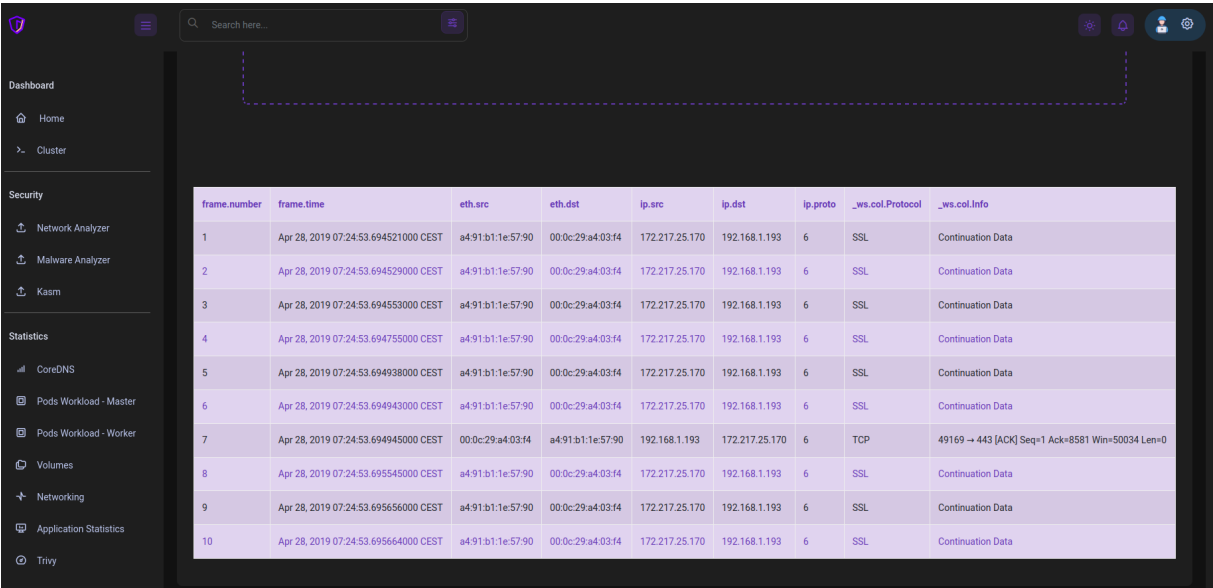


Figura 3.3: Output del Network Analyzer

Malware Analyzer

Questo analyzer è utile per scansionare dei file che possono potenzialmente essere infettati da malware. L'utente può fare l'upload di uno o più file di ogni tipo, con ogni estensione, e successivamente verrà mostrato in output una tabella in cui vengono descritti i malware trovati all'interno dei file. In particolare, viene indicato il nome del malware e il file specifico nella quale si nasconde, più opzionalmente l'autore del malware e la descrizione del malware. L'analisi del file effettiva viene realizzata tramite una chiamata HTTP verso il microservizio **TOOL**. Si veda la Sezione 3.2 per maggiori informazioni su come viene effettuata l'analisi.

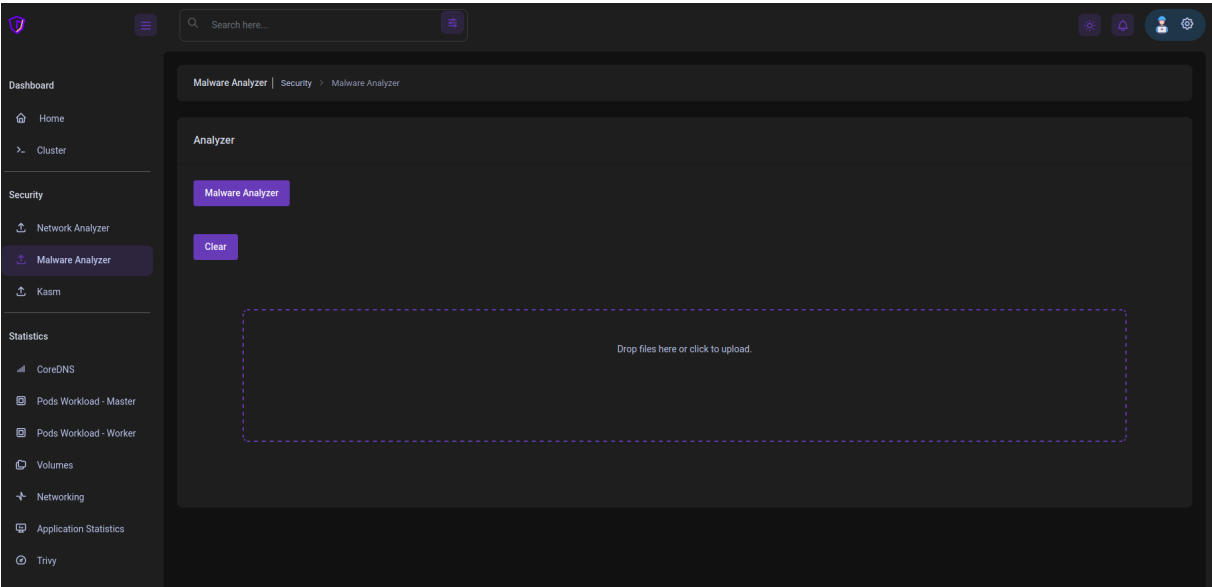


Figura 3.4: Pagina del Malware Analyzer

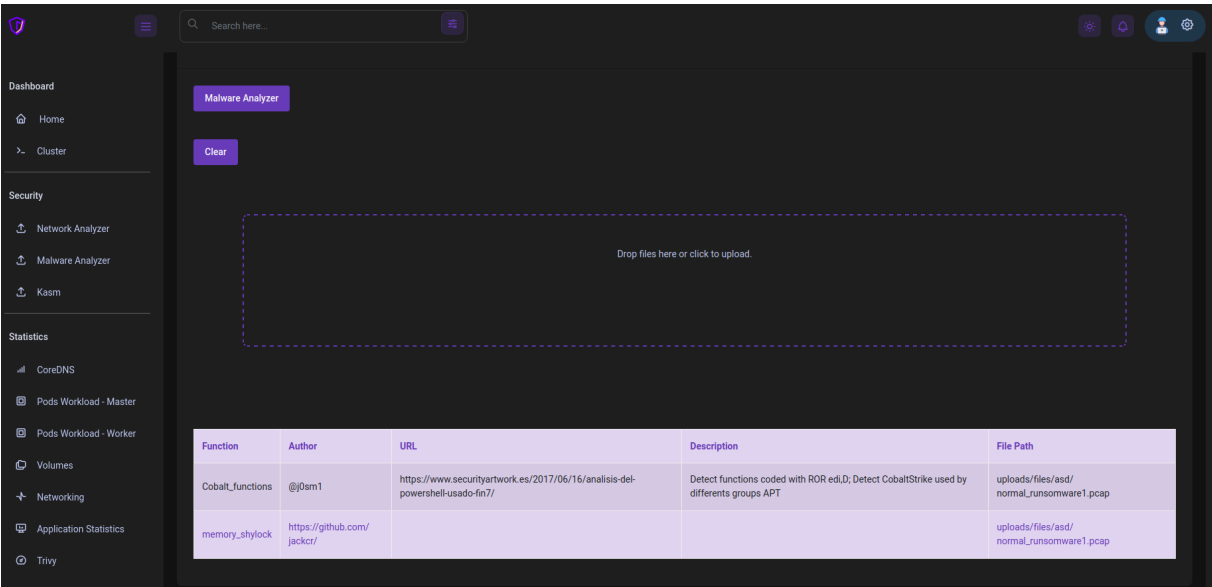


Figura 3.5: Output del Malware Analyzer

Kasm

Kasm è una piattaforma per connessione tramite browser a macchine virtuali e container. Questo servizio permette all'utente che usa il SOC di ottenere delle sandbox per manipolare file pericolosi in modo sicuro. Ad esempio, in queste sandbox si possono aprire file contenenti dei malware e installare dei tool per reverse engineering e malware analysis, senza che il cluster kubernetes sottostante rischi di essere infettato, grazie alle regole di rete presenti nella virtual machine. Per utilizzare Kasm, il microservizio SOC effettua una chiamata direttamente al servizio esterno di Kasm, includendolo nella pagina del SOC.

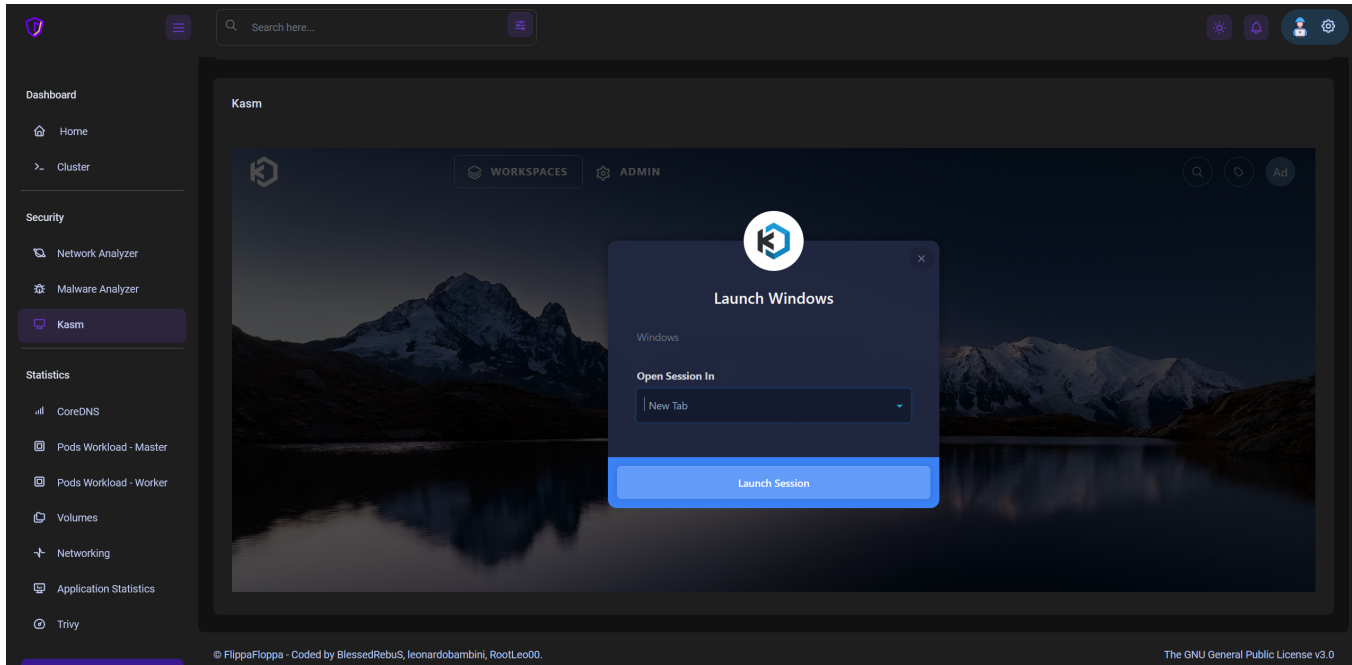


Figura 3.6: KASM

3.1.3 Statistics

Queste informazioni di servizio forniscono un accesso centralizzato a una vasta gamma di statistiche critiche, che sono essenziali per il monitoraggio, la gestione e la protezione della infrastruttura.

CoreDNS

CoreDNS fornisce statistiche DNS in tempo reale. Gli utenti possono accedere a una dashboard grafica alimentata da Grafana, che visualizza le metriche DNS raccolte. Questo strumento aiuta a monitorare e analizzare le query DNS, individuare potenziali problemi di risoluzione dei nomi e ottimizzare le prestazioni del servizio DNS.

Pods Workload - Master

Offre una panoramica delle risorse compute utilizzate dai nodi master di Kubernetes. Questo include il monitoraggio dell'utilizzo della CPU, della memoria e delle risorse di rete. Le statistiche sono presentate attraverso una dashboard Grafana che consente di visualizzare e analizzare il carico di lavoro sui nodi master.

Pods Workload - Worker

Simile alla sezione master, fornisce una visualizzazione dettagliata delle risorse utilizzate dai nodi worker di Kubernetes.

Volumes

Consente di monitorare l'utilizzo dei volumi persistenti all'interno di Kubernetes. Le statistiche fornite includono informazioni sull'allocazione e l'utilizzo dello spazio di archiviazione, permettendo di garantire non ci siano rischi di esaurimento dello spazio.

Networking

Gli utenti possono accedere a statistiche dettagliate sul networking dei pod all'interno del cluster Kubernetes. Questa sezione offre informazioni sulle connessioni di rete, il throughput e le latenze, aiutando a identificare e risolvere problemi di rete che potrebbero influenzare le prestazioni delle applicazioni.

Application Statistics

Fornisce una panoramica delle statistiche relative alle applicazioni in esecuzione. Queste statistiche includono metriche sulle performance delle applicazioni, come il tempo di risposta, il numero di richieste e l'uptime.

Trivy

Consente di accedere alla dashboard di Trivy, uno strumento di sicurezza per la scansione delle vulnerabilità. Trivy analizza immagini container e altri artefatti software alla ricerca di vulnerabilità note, permettendo agli utenti di identificare e risolvere rapidamente le falle di sicurezza.

3.2 Servizio TOOL

Questo microservizio è responsabile della parte di analisi dei file dati in input dall'utente per le operazioni di Network Analyzer e Malware Analyzer (si veda Sezione 3.1.2).

3.2.1 Network Analyzer

Questa operazione permette di analizzare i file di traffico di rete (file di tipo .pcap o .pcapng) caricati dall'utente. I parametri richiesti sono:

1. user: il nome dell'utente che ha caricato i file.
2. quantity: il numero di pacchetti da analizzare.
3. index_start: l'indice di partenza per l'analisi.
4. filters: eventuali filtri per restringere l'analisi a determinati protocolli (ad esempio, SSH, HTTP, DNS).

Il processo di analisi include:

1. Unione dei file .pcap dell'utente in un unico file tramite il comando `mergecap`.
2. Estrazione di un intervallo specifico di pacchetti dal file unito tramite il comando `editcap`.
3. Analisi dei pacchetti estratti tramite il comando `tshark`[4], con eventuali filtri applicati.
4. Restituzione dei risultati in formato CSV.

3.2.2 Malware Analyzer

Questa operazione permette di analizzare i file caricati dall'utente per identificare potenziali malware. L'unico parametro richiesto è il nome dell'utente che ha caricato i file.

Il processo di analisi include:

1. Scansione dei file dell'utente utilizzando YARA con queste regole standard di detection.
2. Ordinamento e formattazione dei risultati della scansione per una migliore leggibilità.

Per fare la scansione viene utilizzato **YARA** [5] con filtri specifici per la malware detection, come ad esempio filtri per rilevare ransomware. In particolare YARA permette di scansionare file di qualsiasi tipo e con qualsiasi estensione. Inoltre yara può scansionare uno o più file chiamando il suo comando da terminale una volta, per poi produrre i risultati di ogni singolo file separatamente. YARA anche permette la flessibilità di scegliere le rules con la quale effettuare pattern di malware noti,

3.3 Servizio APP

Questa è una delle applicazioni che l'utente ha sviluppato e che quindi vuole equipaggiare con il servizio MiniSOC. Nel nostro progetto, questa applicazione è un server nginx che non ha alcuna funzione specifica al di sopra della sua semplice esistenza.

3.4 Screenshots Applicazione

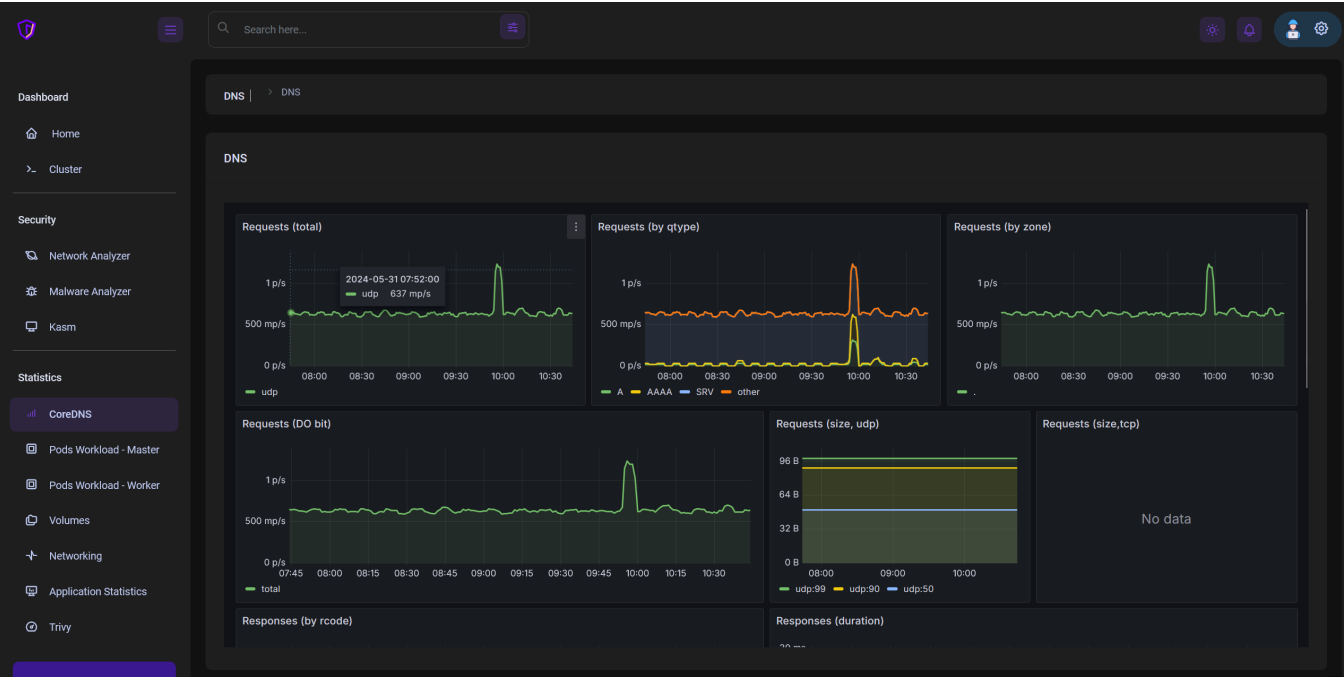


Figura 3.7: CoreDNS

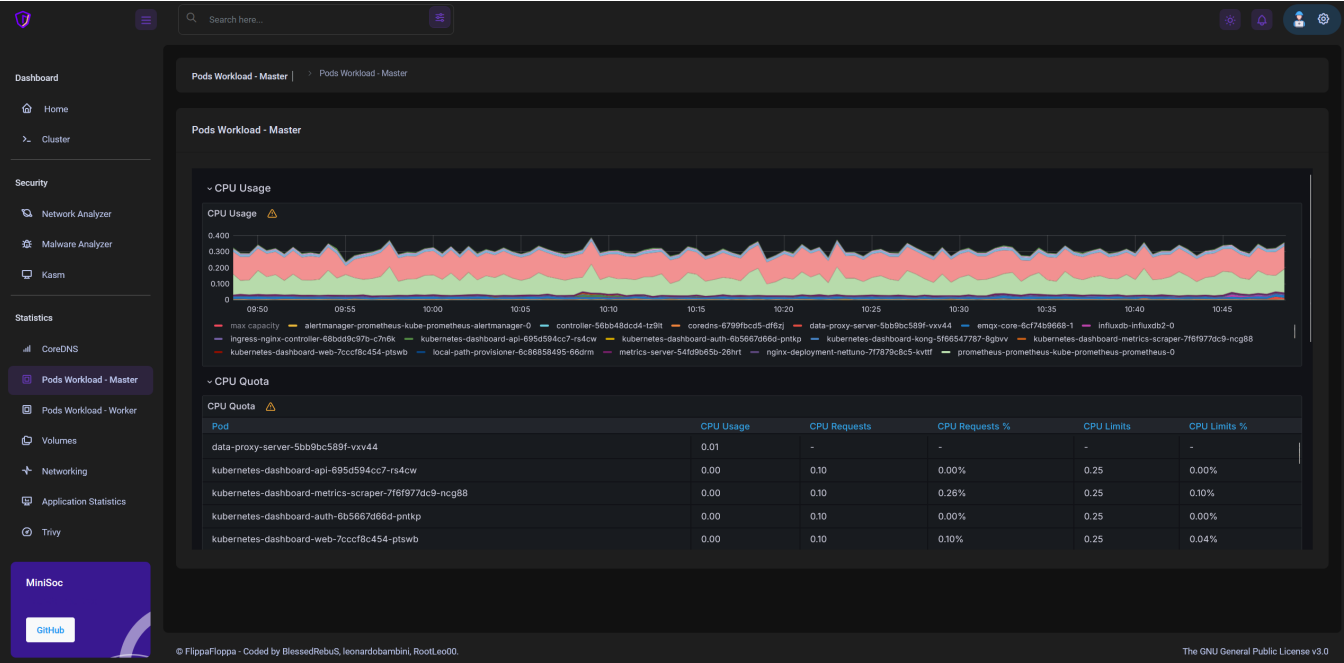


Figura 3.8: Pods workload

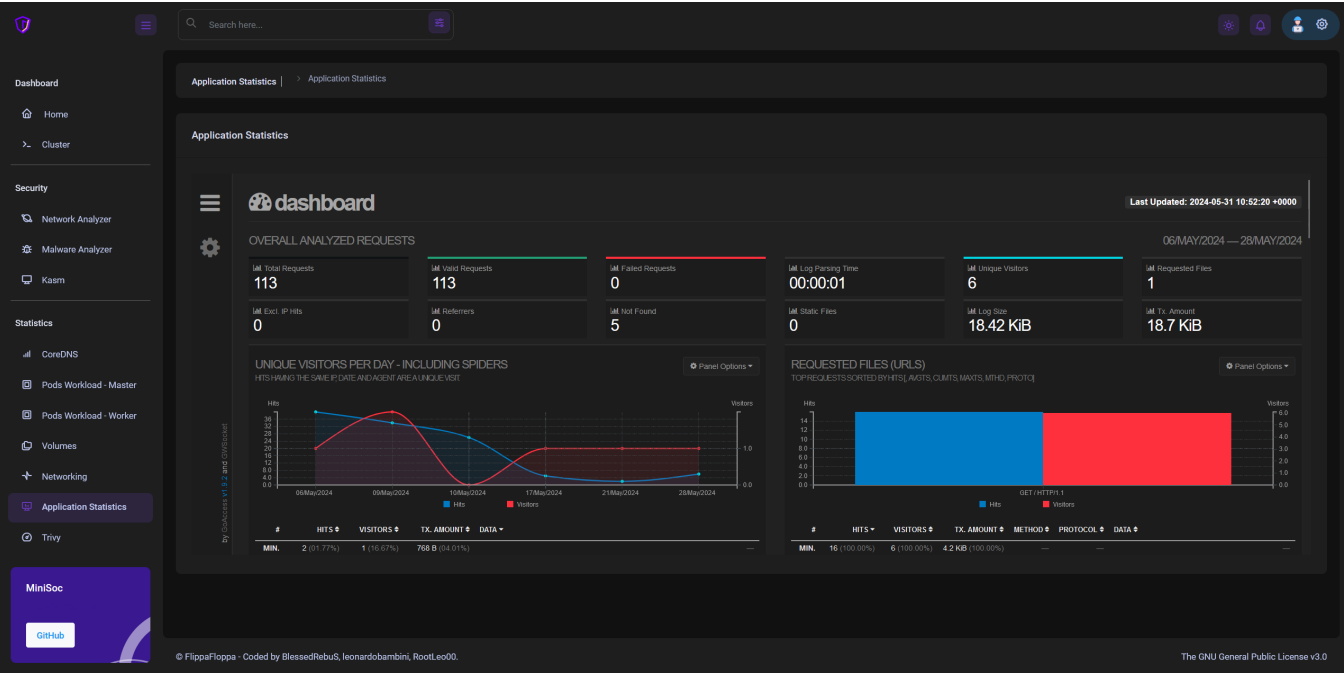


Figura 3.9: Application Statistics

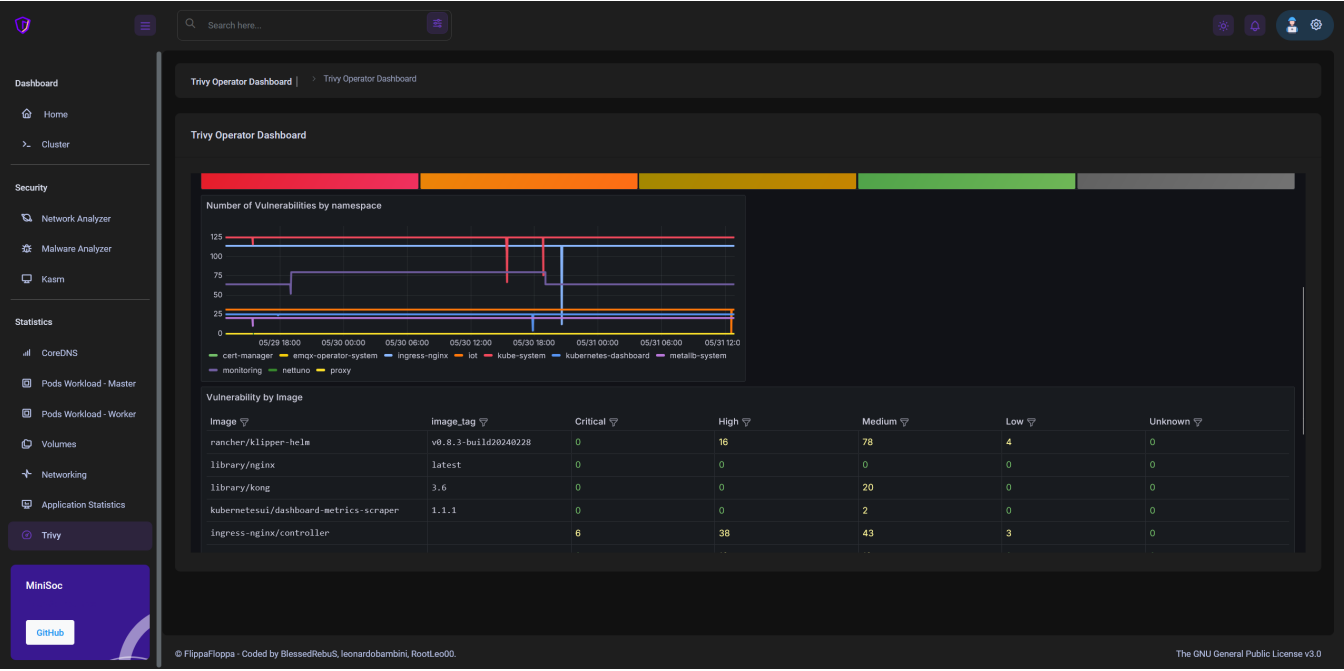


Figura 3.10: Trivy

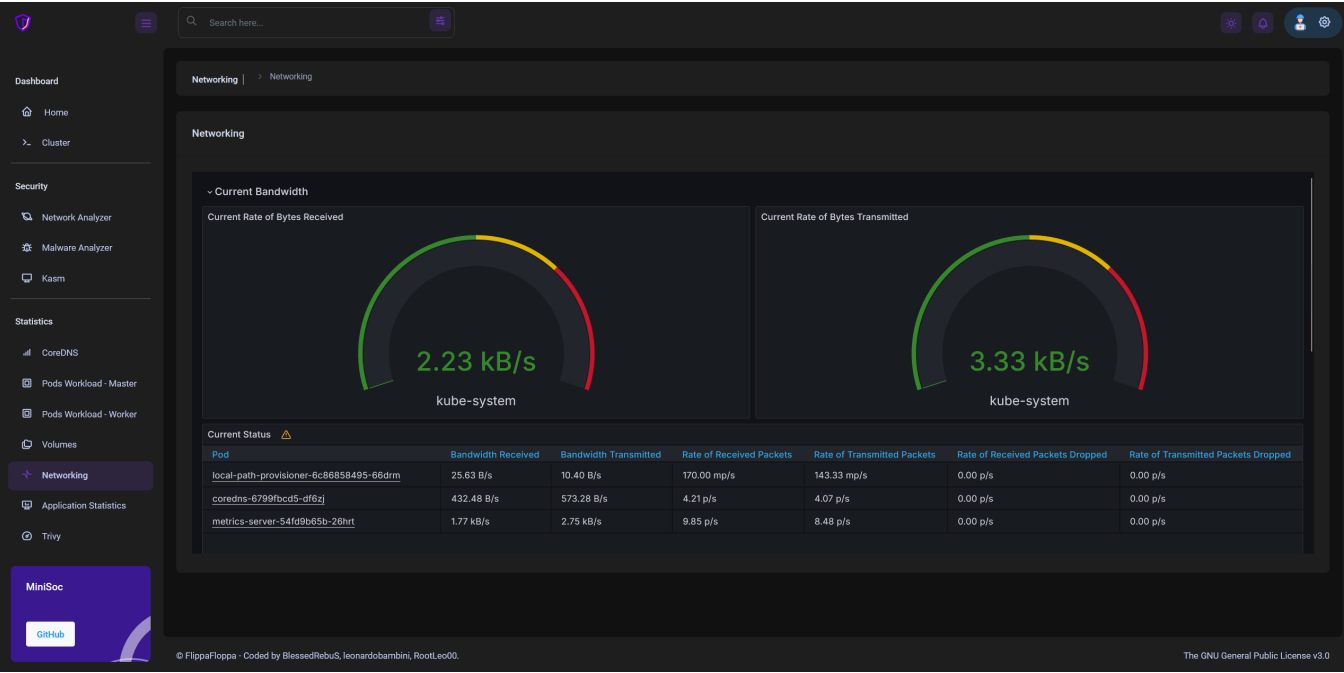


Figura 3.11: Networking

Capitolo 4

Tecnologie Utilizzate

4.1 AWS

AWS (Amazon Web Services) è il public cloud provider di Amazon, nato nel 14 marzo 2006. AWS offre una vasta gamma di servizi di diverse tipologie (computing, storage, database, networking, CDN, ...), attualmente più di 200 sono i servizi offerti da questa piattaforma. AWS è il public cloud provider più utilizzato, seguito da Microsoft Azure e Google Cloud Platform. I suoi datacenter sono presenti in tutto il mondo, con più di 30 regioni disponibili. Nelle immagini sottostanti è possibile vedere ad oggi i più grandi cloud provider e l'estensione dei datacenter di AWS.

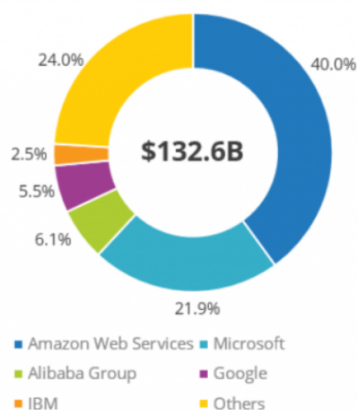


Figura 4.1: Market Capitalization dei public cloud provider[6]



Figura 4.2: Datacenter (region) di AWS[7]

4.2 Terraform

Terraform è uno strumento IaC (Infrastructure as Code) open source. Tramite il linguaggio HCL (Hashi-Corp Language) consente di descrivere le infrastrutture in termini di risorse in modo dichiarativo. IaC è uno dei principi di DevOps e Terraform è lo standard industriale, spesso i cloud provider offrono delle loro alternative (CloudFormation di AWS ad esempio). Terraform rende possibile costruire un'intera infrastruttura (la configurazione di essa spetta ad altri tool come Ansible), gestirne i cambiamenti e distruggerla interamente via software, senza dover fare operazioni manuali. Di seguito, nell'immagine, il workflow di Terraform per la gestione delle risorse infrastrutturali.

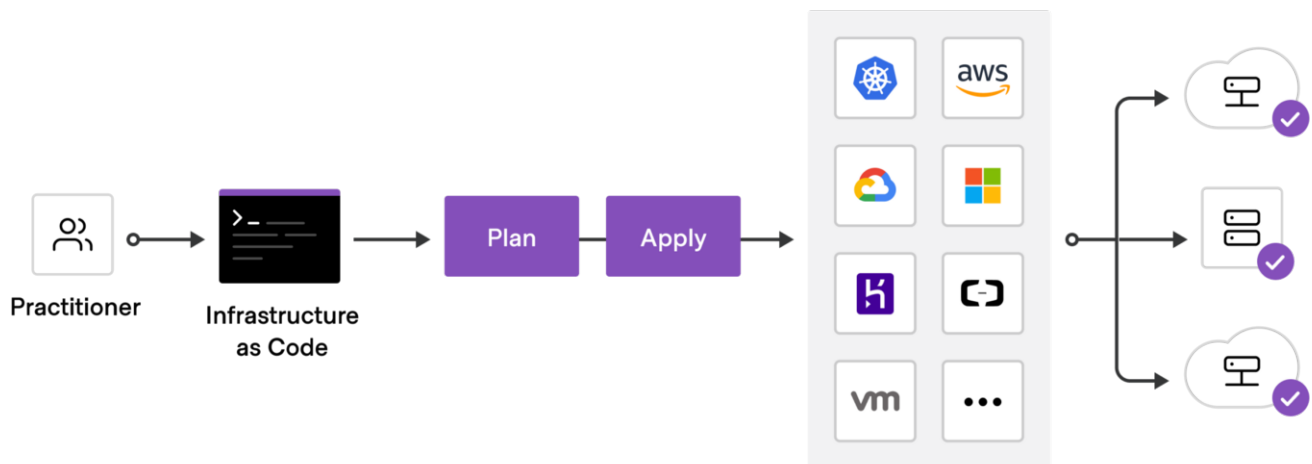


Figura 4.3: Terraform pipeline[8]

4.3 Kubernetes

Kubernetes è lo standard industriale per quanto riguarda l'orchestrazione di container. Un orchestratore di container è un software che gestisce il ciclo di vita dei container, sui quali vengono inseriti microservizi. Gli orchestratori gestiscono anche tutti gli aspetti di networking tra i container, implementando delle vere e proprie reti e sottoreti virtuali. Anche la gestione delle risorse fisiche è delegata all'orchestratore, che secondo diversi criteri, assegna a ogni container diverse tipologie di risorse (CPU, Memoria, Volumi, ecc...). Nell'immagine sottostante, l'architettura interna di Kubernetes.

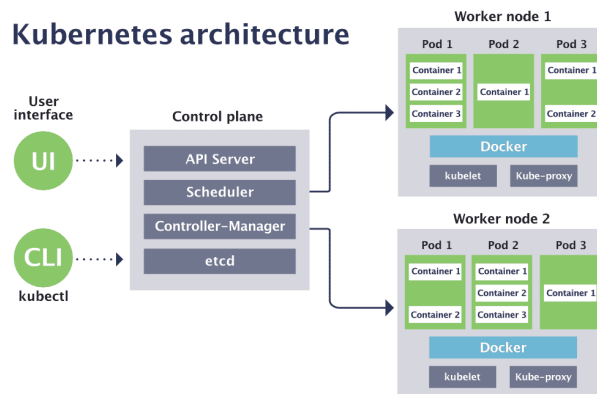


Figura 4.4: Architettura di Kubernetes[9]

4.4 Helm

Helm [10] è un gestore di pacchetti per Kubernetes. Grazie a Helm è possibile mettere insieme diversi componenti (file YAML di Kubernetes) in un Chart e creare dei pacchetti, installabili in un colpo solo. Helm, insieme a tool come ArgoCD, sono una scelta obbligata per cluster non banali, in quanto la gestione dei singoli file YAML risulta infattibile. Helm offre anche l'opportunità di creare dei template riutilizzabili, valorizzati per ogni istanza tramite un Value file dedicato.



Figura 4.5: Helm e Kubernetes

Capitolo 5

Infrastruttura

Il progetto si basa su Kubernetes ed è stato realizzato su due piattaforme diverse:

- AWS
- Cluster Proxmox on-premises

5.1 AWS

Il progetto è stato realizzato, nella sua versione AWS, con il supporto di EKS (Elastic Kubernetes Service), il servizio gestito da AWS per cluster Kubernetes. In Figura 5.1 si riporta il diagramma infrastrutturale, interamente realizzato tramite codice Terraform in modo dichiarativo e facilmente replicabile.

Il cluster si compone di 3 nodi: un master e due worker. I nodi sono gestiti da istanze EC2, tutte con le seguenti specifiche:

- OS: **Ubuntu**
- AMI: **ami-04be05782cc558e6b**
- Instance type: **t3.small**
- Subnet: **srs_subnet1(/2/3)**

Per garantire la massima affidabilità, le macchine EC2 sono state allocate in diverse Availability Zone[11], in modo da poter resistere a un eventuale guasto che coinvolge fino a due data center di AWS.

I nodi si interfacciano a Internet mediante l'ausilio di un Load Balancer, istanziato e configurato automaticamente da EKS, permettendo ai vari microservizi di potersi esporre al pubblico mediante dei Service (Kubernetes) di tipo Load Balancer.

I vari microservizi presenti nel cluster Kubernetes utilizzano delle immagini pubblicate su DockerHub, il repository di immagini ufficiale di Docker. Al momento della creazione dei microservizi, sarà compito del cluster scaricare le immagini dalla repository e usarle per costruire i vari pod.

È stata inserita una macchina Windows per implementare la funzionalità di malware analysis, essa ha le seguenti specifiche:

- OS: **Windows**
- AMI: **ami-0847a7983fdc60e79**
- Instance type: **t2.micro**
- Subnet: **srs_subnet1**

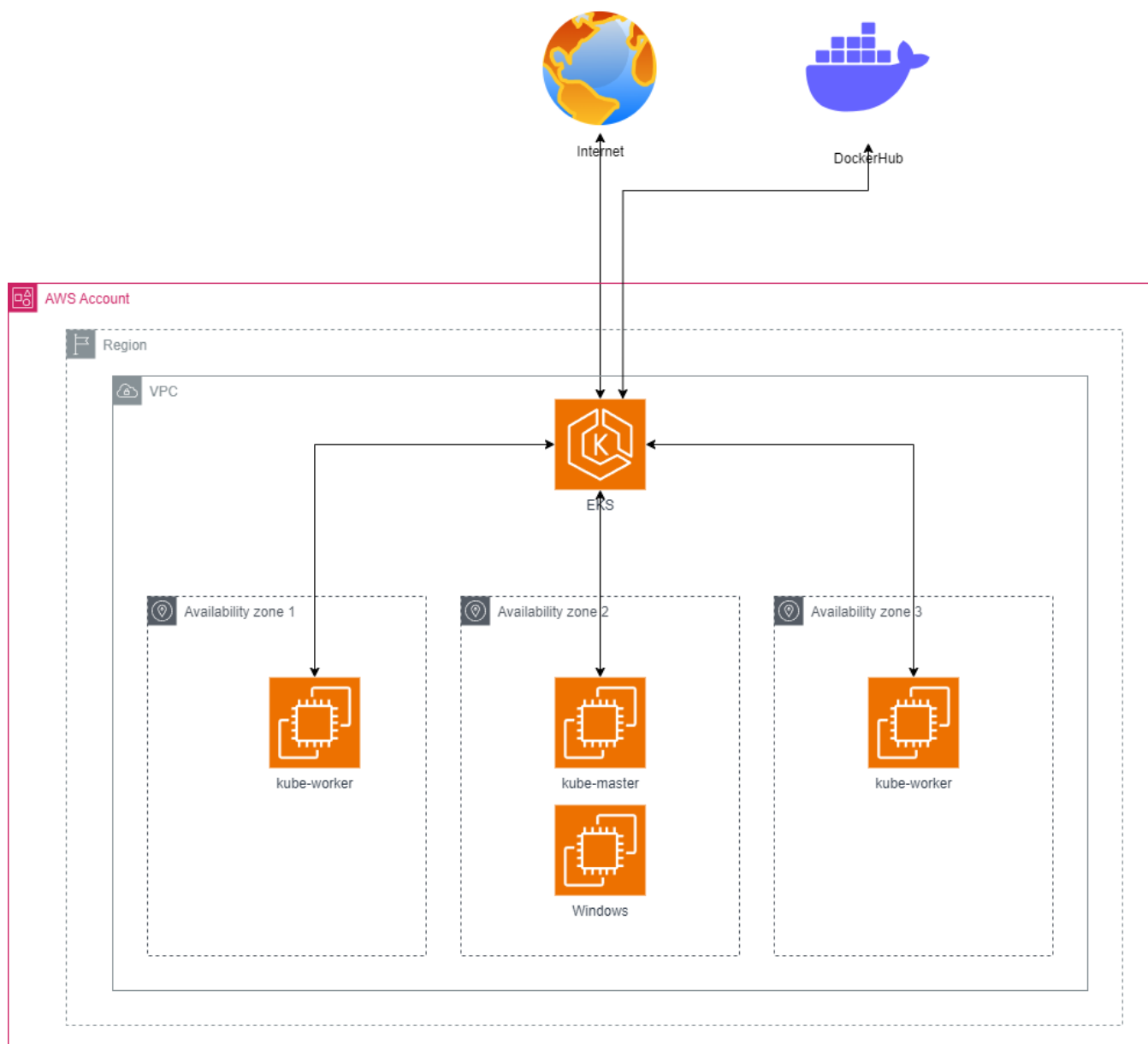


Figura 5.1: Infrastruttura AWS

Capitolo 6

Scalabilità

6.1 Scalabilità EKS

L'approccio **DevOps** (i cui principi sono riportati nell'immagine sottostante), i microservizi e i container hanno portato grande innovazione nel campo della scalabilità. Infatti, dove c'è cultura DevOps, è possibile distruggere e ricostruire un servizio con un click, modificando configurazioni e risorse sottostanti.

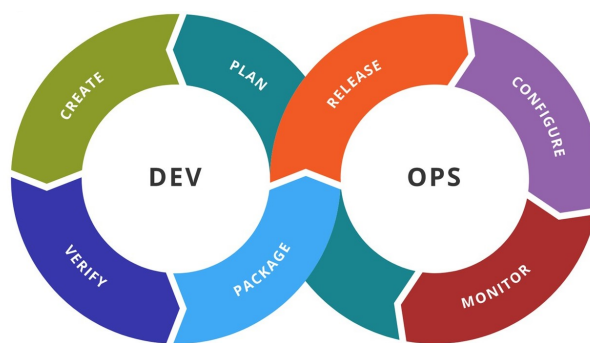


Figura 6.1: Principi di DevOps[12]

Dove possibile, fare scaling orizzontale, ovvero creare copie di un servizio dinamicamente, può permettere di adeguare la capacità del servizio alla richiesta in maniera quasi puntuale, evitando gli sprechi nei casi di picchi. Di seguito, uno schema riassuntivo del funzionamento dello scaling orizzontale.

EKS si integra molto bene con questo approccio e con lo scaling orizzontale, è infatti possibile definire un minimo e massimo di nodi master e worker e una metrica. Tramite l'analisi di particolari metriche (es: uso medio della CPU), EKS riduce o aumenta il numero di nodi (quindi macchine EC2) entro i limiti preimpostati, riuscendo quindi ad adeguarsi alla richiesta in maniera dinamica.

6.2 Scalabilità Volumi

Kuberentes implementa il concetto di volume tramite due risorse:

- Persistent Volume
- Persistent Volume Claim

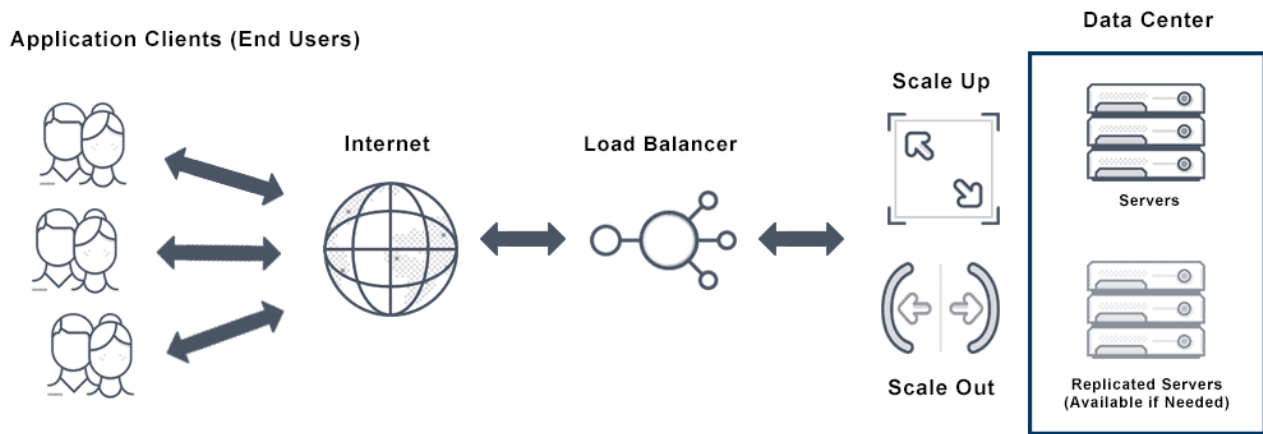


Figura 6.2: Funzionamento dell'autoscaling orizzontale[13]

Un PV identifica una quantità di spazio utilizzabile. Un PVC identifica l'assegnazione di un PV (o parte di esso) a un pod.

Né Kubernetes né AWS consentono di fare autoscaling sui volumi nativamente. Kubernetes però offre una funzionalità che permette di implementare soluzioni terze per risolvere questo problema. I PVC possono essere modificati in termini di dimensioni a runtime, senza doverli deallocare dai pod.

Sulla base di questo principio, si fonda il progetto open source **Kubernetes Volume Autoscaler** [14] che consente di scalare in maniera dinamica i volumi allocati ai pod. Quando un pod sta esaurendo lo spazio a sua disposizione il volume viene aumentato e quando c'è troppo spazio libero viene diminuito.

6.3 Scalabilità Kubernetes

In entrambe le configurazioni del cluster Kubernetes, abbiamo previsto uno scaling automatico dei Pod tramite **HPA** (Horizontal Pod Autoscaler). Questo permette ai deployment SOC, TOOL e APP, ovvero i microservizi che sappiamo essere sottoposti ad un carico variabile perché esposti all'utente, di scalare automaticamente in base al numero di richieste basandosi sulla percentuale di RAM e CPU usata.

L'HPA per il SOC e APP è stato impostato per un minimo di 1 Pod e un massimo di 100 Pods. L'HPA per il microservizio TOOL è impostato invece per un minimo di 1 Pod e un massimo di 10, essendo un microservizio meno utilizzato e non esposto.

Nel caso del deploy su AWS abbiamo anche introdotto un **Node Autoscaler** che scala il numero di nodi in caso il numero di richieste che il cluster deve gestire superi la sua capacità.

6.4 Test di Scalabilità

Abbiamo testato l'applicazione con dei tool appositi per fare stress test di cluster Kubernetes, come ad esempio **K6** [15], ma anche con tool scritti in bash e python che generano un alto numero di richieste per secondo.

I test effettuati e visibili nella timeline delle immagini sottostanti sono in ordine: **spike test**, **stress test** e **breakpoint test**.

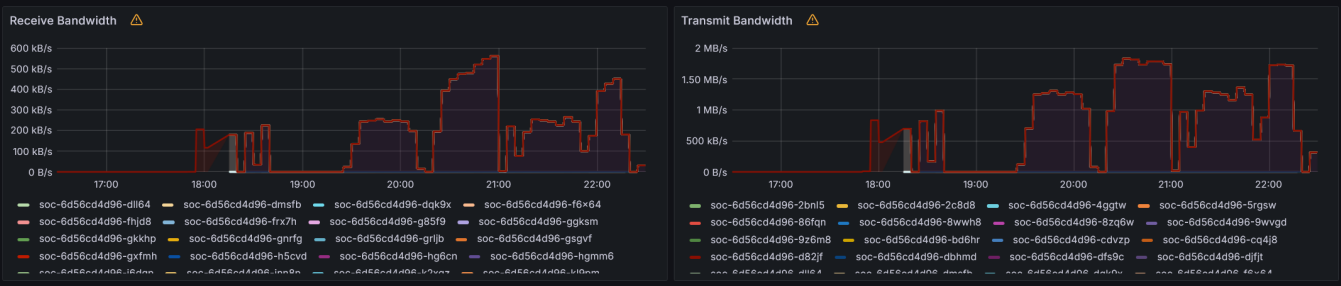


Figura 6.3: Banda Utilizzata durante gli stress test

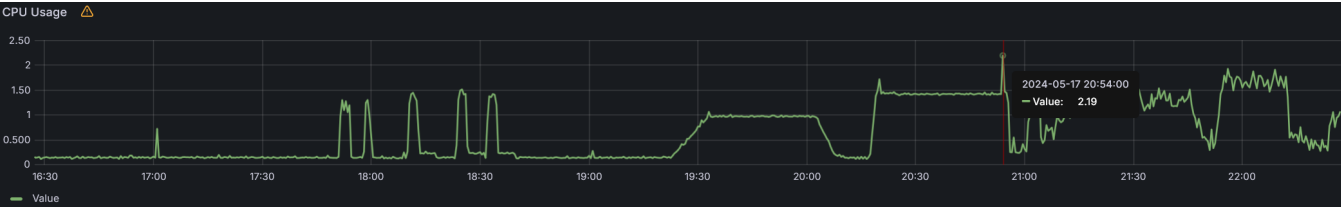


Figura 6.4: CPU Utilizzata durante gli stress test

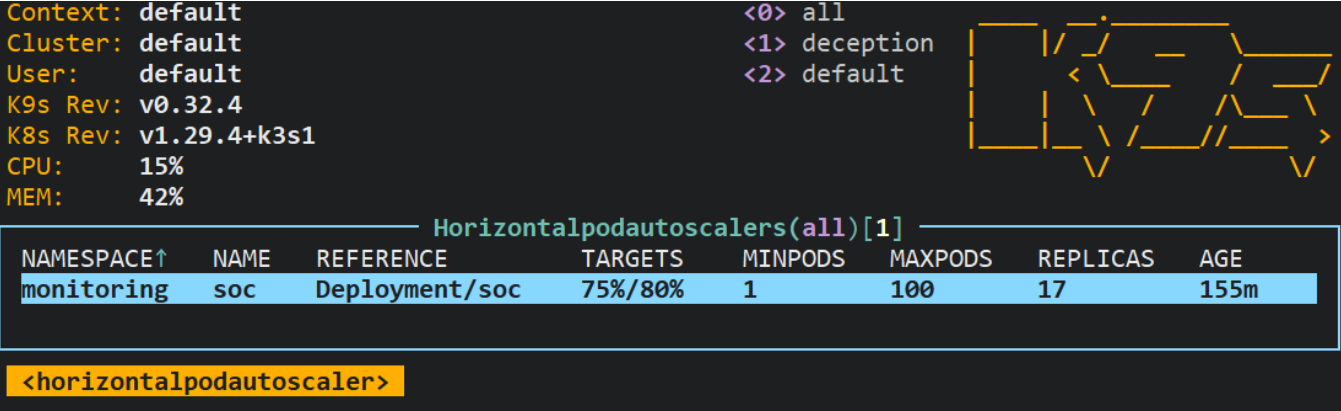


Figura 6.5: Pod autoscalati da HPA

Capitolo 7

Affidabilità

7.1 Affidabilità Kubernetes

Kubernetes si basa su due tipologie di nodi:

- **Master**
- **Worker**

In caso di problemi a un nodo worker, i nodi master provvedono, una volta che gli health check periodici non vengono superati, alla migrazione di tutti i pod sui nodi attivi. Ciò riduce il downtime al minimo e consente di poter resistere a eventi di questo tipo.

Tra tutti i nodi master disponibili ce n'è sempre uno con importanza massima, il leader, che viene eletto dagli altri nodi master tramite un algoritmo interno a Kubernetes. In caso di problemi a un nodo master, i nodi master rimanenti provvedereanno, una volta identificato il problema con il leader, alla rielezione di un nuovo leader.

7.2 Affidabilità AWS

Facendo riferimento all'infrastruttura AWS proposta, è possibile notare che i nodi sono stati posti in diverse Availability Zone. Le Availability Zone sono un concetto di AWS che identifica una serie di datacenter appartenenti a uno stesso fault domain.

Nell'ipotesi che AWS abbia problemi a un datacenter, questi problemi (a meno di cause di forza maggiore) non si possono propagare alle altre AZ di una stessa Region, diminuendo drasticamente il downtime del cluster.

Le region sono una collezione di AZ localizzate in una precisa area geografica, sufficientemente distante dalle altre Region per evitare che problemi di tipo catastrofico o politici possano impattare due Region diverse. Purtroppo EKS non è nativamente implementabile su più region. L'immagine seguente mostra la differenza tra AZ e Region.

Capitolo 8

Dashboard e Frontend

L'interfaccia di MiniSOC è caratterizzata con un layout a pannelli che facilita la navigazione tra le diverse sezioni. Sul lato sinistro della schermata è presente un pannello di navigazione laterale che include le operazioni principali. Cliccando su tali link, si aprirà l'interfaccia della operazione nel pannello centrale. In alto a destra sono presenti dei bottoni per cambiare il tema da chiaro a scuro, gestire le notifiche e la sessione dell'utente.

Per poter accedere alla dashboard l'utente prima viene indirizzato ad una pagina di login nella quale deve inserire il suo username e password, oppure può prima registrarsi o cambiare la password di uno username associato.

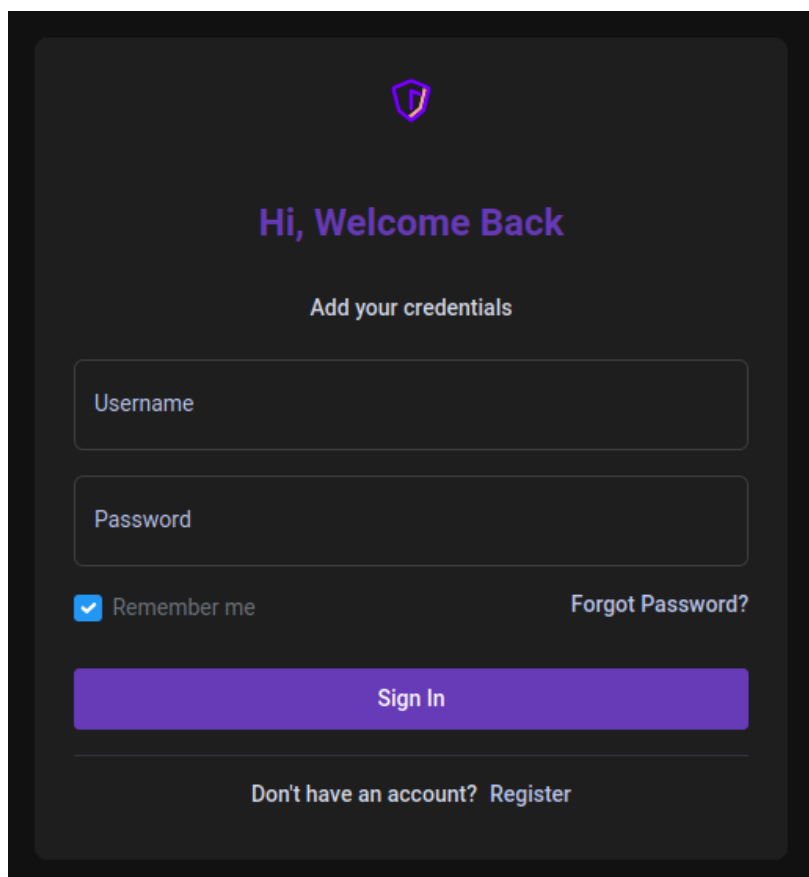
The image shows a login page for the SOC microservice. It features a dark background with a central light gray panel. At the top of the panel is a small shield icon with a red 'X'. Below the icon, the text 'Hi, Welcome Back' is displayed in a bold, dark font. Underneath this, the text 'Add your credentials' is shown in a smaller, lighter font. There are two input fields: 'Username' and 'Password', both with light gray borders and placeholder text. Below the 'Username' field is a checkbox labeled 'Remember me' with a blue checkmark. To the right of the checkbox is a link 'Forgot Password?'. At the bottom of the input section is a large, solid blue button labeled 'Sign In'. Below the button is a horizontal line, and at the very bottom, the text 'Don't have an account? Register' is displayed, with 'Register' being a link.

Figura 8.1: Pagina di login del microservizio SOC

Il microservizio responsabile della dashboard è SOC. Esso è realizzato come applicazione **Flask** e dunque il suo backend è realizzato in Python, mentre la parte di frontend è in **HTML**, **CSS**, **Javascript** e **Jinja**. Le varie dashboard utilizzate in questo MiniSOC vengono renderizzate come iframe.

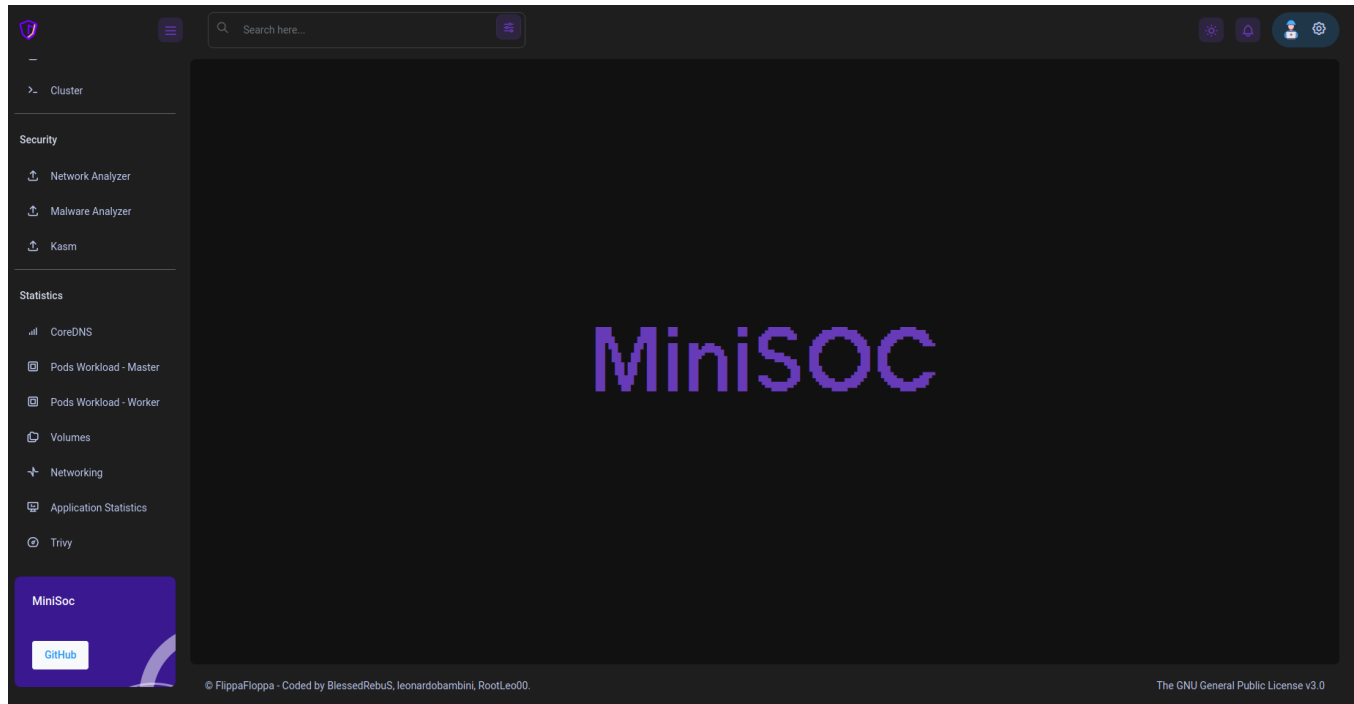


Figura 8.2: Landing Page del microservizio SOC

L'intero progetto è disponibile su GitHub a questo link: <https://github.com/FlippaFloppa/MINISOC>

Bibliografia

- [1] <https://metallb.universe.tf/>, “MetalLB Kubernetes LoadBalancer.”
- [2] <https://k3s.io/>, “K3S.”
- [3] <https://kasmweb.com/>, “KASM.”
- [4] <https://www.wireshark.org/docs/man-pages/tshark>, “tshark.”
- [5] <https://github.com/VirusTotal/yara>, “YARA.”
- [6] IDC, “Worldwide semiannual public cloud services tracker, 2H 2021.”
- [7] M. Nair, “AWS’s Global Infrastructure- How Regions, Zones, and Edge Locations Shape Our Cloud Experience.”
- [8] GovCIO, “What Is Terraform And Why Is It Needed?.”
- [9] C. N. C. Foundation, “How Kubernetes works.”
- [10] <https://helm.sh/>, “HELM.”
- [11] AWS, “Global Infrastructure.”
- [12] C. Comunicazioni, “DevOps, come sviluppare una strategia e i 20 tools più popolari.”
- [13] Avinetworks, “Auto Scaling.”
- [14] Avinetworks, “Volume Auto Scaling.”
- [15] <https://k6.io/>, “K6.”