UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

SISTEMA DE DETECCIÓ D'AMENACES CIBERNÈTIQUES

Arnau Cirera Bosch

Director: Manel Lopez De Miguel | Raül Roca Cánovas
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 17 de Gener de 2024

RESUM

Una infraestructura TIC exposada a Internet necessita protegir-se davant de possibles atacs cibernètics que poden fer parar el servei que dona o comprometre la integritat de les dades que conté. Per tenir un control de la ciberseguretat d'una infraestructura, les organitzacions monitoren els seus sistemes amb solucions de detecció d'amenaces per tal de poder gestionar els incidents que succeeixin.

Aquest projecte pretén desenvolupar una solució unificada i autoescalable de detecció d'amenaces cibernètiques que permeti monitorar una xarxa TIC d'una manera simple i visual per l'usuari. Això implica, per una banda, monitorar diversos aspectes, tals com logs d'aplicacions i dispositius de xarxa, tràfic de xarxa, alertes de sistemes de detecció d'intrusions, modificació d'arxius i esdeveniments de dispositius finals (servidors i ordinadors personals). A continuació se centralitzarà aquesta informació i es filtrarà amb un sistema de detecció d'amenaces cibernètiques basat en regles de detecció, el qual generarà alertes que es visualitzaran en un conjunt de panells de control per poder gestionar les incidències que es puguin ocasionar a la infraestructura.

El sistema es desplegarà utilitzant Kubernetes, una plataforma de codi obert per automatitzar la implementació, l'escalat i l'administració d'aplicacions en contenidors. Això permet desplegar diferents solucions en contenidors de forma automàtica i escalable a una infraestructura al núvol, facilitant així la creació de solucions basades en microserveis les quals necessitin un sistema fiable i escalable de forma eficaç.

El nucli de la detecció d'amenaces serà donat per la solució Wazuh, una plataforma de seguretat gratuïta i de codi obert que unifica les capacitats XDR (Detecció i Resposta Esteses) i SIEM (Gestió d'Informació i Esdeveniments de Seguretat). Conté diferents mòduls els quals aporten diferents funcionalitats al sistema tal com detecció d'amenaces basada en regles, centralització de logs, anàlisi i cerca de text, resposta activa i remota davant d'incidents de seguretat, anàlisi de vulnerabilitats de programari i control de compliment normatiu (tal com PCI DSS, GDPR i CIS, entre altres).

RESUMEN

Una infraestructura TI expuesta en Internet necesita protegerse ante posibles ataques cibernéticos que pueden hacer parar el servicio que presta o comprometer la integridad de los datos que contiene. Para tener un control de la ciberseguridad de una infraestructura, las organizaciones monitorizan sus sistemas con soluciones de detección de amenazas para poder gestionar los incidentes que sucedan.

Este proyecto pretende desarrollar una solución única y autoescalable de detección de amenazas cibernéticas que permita monitorizar una red TI de una forma simple y visual para el usuario. Esto implica, por un lado, monitorizar varios aspectos, tales como logs de aplicaciones y dispositivos de red, tráfico de red, alertas de sistemas de detección de intrusiones, modificación de archivos y eventos de dispositivos finales (servidores y ordenadores personales). A continuación se centralizará esta información y se filtrará con un sistema de detección de amenazas cibernéticas basado en reglas de detección, que generará alertas que se visualizarán en un conjunto de paneles de control para poder gestionar las incidencias que se puedan ocasionar en la infraestructura.

El sistema se desplegará utilizando Kubernetes, una plataforma de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones en contenedores. Esto permite desplegar diferentes soluciones en contenedores de forma

automática y escalable a una infraestructura en la nube, facilitando así la creación de soluciones basadas en microservicios que necesiten un sistema fiable y escalable de forma eficaz.

El núcleo de la detección de amenazas será dado por la solución Wazuh, una plataforma de seguridad gratuita y de código abierto que unifica las capacidades XDR (Detección y Respuesta Extendidas) y SIEM (Gestión de Información y Eventos de Seguridad). Contiene diferentes módulos que aportan diferentes funcionalidades al sistema tales como detección de amenazas basada en reglas, centralización de logs, análisis y búsqueda de texto, respuesta activa y remota frente a incidentes de seguridad, análisis de vulnerabilidades de software y control de cumplimiento normativo (tales como PCI DSS, GDPR y CIS, entre otros).

ABSTRACT

An IT infrastructure exposed to the Internet needs to protect itself against potential cyberattacks that could stop the service it provides or compromise the integrity of the data it contains. To have control over the cybersecurity of an infrastructure, organizations monitor their systems with threat detection solutions in order to be able to manage incidents that occur.

This project aims to develop a unified and self-scalable cyber threat detection solution that allows monitoring of an IT network in a simple and visual way for the user. This involves, on the one hand, monitoring various aspects, such as application logs and network devices, network traffic, alerts from intrusion detection systems, file modification and events from end devices (servers and personal computers). This information will then be centralized and filtered with a cyber threat detection system based on detection rules, which will generate alerts that will be displayed on a set of control panels to be able to manage incidents that may occur in the infrastructure.

The system will be deployed using Kubernetes, an open source platform for automating the deployment, scaling and management of containerized applications. This makes it possible to deploy different solutions in containers in an automatic and scalable way to a cloud infrastructure, thus facilitating the creation of solutions based on microservices that need a reliable and scalable system effectively.

The core of threat detection will be provided by Wazuh solution, a free and open source security platform that unifies XDR (Extended Detection and Response) and SIEM (Security Information and Event Management) capabilities. It contains different modules that bring different functionalities to the system such as rule-based threat detection, log centralization, text analysis and search, active and remote response to security incidents, software vulnerability analysis and compliance control regulatory (such as PCI DSS, GDPR and CIS, among others).

ÍNDEX

ESTAT DE LA SITUACIÓ

1. INTRODUCCIÓ

1.1 Context

Els ciberatacs estan creixent exponencialment en els últims anys. Amb la pandèmia i l'inici del teletreball massiu, aquests ciberatacs s'han incrementat encara més, ja que els ciberdelinqüents tenen més portes d'entrada (i menys controlades) per accedir a una infraestructura d'una organització i aturar el servei o comprometre les dades que s'hi troben distribuïdes. La ciberseguretat és un conjunt de capes format per tecnologia (maquinari i programari) i procediments que una organització implementa al llarg de la seva vida de forma incremental segons les seves necessitats, per aconseguir una protecció contra ciberamenaces que puguin afectar a la integritat, continuïtat i reputació d'una organització.

La seguretat total no existeix, ja que sempre hi haurà un error (humà o tecnològic) que faci vulnerable un sistema. És per això que el control de la ciberseguretat a una entitat s'ha de realitzar amb prevenció, protecció i remediació. Un sistema de vigilància tracta aquestes tres branques, i per això el fa un dels sistemes més importants per controlar la seguretat. En el cas de les tecnologies de la informació, això és causat per un SOC (Centre d'Operacions de Seguretat). Un conjunt de personal expert en ciberseguretat, tecnologia de detecció d'amenaces i procediments de gestió d'incidents, formen un SOC el qual du a terme un seguiment constant de la seguretat d'una infraestructura TIC per tal de detectar possibles amenaces i solucionar-les perquè no afectin la infraestructura.

Actualment, en el mercat trobem moltes solucions que permeten, de diferents maneres, detectar amenaces i gestionar incidents d'una infraestructura TIC, monitorant els diferents sistemes que la componen. Tot i això, aquestes solucions solen ser cares o complexes, el que fa que una gran part de les empreses, en especial les PIMEs (Petites i Mitjanes Empreses), ja que no solen tenir els recursos econòmics o tecnològics (personal tècnic, departament de ciberseguretat, infraestructura complexa de monitorar, etc.).

En la meva empresa dirigeixo el desenvolupament d'un producte de detecció d'amenaces i gestió d'incidents adaptat a la infraestructura que controla per donar a l'usuari el seu propi SOC com a programari, sense la necessitat d'un equip d'experts en ciberseguretat. Apropant així la ciberseguretat a tothom i donant el control de la seva ciberseguretat a PIMEs, que actualment no es poden permetre els costos d'externalitzar el SOC com a servei ni de crear un SOC intern. El nostre producte utilitza una intel·ligència artificial per aprendre del comportament i funcionament de la infraestructura que està monitorant, per adaptar solucions de ciberseguretat a aquella infraestructura de forma automàtica, facilitant així la gestió d'incidents a l'usuari final.

Aquest producte necessita una base tecnològica formada per diferents solucions de detecció d'amenaces i gestió a incidents que s'adaptarà a la infraestructura amb l'ajuda de models d'aprenentatge automàtic. Aquest projecte consisteix en la creació i desplegament d'una base tecnològica, simplificada, però similar a la del producte que desenvolupem en la meva empresa.

1.2 Motivació

La idea d'aquest projecte és causat pel producte que desenvolupem a la meva empresa. Aquest producte el vaig iniciar a dissenyar i desenvolupar jo durant la meitat del transcurs de les meves pràctiques a empresa. Actualment, dirigeixo la direcció del producte i a l'equip que el desenvolupa.

Des de la decisió de valor del producte fins a l'anàlisi de les tecnologies i el disseny de l'arquitectura.

El lligam amb el desenvolupament d'aquest producte m'ha donat la motivació per fer aquest treball. Ja que després d'analitzar les tecnologies i el mercat durant els meus inicis a l'empresa, he trobat el monitoratge i la gestió d'incidents de ciberseguretat com una peça clau en la protecció d'una organització davant d'amenaces cibernètiques. I que en l'actualitat, existeixen forats en el seu mercat, que fa que no totes les empreses, puguin beneficiar-se d'aquest control.

Una gran part de la motivació per aquest projecte i que em va motivar a desenvolupar el producte a la meva empresa, va ser el descobriment de la tecnologia de contenidors. Em vaig iniciar amb aquests conceptes gràcies a l'assignatura de Sistemes Operatius i la introducció a la contenització amb chroot i el descobriment de docker gràcies a companys de la Universitat. Però no va ser fins en iniciar aquest projecte amb l'empresa que vaig entrar més en detall en l'ús d'aquesta tecnologia que no tenia un especial interès en particular a part de la curiositat per nou coneixement. El meu interès profund per aquesta tecnologia va arribar en descobrir l'orquestració de contenidors, en concret Kubernetes. El fet d'aconseguir desplegar un conjunt d'aplicacions connectades entre si a una infraestructura al núvol, sense jo tenir coneixement de desplegament d'infraestructura al núvol i amb coneixements molt bàsics de desplegament d'aplicacions distribuïdes, va fer despertar amb força, un gran interès per aquesta tecnologia i per l'administració de sistemes distribuïts al núvol.

Aquest interès es va ajuntar amb el ja desenvolupat interès per la ciberseguretat, generant-me així, un desig de crear una eina de ciberseguretat escalable de forma automàtica que permeti apropar la ciberseguretat a petites infraestructures dotant-les d'un sistema de detecció d'amenaces cibernètiques Plug & Play centralitzat i fàcil d'utilitzar per a l'usuari.

1.3 Metodologia

2. OBJECTIUS GENERALS

Per aquest projecte, es proposa que la solució a implementar, compleixi amb els següents objectius:

- **Desenvolupar una Plataforma Integral de Ciberseguretat:** L'objectiu principal és crear una plataforma integral que combini múltiples tecnologies de seguretat, com un

XDR, per proporcionar una solució completa de detecció i resposta a amenaces cibernètiques.

- **Automatitzar l'Escalabilitat:** Dissenyar i implementar un entorn Kubernetes que permeti l'autoescalabilitat de la infraestructura i les aplicacions de seguretat. Això assegura que la plataforma pugui gestionar un augment en la càrrega de treball i les dades de manera eficient.
- **Automatitzar el desplegament:** Crear un desplegament àgil i ràpid que permeti el control de versions del sistema.

3. PLANIFICACIÓ

Per dur a terme el desenvolupament del meu treball de recerca, l'he dividit en tres fases:

Fase 1. Desenvolupament de l'arquitectura Cloud

En aquesta fase es desenvoluparà i desplegarà l'arquitectura d'infraestructura Cloud i les aplicacions que formaran part de la solució. Contindrà les següents tasques:

a) Desenvolupament dels manifests declaratius de Kubernetes en un repositori de codi (Github)
b) Desplegament dels manifests amb CI/CD amb l'ús de l'eina ArgoCD
c) Configuració de Wazuh un cop desplegat adaptant-ho al projecte

Fase 2. Desplegament de la solució a la infraestructura a monitorar

En aquesta fase es desplegarà els agents de Wazuh a la infraestructura a monitorar, en aquest cas, un servidor web vulnerable desplegat en contenidors a Kubernetes.

a) Desplegament de l'actiu a integrar a la plataforma
b) Instal·lació i configuració de l'agent de Wazuh

Fase 3. Redacció de la memòria

En aquesta fase es redactarà la memòria del projecte la qual contindrà els següents punts:

- Estat de la situació
- Enginyeria de concepció
- Enginyeria de detall

A continuació es mostra el roadmap que se seguirà pel desenvolupament del projecte:

ACTIVITAT	INICI DEL PLÀ	DURACIÓ DEL PLÀ	SETMANES 1 2 3 4 5 6 7
Desenvolupament Kubernetes	1	1	
Desplegament amb ArgoCD	2	1	
Configuració de Wazuh	2	1	
Desplegament dels actius	3	1	

Instal·lació de Wazuh	4	1
PoC	5	1
Redacció de la memòria	3	5

4. COSTOS

Per calcular els costos del projecte s'ha analitzat l'estimació de costos del mateix proveïdor Cloud. Els costos d'aquesta solució depenen principalment de dos factors:

El primer és la quantitat de recursos que es consumeixen en el cloud, directament proporcional a la capacitat o quantitat dels contenidors desplegats. En aquesta solució en concret, directament proporcional al nombre d'agents que es monitoren, ja que la variació d'aquest nombre, afecta la quantitat i capacitat dels nodes necessaris de Wazuh per processar les dades dels agents.

El segon, és el preu/hora del proveïdor Cloud en els recursos tecnològics a consumir que varia entre les diferents opcions que hi ha actualment al mercat.

A continuació es mostra la taula de costos calculada en la dimensió d'aquest projecte:

Architecture		
Wazuh manager nodes		3
CPU	0.5 cores	
RAM	512MB	
Disk	500MB	
Wazuh Indexer nodes		1
CPU	1 core	
RAM	1,5GB	
Disk	10GB	

Costs	
Kubernetes nodes	78,86 €
Load Balancers	78,88 €
Volumes	1,92 €
Total	159,66 €

Per una infraestructura a producció que monitori una gran infraestructura, els recursos necessaris variarien en quantitat. Per veure una estimació d'aquests costos, he fet una estimació de recursos necessaris per monitorar una infraestructura de 800 ordinadors, 100 servidors i 200 dispositius de xarxa (Que enviarien logs a Wazuh a través d'un servidor Syslog):

Architecture		
Wazuh manager nodes		3
CPU	8 cores	
RAM	16 GB	
Disk	110 GB	
Wazuh Indexer nodes		3
CPU	8 cores	
RAM	16 GB	
Disk	655 GB	

5. ANÀLISI

Analitzant el mercat de la ciberseguretat en PIMEs a Catalunya, arran de la meva experiència en l'empresa donant serveis i productes de ciberseguretat, he pogut veure que, a excepció d'algunes empreses del sector tecnològic, les organitzacions tenen una manca de control de la seva ciberseguretat principalment per tres factors:

- Factor econòmic: Les petites i mitjanes empreses, no es poden permetre una gran quantitat de despeses en ciberseguretat. Però en el mercat, molts dels serveis, llicenciament de programari, maquinari i personal expert, està suposant unes xifres que en comparació amb el volum de l'empresa (tant econòmic com tecnològic) fa que no sigui assequible per l'organització.
- Factor tecnològic: La suma del factor econòmic i el factor operacional, genera una falta de recursos tecnològics no adients a la dimensió de l'organització. Per implementar una infraestructura d'una empresa, es prioritza la funcionalitat i els costos, pel que fa que es generi infraestructures poc fiables que poden ser fàcilment vulnerables, tals com dispositius antics o amb programari desactualitzat o obsolet. També se sol trobar males configuracions dels aplicatius els quals generen errors i mal funcionament que no és arreglat inicialment, ja que, així i tot, poden donar un servei mitjanament continu. Això genera un mal estat de la infraestructura, que fa molt difícil solucionar posteriorment per la falta de recursos, sobretot en personal. En utilitzar softwares de detecció d'amenaces i monitoratge, es generà una gran quantitat d'alertes a causa d'errors d'aplicacions i configuracions no segures de les tecnologies, les quals embruta el sistema de falsos positius que dificulten el control de la ciberseguretat.
- Factor operacional: Aquest nivell de mercat, especialment a Catalunya i Espanya, no té una bona maduresa en tecnologia, ja que es veu la tecnologia, i especialment la ciberseguretat, com un cost, i no com una inversió, perquè no es contempla la necessitat d'aquests recursos pel bon funcionament de l'empresa, fent que el principal factor de decisió de les tecnologies a usar sigui el cost. Això principalment és degut a la mentalitat sud-europea en la qual la protecció no es veu una necessitat, fins que no succeeix la incidència. Mentre que països del centre i nord d'Europa, tenen una cultura més preventiva i veuen la necessitat d'inversió en protecció.

5.1 Anàlisi de les tecnologies

Un Security Operation Center, està format per un conjunt de tecnologies, persones i procediments per dur a terme el control de la ciberseguretat d'una organització. Les tecnologies d'un SOC format un conjunt de solucions de ciberseguretat, implementades en una infraestructura, on-premise, al cloud, o híbrida.

Per aquest projecte, he analitzat les diferents solucions que hi ha en el mercat de la ciberseguretat. La principal solució de ciberseguretat utilitzada per un SOC és un SIEM (Security Information and Event Management), un sistema de seguretat que centralitza dades d'una infraestructura (logs d'aplicació i dispositius, esdeveniments, mètriques) i mitjançant un sistema de regles, correlaciona aquesta informació per detectar possibles amenaces i llançar alertes de seguretat. L'equip del SOC gestiona el SIEM visualitzant les alertes i analitzant la

informació per validar si és una amenaça o és un fals positiu, per així gestionar els possibles incidents de seguretat que rebi l'organització.

Per enviar la informació cap al SIEM fan servir diverses tecnologies depenent de la informació que es vulgui extreure i l'origen d'aquesta. Per dispositius finals, tals com ordinadors i servidors, el més usat avui en dia és un EDR (Endpoint Detection and Response). Un programari que permet recol·lectar esdeveniments del dispositiu, analitzar programari maliciós i processos maliciosos i respondre a les incidències detectades. Pels dispositius de xarxa, tals com tallafocs, switchs, rúters, etc. que permeten extreure logs d'activitat d'aquests, i pels logs d'aplicacions tals com sistemes de còpies de seguretat, antivirus centralitzats i aplicacions webs, se sol emprar un servidor de centralització de logs, que permeti centralitzar-los, filtrar-los i enviar-los al SIEM.

Altres solucions que se solen fer servir són un SOAR (Security Orquestration and Automation Respons), el qual permet crear processos d'automatització de tecnologies per gestionar la resposta a incidents de ciberseguretat, una eina de ticketing, per gestionar i procedimentar els casos que es tracten al SOC de les possibles amenaces, i una eina de Threat Intel·ligence, que permeti correlacionar els elements claus d'una possible amenaça, anomenats Indicadors de Compromís (IoC), amb bases de dades d'altres amenaces passades de tot el món, per poder detectar si l'amenaça és real.

6. INFRAESTRUCTURA

Un dels elements claus per tenir un bon SOC és la infraestructura la qual suportarà aquestes tecnologies. En el meu cas, he utilitzat una infraestructura híbrida, on el nucli resideix en el Cloud. En la infraestructura del client, s'implementaran agents EDR que permetran recol·lectar els esdeveniments i enviar-los al SIEM. El SIEM estarà en el Cloud, hostejada en un clúster de servidors fent ús de la tecnologia de contenidors.

La tecnologia de contenidors, permet desplegar una aplicació continguda en un petit entorn el qual conté l'estructura i llibreries necessàries per executar aquesta aplicació ja provada amb anterioritat, podent així assegurar un desplegament net i fiable a qualsevol entorn que permeti la tecnologia de contenidors.

Kubernetes

La tecnologia Docker, ha sigut una de les tecnologies de contenidors més utilitzada en els últims anys, i que va realitzar un canvi important en el desplegament d'aplicacions distribuïdes. Amb l'arribada del Cloud, la tecnologia Docker s'ha fet servir massivament per moltes empreses que utilitzaven la gran disponibilitat de recursos que ofereix el Cloud per desplegar les seves aplicacions d'una forma eficaç. Això va donar peu a l'origen de noves tecnologies usades per desplegar aplicacions en contenidors de forma distribuïda al Cloud i amb una alta disponibilitat.

La tecnologia per excel·lència que aconsegueix aquest objectiu és Kubernetes. Un projecte de Google creat inicialment per desplegar de forma interna les aplicacions i els serveis d'aquesta empresa, però que l'any 2014, es va alliberar a la comunitat de codi obert.

Kubernetes és una plataforma portable i extensible de codi obert per administrar càrregues de treball i serveis. Kubernetes facilita l'automatització i la configuració declarativa. Compta amb una gran comunitat darrere que ha usat i utilitza Kubernetes per desplegar aplicacions en producció a gran escala per més d'una dècada.

Kubernetes ofereix un entorn d'administració centrat en contenidors. Kubernetes orquestra la infraestructura de còmput, xarxes i emmagatzematge perquè les càrregues de treball dels usuaris no ho hagin de fer. Això ofereix la simplicitat de les Plataformes com a Servei (PaaS) amb la flexibilitat de la Infraestructura com a Servei (IaaS) i permet la portabilitat entre proveïdors d'infraestructura.

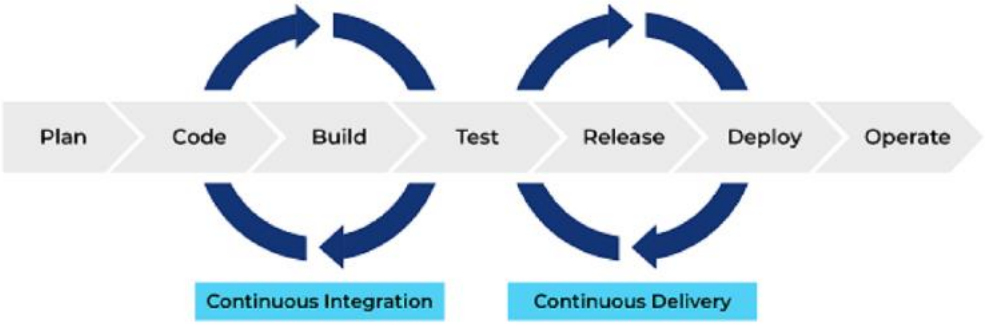
Es detalla l'estructura i detall de Kubernetes a l'Annex 1: Kubernetes.

Argocd

Per desplegar la infraestructura de Kubernetes faré servir manifestos .yaml, que declaren els diferents objectes de Kubernetes a desplegar. Per poder gestionar correctament aquests arxius .yaml faré servir un repositori a Github per controlar les versions que vagi desplegant.

Per poder desplegar una gran infraestructura escalable que permeti el desplegament de més infraestructura de forma automàtica, és necessari desenvolupar una estratègia de desplegament amb CI/C.

Les sigles CI/CD són abreviatures de Continuous Integration / Continuous delivery.Descriu una metodologia per al desenvolupament i distribució d'aplicacions d'alta freqüència d'actualització usant mecanismes totalment automatitzats. Simplificant la integració i les tasques de desplegament dels desenvolupadors, fent possible desplegar actualitzacions diàries amb plena confiança en la correcta funcionalitat d'aquests, proporcionada per l'automatització de les proves.



Com es pot veure a la figura 8.2, el concepte CI/CD s'aplica al cicle de vida de les aplicacions, afegint processos i proves que augmenten la confiança en una versió determinada. A més, assegura que els desplegaments es fan sempre seguint els mateixos passos, utilitzant scripts totalment automatitzats. És important remarcar que les fases del cicle de vida de l'aplicació és un concepte flexible, adaptat a metodologies àgils, sent mal·leable a cada equip de treball segons les necessitats i requisits interns. A continuació, hi ha una descripció de cada pas d'aquesta metodologia:

- **CI**
    - Codi: Els desenvolupadors escriuen codi, afegint algunes funcionalitats noves o solucionant problemes.
    - Build: Es compilen els canvis penjats al repositori remot (si cal).
    - Test: S'executa un conjunt de proves contra la nova versió de codi, d'aquesta manera, podem validar les funcionalitats del programa preestablertes. Hi ha 4 tipus de proves diferents: Unitat (verificar la funcionalitat de les funcions i fragments de codi), Integració (verificar la funcionalitat dels diferents mòduls d'aplicació que treballen conjuntament), Funcional (orientada als requisits

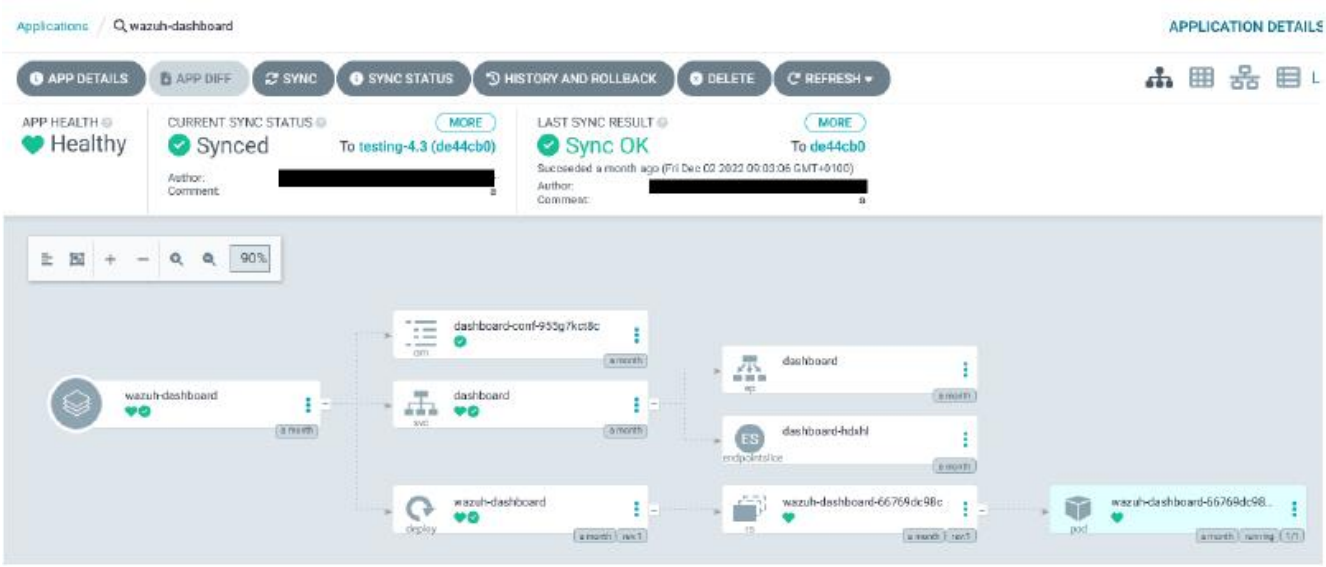
empresarials), E2E (End-to-End, imitar el comportament d'un usuari real, en un entorn d'aplicació complet).

- **CD**
  - Release: Construeix l'executable o artefacte que més endavant es desplegarà en producció, amb les configuracions corresponents establertes.
  - Deploy: Desplega l'executable al servidor de producció.

Com es pot imaginar, el desplegament a Kubernetes no és una tasca fàcil ni trivial. A més, una única càrrega de treball pot implicar moltes definicions d'aplicacions, configuracions, que s'han de desplegar automàticament i fàcilment de manera replicable. A més, els desplegaments haurien de ser controlats per versions, assegurant-se que una versió del repositori determinada, juntament amb totes les definicions implicades, s'hagi implementat amb èxit o no.

Haver de desplegar-se manualment amb confiança en un clúster de Kubernetes de producció es converteix en una feina difícil, que ArgoCD intenta simplificar. ArgoCD es presenta com una eina declarativa de desplegament continu de GitOps per a Kubernetes.

ArgoCD proporciona una capa sobre l'API K8s per lliurar aplicacions a Kubernetes, utilitzant un *CRD*[1] personalitzat anomenat "Aplicació", podríem definir la ubicació juntament amb els manifests que pertanyen a una aplicació determinada. Cada vegada que sincronitzem ArgoCD amb el nostre repositori de codi, detecta els canvis necessaris per fer coincidir la versió desitjada amb la real desplegada. A més, proporciona una representació gràfica amb tots els recursos desplegats i la jerarquia implicada, tal com es mostra a la figura 8.3.



S'ha de tenir compte que fer servir ArgoCD com a solució de GitOps per al desplegament continu implica reestructurar el repositori de codi per evitar definicions YAML duplicades i suport de desplegament multientorn, seguint les millors pràctiques recomanades d'ArgoCD.

L'estructura de fitxers proporciona una capa base que estableix els recursos comuns de desplegament, però permet la definició de "superposicions" per descriure les configuracions d'altres entorns com ara la preparació o la producció. Aquest enfocament de superposició té

[1] Custom Resource Definition: Recurs de Kubernetes que permet la creació de nous recursos per tal de poder afegir integracions a Kubernetes per donar més valor a les tecnologies. Això et permet crear un objecte nou i gestionar-ho de la mateixa manera que els recursos natius de Kubernetes.

avantatges definitius sobre la plantilla perquè limita la substitució a les configuracions parametritzades.

També permet canvis específics de l'entorn sense duplicar fitxers YAML ni utilitzar plantilles. Això vol dir que podríem afegir, eliminar o actualitzar opcions de configuració al manifest de Kubernetes sense bifurcació ni control de font difícil de manejar. Per oferir aquestes funcionalitats, faré servir Kustomize per arrencar un entorn sencer. Permet canvis específics de l'entorn sense duplicar fitxers YAML ni utilitzar plantilles. Per tant, podríem modificar les opcions de configuració del manifest YAML sense bifurcar el codi font en diverses branques per entorn.

Per aquest projecte, he dissenyat una estructura de carpetes per tal de poder desplegar de forma distribuïda les diferents aplicacions i poder afegir configuracions per sobre les aplicacions depenent de l'entorn que es vulgui desplegar sense modificar una base d'aplicacions generalitzada per qualsevol entorn. L'estructura de carpetes és la següent:

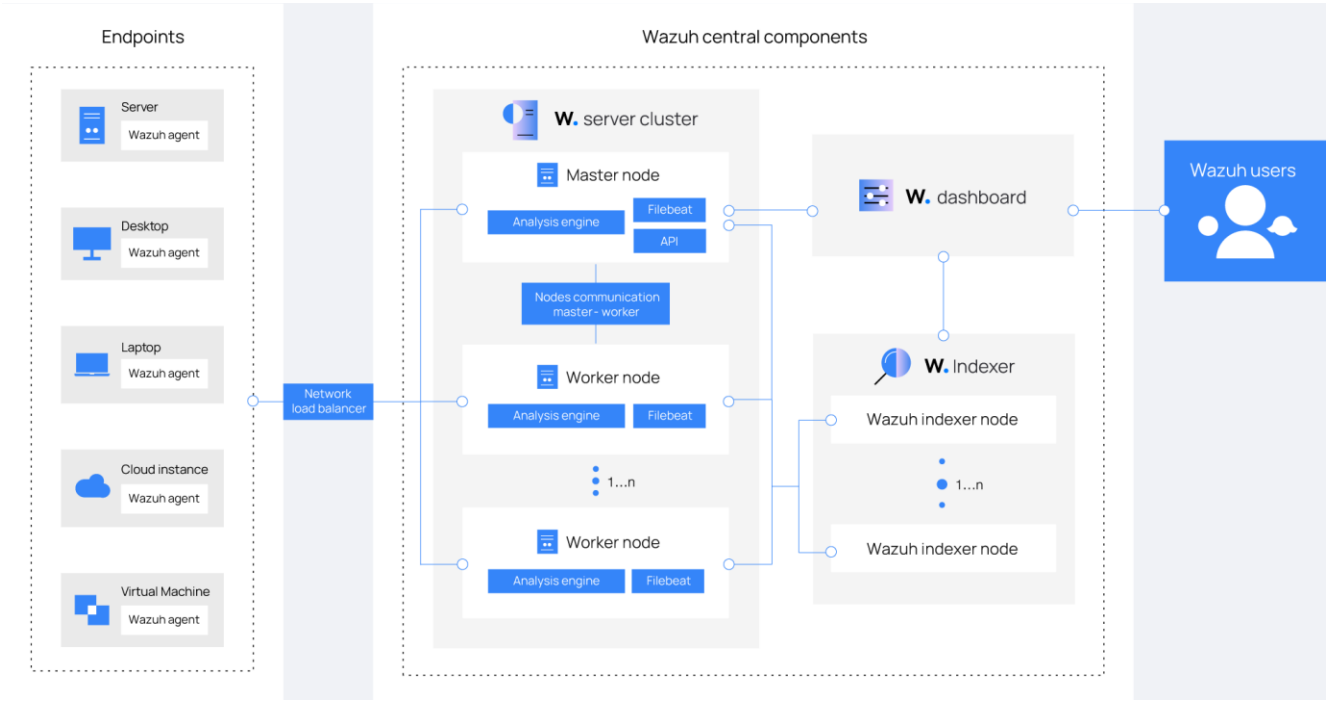
```
├── apps
│   ├── aws
│   │   ├── cluster-autoscaler
│   │   └── ebs-driver
│   ├── victim
│   └── wazuh
│       ├── envs
│       │   ├── eks
│       │   └── local-env
│       └── wazuh
│           ├── base
│           ├── certs
│           │   ├── dashboard_http
│           │   └── indexer_cluster
│           ├── indexer_stack
│           │   ├── wazuh-dashboard
│           │   │   └── dashboard_conf
│           │   └── wazuh-indexer
│           │       ├── cluster
│           │       └── indexer_conf
│           ├── secrets
│           └── wazuh_managers
│               └── wazuh_conf
└── cluster
    ├── argocd
    │   └── apps
    │       ├── aws
    │       │   ├── cluster-autoscaler
    │       │   │   ├── production
    │       │   │   └── testing
    │       │   └── ebs-driver
    │       │       ├── production
    │       │       └── testing
    │       ├── victim
    │       │   ├── production
    │       │   └── testing
    │       └── wazuh
    │           ├── production
    │           └── testing
    └── environments
        ├── eks
        │   ├── production
        │   └── testing
```

```
│       ├──namespaces
│       └──storage_class
└──local
    ├──production
    └──testing
```

Wazuh

Per detectar amenaces cibernètiques a Endpoints (ordinadors personals i servidors) es necessita un programari que pugui extreure esdeveniments, logs i registres d'un dispositiu i filtrar-los per un conjunt de regles que llancin alertes quan es compleix una de les regles. Per això les eines que se solen utilitzar és un EDR (Endpoint Detection and Response) que permet detectar amenaces d'un dispositiu finar i respondre a incidents, i un SIEM (Security Information and Event Management) el qual conté el conjunt de regles que filtraran els esdeveniments per detectar amenaces i avisar als usuaris amb alertes de diferent criticitat amb els valors que facin referència a l'amenaça per tal de poder realitzar una investigació i analitzar si es tracta d'un fals positiu (alerta que es rep de possible amenaça que realment no afecta la infraestructura) o no, i respondre a l'incident si fos necessari.

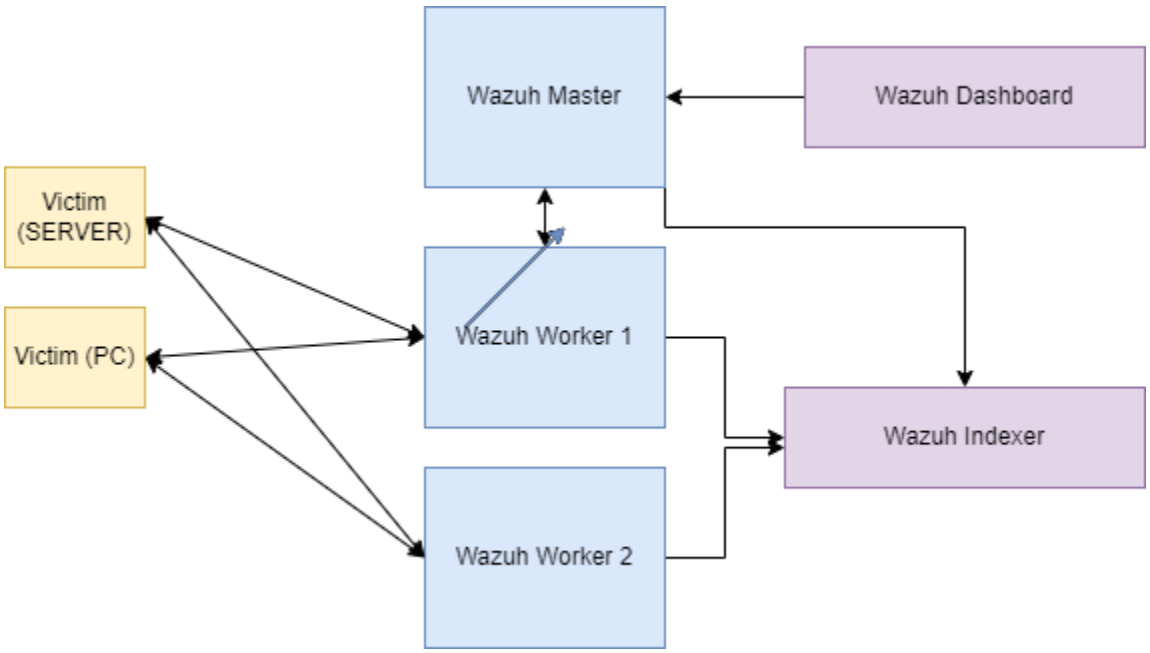
Tant com a EDR com per la plataforma de SIEM faré servir Wazuh, del qual es pot trobar més detall del funcionament i arquitectura de la tecnologia en el Annex 2: Wazuh.



DISSENY

7.2 Disseny de l'arquitectura

Per aquest projecte s'ha dissenyat la següent arquitectura:



7. IMPLEMENTACIÓ

7.1. Desenvolupament dels manifests declaratius de Kubernetes en un repositori de codi (Github)

En el desenvolupament dels manifests declaratius, s'ha utilitzat una estructura escalar en el que s'ha declarat els manifests de les dos aplicacions a desplegar a través del recurs *Application* d'ArgoCD que s'explica en el desplegament dels manifests.

Per l'aplicació de wazuh s'ha utilitzat el repositori que Wazuh ha creat per realitzar una instal·lació de tots els components a un clúster de Kubernetes[2]. Aquest defineix els recursos necessaris per desplegar els diferents components del sistema de Wazuh: el Wazuh Server, el Wazuh Dashboard i el Wazuh Indexer.

Per la creació de la infraestructura a monitorar, s'ha creat un Deployment simple on es declara el contenidor a utilitzar, extret del repositori de contenidors oficial de docker[3]. Aquest *Deployment* té la següent estructura:

[2] [https://github.com/wazuh/wazuh-kubernetes.git]
[3] [https://hub.docker.com/r/vulnerables/web-dvwa]

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: victim-deployment
  namespace: infrastructure
  labels:
    app: victim
spec:
  replicas: 2
  selector:
    matchLabels:
      app: victim
  template:
    metadata:
      labels:
        app: victim
    spec:
      containers:
      - name: victim
        image: vulnerables/web-dvwa
        ports:
        - containerPort: 80
```

També es declara un *service* balancejador de càrrega per poder accedir a la web des d'Internet:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: victim
  namespace: infrastructure
  labels:
    app: victim
spec:
  type: LoadBalancer
  selector:
    app: victim
  ports:
    - name: http
      port: 80
      targetPort: 80
```

Finalment es crearà els .yaml que definiran les *Application* explicades en el pròxim punt:

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: wazuh
  namespace: argocd
spec:
  source:
    path: apps/wazuh/app
    repoURL: git@github.com:ArnolSwet/tfg_arnau-cirera_ub2022-2023.git
    targetRevision: testing
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: wazuh
  project: default
```

7.2. Desplegament dels manifests amb CI/CD amb l'ús de l'eina ArgoCD

Per desplegar l'arquitectura Cloud primer hem d'escollir un proveïdor Cloud el qual crearem una conta per desplegar la nostra infraestructura.
En el meu cas faré servir Amazon Web Services, utilitzant el servei d'Elastic Kubernetes Sevice (EKS) ja que és el que utilitzem a la meva empresa i el que permet una connexió més automàtica amb Kubernetes. En el cas d'un projecte en producció, s'estudiaria les diferents característiques dels proveïdors que trobem al mercat per poder escollir el que més ens ajuda a complir els objectius desitjats.

1. Instal·lació de AWS CLI

L'AWS Command Line Interface (CLI) és una eina poderosa que et permet interactuar amb els serveis d'AWS directament des de la línia de comandes. És particularment útil per a automatitzar tasques repetitives a AWS, ja que permet executar scripts i comandes per a gestionar recursos. L'ús de l'AWS CLI és clau per a la creació de clusters Elastic Kubernetes Service (EKS), ja que permet una gestió més fina i un control més gran sobre la configuració, com per exemple, la creació i configuració de Virtual Private Clouds (VPCs) i rols d'Identity and Access Management (IAM). Això facilita una integració més eficient i segura dins de l'ecosistema d'AWS.

Per instal·lar l'eina en Windows hem de realitzar els següents passos:

I. Executar l'instal·lador MSI de l'eina, per això podem utilitzar la comanda msiexec en una consola PowerShell.

```
msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

II. Un cop seguit els passos de l'instal·lador, verifiquem la instal·lació de l'eina.

```
aws --version
```

sortida d'instal·lació correcte:

```
aws-cli/2.11.27 Python/3.11.3 Windows/10 exe/AMD64 prompt/off
```

III. Si la instal·lació s'ha realitzar correctament, connectarem la CLI amb el nostre compte d'AWS.

```
aws configure
```

ens demanarà l'identificador (ID) de la clau d'accès i la clau secreta. A continuació ens demanarà la regió per defecte, en el nostre cas eu-central-1 (Frankfurt), ja que és la més propera amb compatibilitat amb totes les funcionalitats necessàries per la creació del clúster.

2. Creació de VPCs i Subxarxes

Perquè el clúster de Kubernetes pugui connectar-se entre si i amb Internet, necessita una Xarxa Virtual Privada (Virtual Private Cloud a AWS) i subxarxes tant internes com externes.

I. Utilitzem la següent comanda *create-vcp* per crear una VPC amb el bloc CIDR de IPv4 especificat i les següents etiquetes:

```
aws ec2 create-vpc \
--cidr-block 20.0.0.0/16 \
--query Vpc.VpcId --output text \
--tag-specification ResourceType=vpc, Tags=[{Key=Name,Value=tfg-vpc},
{Key=kubernetes.io/cluster/tfg-testing,Value=”Shared”}]
```

II. Ens retornarà l'identificador de la vpc, en el nostre cas *vpc-07fe01f1f19ae4d75.* A continuació es crearà una subxarxa privada i una subxarxa pública:

```
aws ec2 create-subnet \
--vpc-id vpc-07fe01f1f19ae4d75\
--cidr-block 20.0.1.0/24 \
--availability-zone eu-central-1a \
--query Subnet.SubnetId --output text \
--tag-specification ‘ResourceType=subnet, Tags=[{Key=Name,Value=tfg-private-
subnet}, {Key=kubernetes.io/role/internal-elb,Value=”1”},
{Key=kubernetes.io/cluster/tfg-testing,Value=”Shared”}]’

aws ec2 create-subnet \
--vpc-id vpc-07fe01f1f19ae4d75 \
--cidr-block 20.0.101.0/24 \
--availability-zone eu-central-1a \
--query Subnet.SubnetId --output text \
--tag-specification ‘ResourceType=subnet, Tags=[{Key=Name,Value=tfg-public-
subnet}, {Key=kubernetes.io/role/elb,Value=”1”},
{Key=kubernetes.io/cluster/tfg-testing,Value=”Shared”}]’
```

III. Tot seguit, creem una porta d'enllaç a internet per tal de que la subxarxa pública pugui estar connectada i l'afagirem a la nostra VPC:

```
aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId --
output text
```

```
aws ec2 attach-internet-gateway --vpc-id vpc-07fe01f1f19ae4d75 --internet-gateway-id igw-0250aa5c22bbbfbac
```

IV. Creem una taula d'enrutament personalitzada per la subxarxa pública mitjançant la comanda *create-route-table.* Després creem una ruta en la taula que permeti tot el tràfic IPv4 a la porta d'enllaç.

```
aws ec2 create-route-table --vpc-id vpc-07fe01f1f19ae4d75 --query RouteTable.RouteTableId --output text
aws ec2 create-route --route-table-id rtb-07e3fdf175103c17d --destination-cidr-block 0.0.0.0/0 --gateway-id igw-0250aa5c22bbbfbac
```

V. Finalment associem la taula d'enrutament a la subxarxa pública:

```
aws ec2 associate-route-table --route-table-id rtb-07e3fdf175103c17d --subnet-id subnet-009ffadfa24960c82
```

VI. Els nodes del clúster es mantindran en la subxarxa privada per protegir els servidors d'accessos indeguts, ja que amb l'ús de Kubernetes, no necessitarem accedir als servidors. Tot i així, necessitem crear un accés de sortida del tràfic a Internet perquè es puguin descarregar les imatges dels contenidors. Per això, primer crearem una porta d'enllaç NAT pels recursos. Creem una direcció IP elàstica per la porta d'enllaç i la porta d'enllaç, especificant la direcció IP creada:

```
aws ec2 allocate-address --domain vpc --query AllocationId --output text
aws ec2 create-nat-gateway --subnet-id subnet-0c9b9ee248c5d32bb --allocation-id eipalloc-0a654bda147cf9077
```

VII. Després creem una taula d'enrutament per la subxarxa privada i una ruta en aquesta que permeti tot el tràfic IPv4 a la porta d'enllaç NAT:

```
aws ec2 create-route-table --vpc-id vpc-07fe01f1f19ae4d75 --query RouteTable.RouteTableId --output text
aws ec2 create-route --route-table-id rtb-0f3c8cd18bdb52010 --destination-cidr-block 0.0.0.0/0 --gateway-id nat-0e2c3be99b41df28f
```

VIII. Finalment associem la taula d'enrutament a la subxarxa privada:

```
aws ec2 associate-route-table --route-table-id rtb-0f3c8cd18bdb52010 --subnet-id subnet-0c9b9ee248c5d32bb
```

3. Creació dels rols de IAM

Es crearà tres rols de IAM (Identity Access Management) per poder utilitzar amb el cluster de Kubernetes i poder executar accions de la API de AWS. Per fer-ho es redacta les polítiques en format JSON (Aquestes polítiques es troben a la carpeta *policies* del codi entregat):

I. *eks-cluster-role-trust-policy.json*: Política de gestió interna del clúster EKS.
II. *iam_policy_autoscaler.json*: Política de gestió dels grups d'autoescalatge per la implementació de l'aplicació *Cluster Autoscaler.*

Es crea els rols utilitzant les polítiques descrites amb la comanda *create-rol*:

```
aws iam create-role --role-name amazonClusterRoleEKS \
--assume-role-policy-document file://".\eks-cluster-role-trust-policy.json"

aws iam create-role --role-name clusterAutoscalerRole \
--assume-role-policy-document file://".\iam_policy_autoscaler.json"
```

4. Instal·lació de Kubectl

Per aplicar els manifestos de Kubernetes i gestionar el cluster, s'utilitza l'API del clúster de Kubernetes. Per facilitar-ne la gestió, s'utilitza una CLI anomenada *kubectl* que permet fer qualsevol acció en un clúster de Kubernetes.

Per instal·lar-la, el més comú és descarregar i executar el seu binari. Si disposem de l'eina *curl* el podem descarregar amb la següent comanda:

```
curl -LO https://dl.k8s.io/release/v1.29.0/bin/windows/amd64/kubectl.exe
```

Un cop descarregat, validem que funciona correctament executant la següent comanda:

```
kubectl version –client
```

5. Creació del clúster EKS

Finalment es crea el clúster utilitzant la CLI d'AWS i afegint els rols, VPCs i etiquetes necessàries:

```
aws eks create-cluster --region eu-central-1 --name tfg-testing --kubernetes-version 1.28 \
        --role-arn arn:aws:iam::265270975032:role/amazonClusterRoleEKS \
        --resources-vpc-config subnetIds=subnet-0c9b9ee248c5d32bb,subnet-0306627d0db058127
```

La creació del clúster EKS sol trigar uns minuts, un cop estigui creat, descarregarem l'arxiu d'autenticació del clúster (kubeconfig) per connectar-se a la API del clúster i així, poder utilitzar kubectl:

```
aws eks update-kubeconfig --region eu-central-1 --name tfg-testing
```

Si s'ha creat correctament, al finalitzar podrem executar la comanda *kubectl get ns* per poder veure si estem connectats correctament. Amb això mostrarem els *namespaces* creats al cluster:

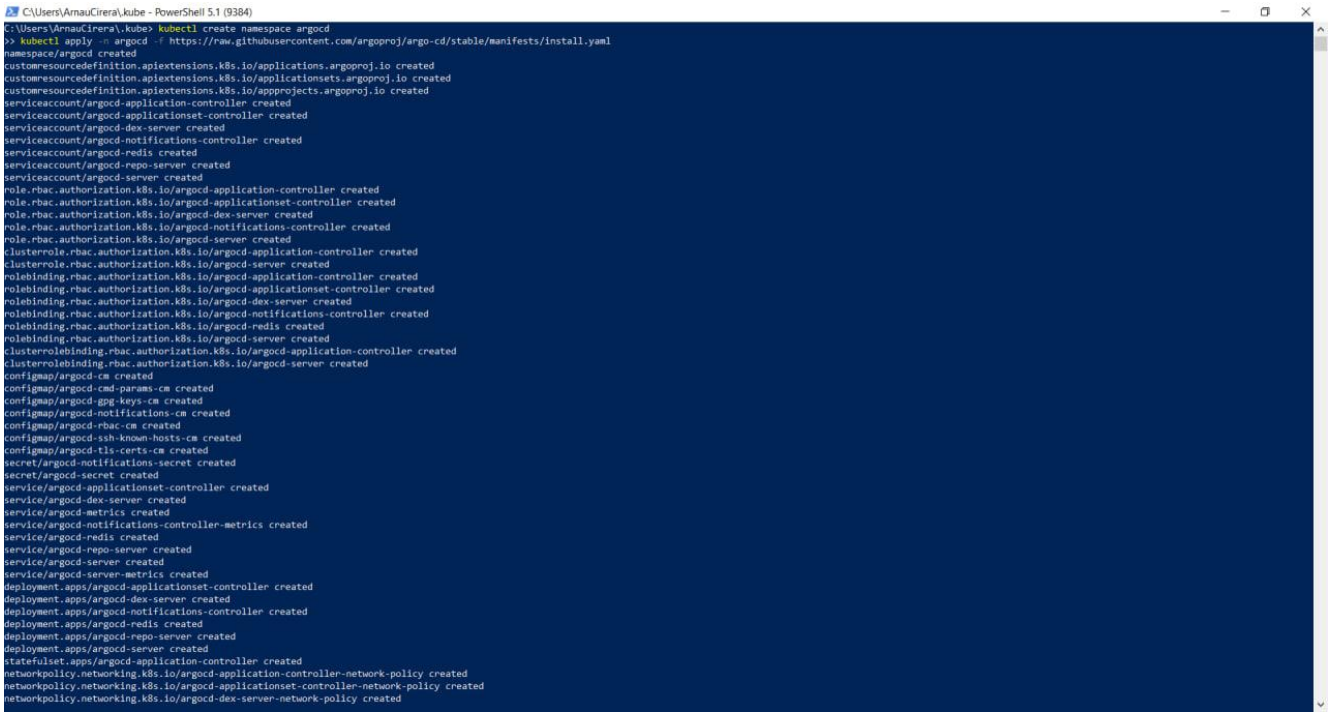
Si es mostra els namespaces per defecte ens indicarà que s'ha creat correctament el cluster.

6. Desplegament d'ArgoCD

A continuació instal·larem ArgoCD. Per això, si seguim la seva documentació oficial, ens indicarà que primer s'ha de crear el *namespace* on anirà els recursos d'ArgoCD i a continuació aplicarem els diferents manifests que ArgoCD té automatitzats en un repositori amb les següents comandes:

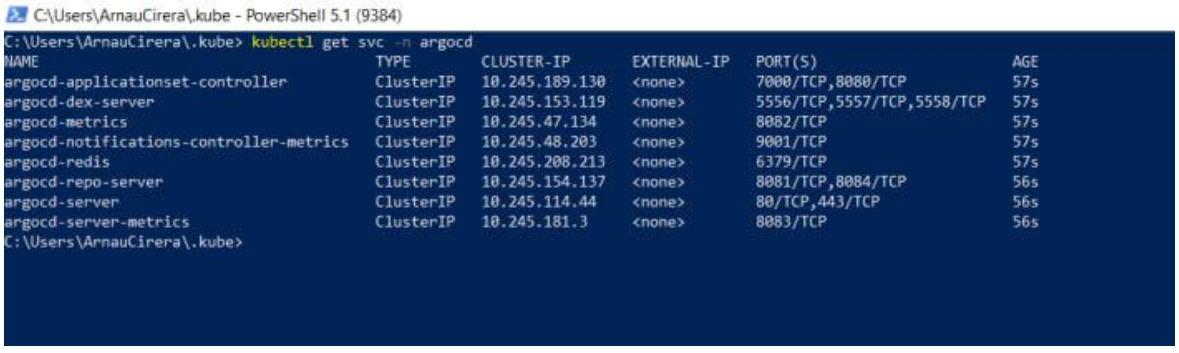
```
kubectl create namespace argocd
```

```
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-
cd/stable/manifests/install.yaml
```



Això crearà els recursos d'ArgoCD necessaris i es mostrarà quan s'han creat.

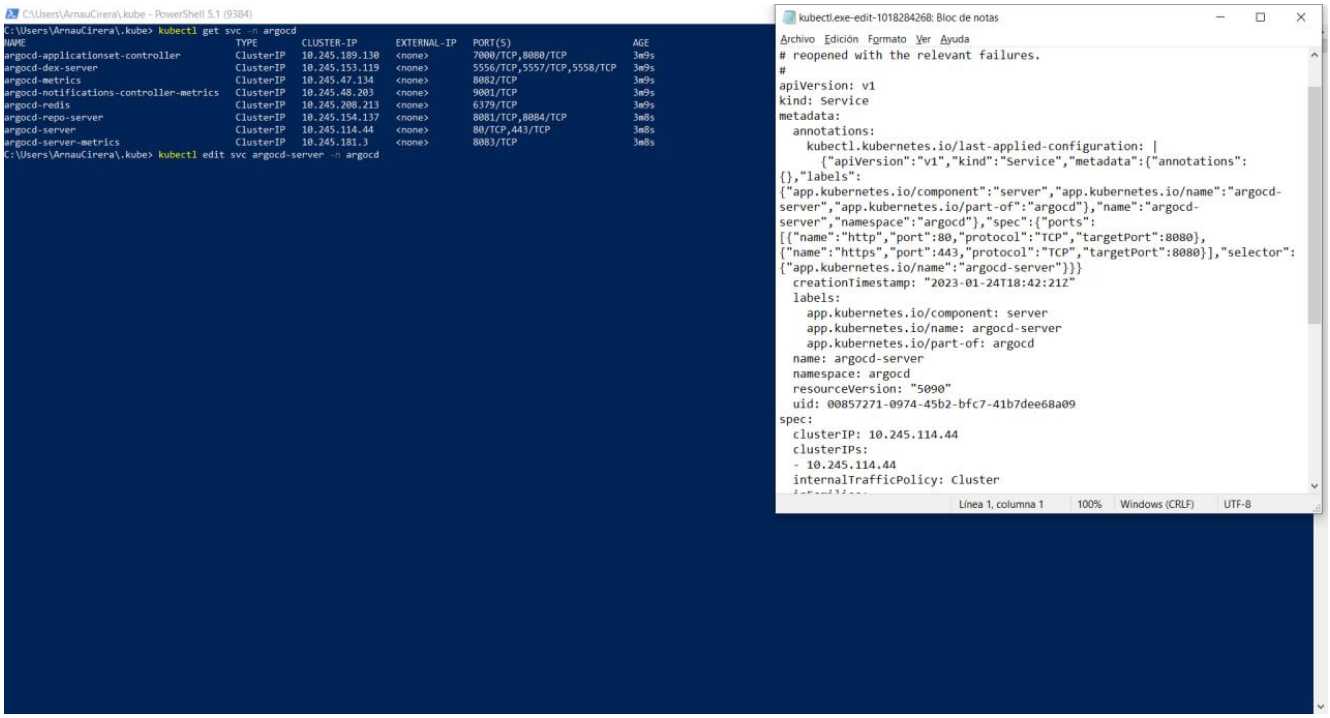
Tot seguit, visualitzarem els serveis creats per ArgoCD, per buscar el servei responsable de donar accés a la interfície gràfica, per així, poder controlar els desplegaments d'una forma més visual i no per terminal.



El servei desitjat és el servei argocd-server. Haurem de modificar aquest servei per tal de canviar de tipus ClusterIP a LoadBalancer, per això ho editarem amb l'eina de kubectl:

```
kubectl edit svc argocd-server -n argocd
```

Això ens obrirà un editor de text amb el manifest del servei per tal d'editar-ho:

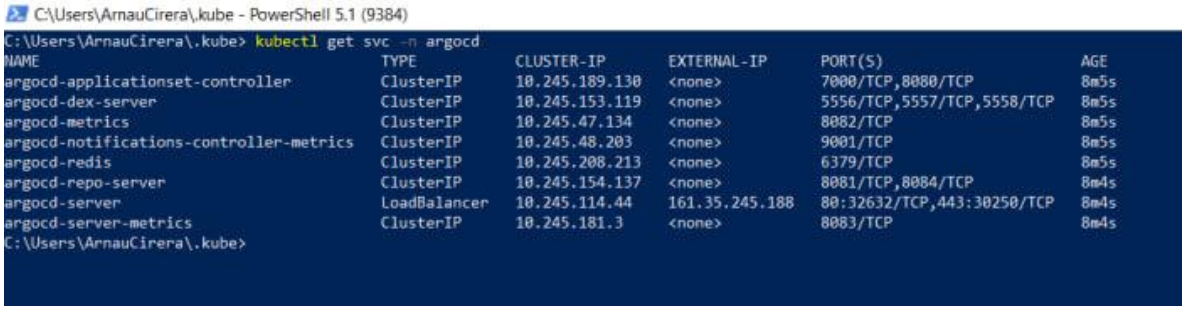


Haurem d'editar l'arxiu canviant el valor de *type* de *ClusterIP* a *LoadBalancer:*

```
  type: LoadBalancer
status:
  loadBalancer: {}
```

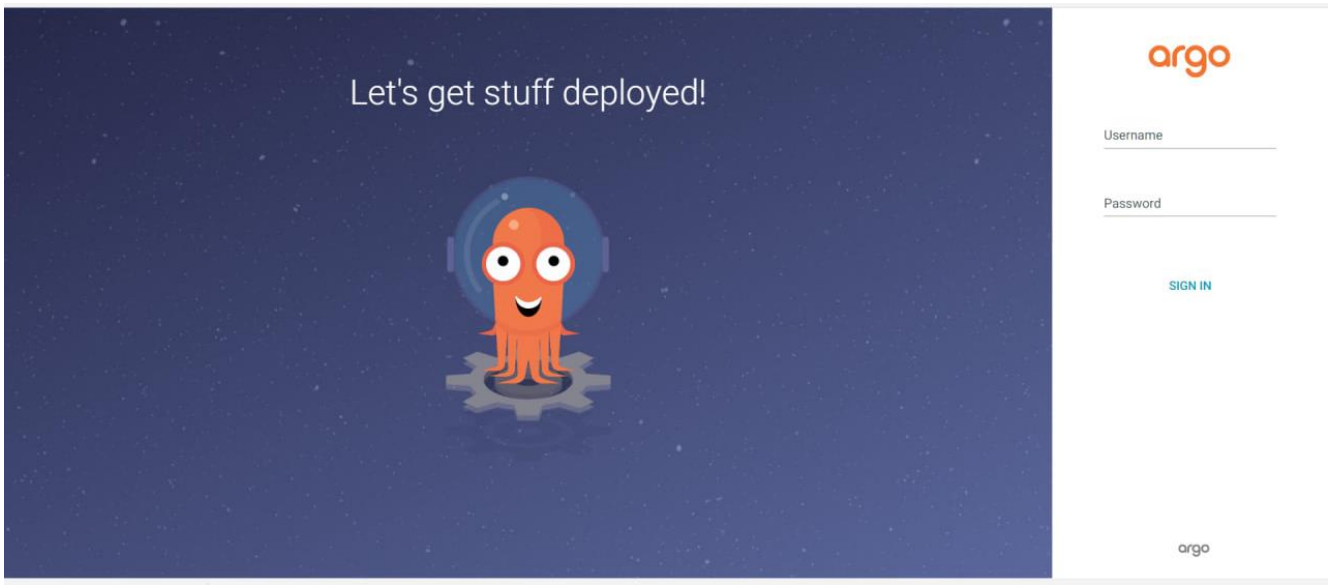
A continuació guardarem l'arxiu i tancarem l'editor de text. Automàticament els canvis es modificaran en el manifest.

Tornem a visualitzar els serveis per veure el canvi:



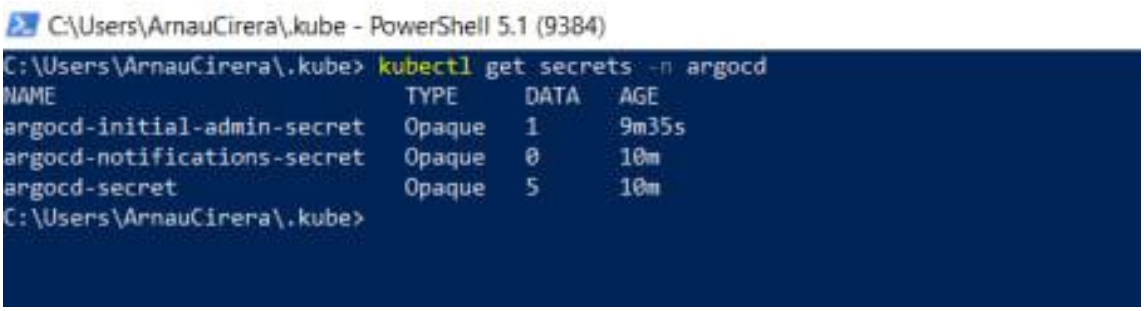
Com es pot veure, el servei ha canviat a tipus LoadBalancer i ens ha creat una adreça IP (en el proveïdor Cloud) per tal de poder accedir.

Ara ja podem accedir per un navegador, el qual ens portarà al inici de sessió de Argocd:



Per saber les credencials d'accès a ArgoCD, haurem de visualitzar els secrets d'Argo. Per fer això utilitzem la comanda:

```
kubectl get secrets -n argocd
```



El que volem visualitzar és el anomenat *argocd-initial-admin-secret* el qual conté les credencials per accedir amb la conta administrador, les quals es generen automàticament al crear les aplicacions. Per visualitzar-ho utilitzarem la comanda:

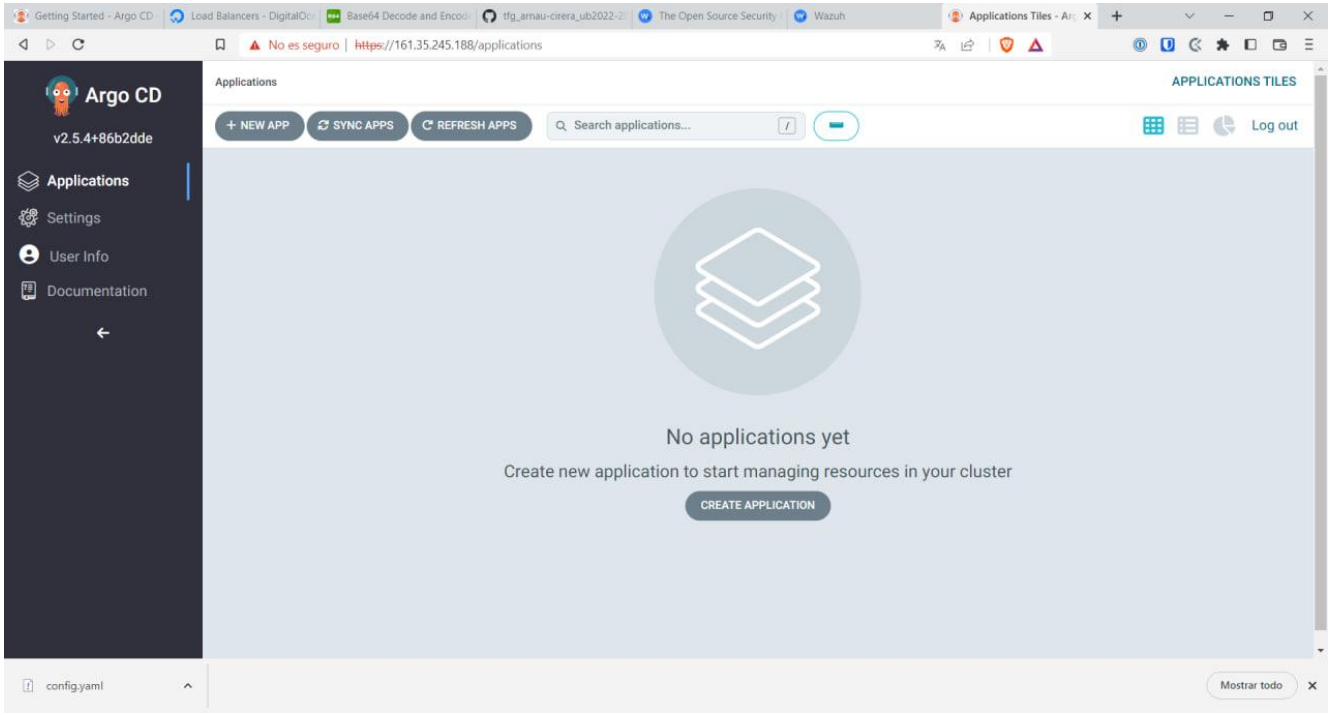
```
kubectl get secret argocd-initial-admin-secret -n argocd -o yaml
```

El qual permet visualitzar la definició d'un secret en format .yaml, per tal de visualitzar la contrasenya en base64(Ja que Kubernetes codifica tots els secrets per emmagatzemar-ho):

```
C:\Users\ArnauCirera\.kube - PowerShell 5.1 (9384)
C:\Users\ArnauCirera\.kube> kubectl get secrets -n argocd
NAME                          TYPE     DATA   AGE
argocd-initial-admin-secret   Opaque   1      9m35s
argocd-notifications-secret   Opaque   0      10m
argocd-secret                 Opaque   5      10m
C:\Users\ArnauCirera\.kube> kubectl get secret argocd-initial-admin-secret -n argocd -o yaml
apiVersion: v1
data:
  password: eTVKLWxRZU1xNUFrN25MNQ==
kind: Secret
metadata:
  creationTimestamp: "2023-01-24T18:42:57Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "5377"
  uid: dbecf328-a3ea-4bf8-a26e-07c9bfa59645
type: Opaque
C:\Users\ArnauCirera\.kube>
```

Utilitzant un decodificador de base64, podem veure que la contrasenya és: y5J-lQeMq5Ak7nL5

Amb això ja podrem accedir al portal d'ArgoCD, el qual veurem la següent pantalla d'inici:



Un cop creat el clúster de Kubernetes i instal·lat ArgoCD per poder desplegar l'infraestructura de forma automàtica declarant-la amb codi en un repositori tal com hem fet al punt A, procedirem a crear les *Applications* encarregades de desplegar les diferents parts de la infraestructura.

Una *Application* és un objecte de Kubernetes creat gràcies al CRD de ArgoCD, associat a un repositori, una carpeta mare i una branca el qual escoltarà els canvis efectuats per el desenvolupador, i interpretarà els manifests .yaml per declarar la infraestructura necessària a un clúster i *namespace* de Kubernetes determinat.

Amb l'ajuda d'aquest objecte, podríem desplegar tota la infraestructura amb una sola aplicació per tot el repositori, el que ens permet replicar tota la infraestructura només amb una comanda de l'estil:

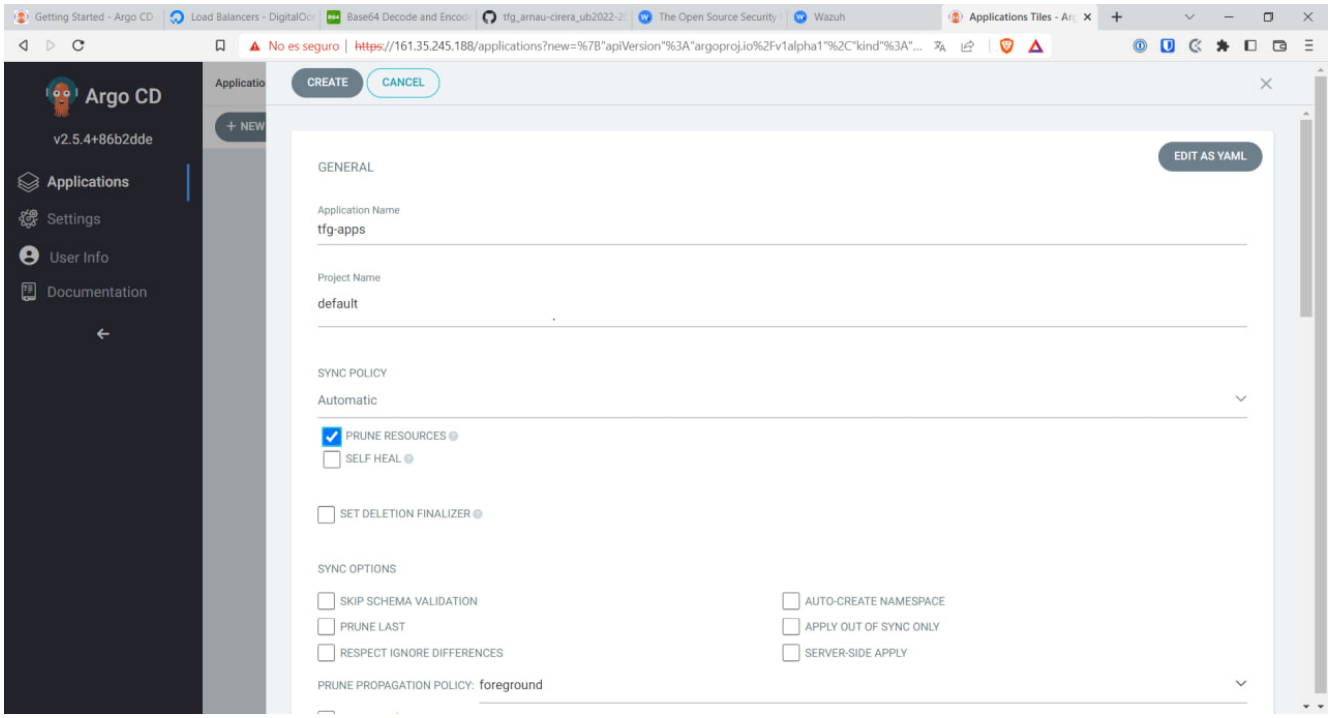
```
kubectl apply -f application.yaml -n <<namespace>>
```

En el meu cas, distribuiré la infraestructura en diferents *Applications*. No només per tenir una visualització de la infraestructura més clara, ja que amb una solució es veuria tots els recursos del clúster en un sol esquema, sense diferenciació clara de les aplicacions que conté o els *namespaces* als qual pertanyen, sinó que també obtenim un millor control de les diferents aplicacions del projecte (Wazuh, Servidor Victima i altres aplicacions que es podrien afegir per donar més valor a la solució). Podent així, realitzar canvis en aplicacions determinades sense perill de afectar a les altres aplicacions en el moment de realitzar una sincronització.

També s'ha de tenir en compte, que una *Application* està associada a un sol namespace, per tant, tota la infraestructura del repositori es desplegaria en un sol *namespace*.

Per obtenir el millor control tant de la totalitat de la infraestructura i la individualitat de les aplicacions, la millor opció és crear una aplicació, que desplegui les altres aplicacions, donant així els dos controls.

Primer de tot es crearà la *Application* mare, la qual anomenarem tfg-apps. Clicarem al botó "create application", el qual ens obrirà la següent pantalla:



Definirem el nom, el projecte (en el nostre cas el default, ja que no realitzem més d'un) i triarem la política de sincronització. Això especifica com es produeix una sincronització:

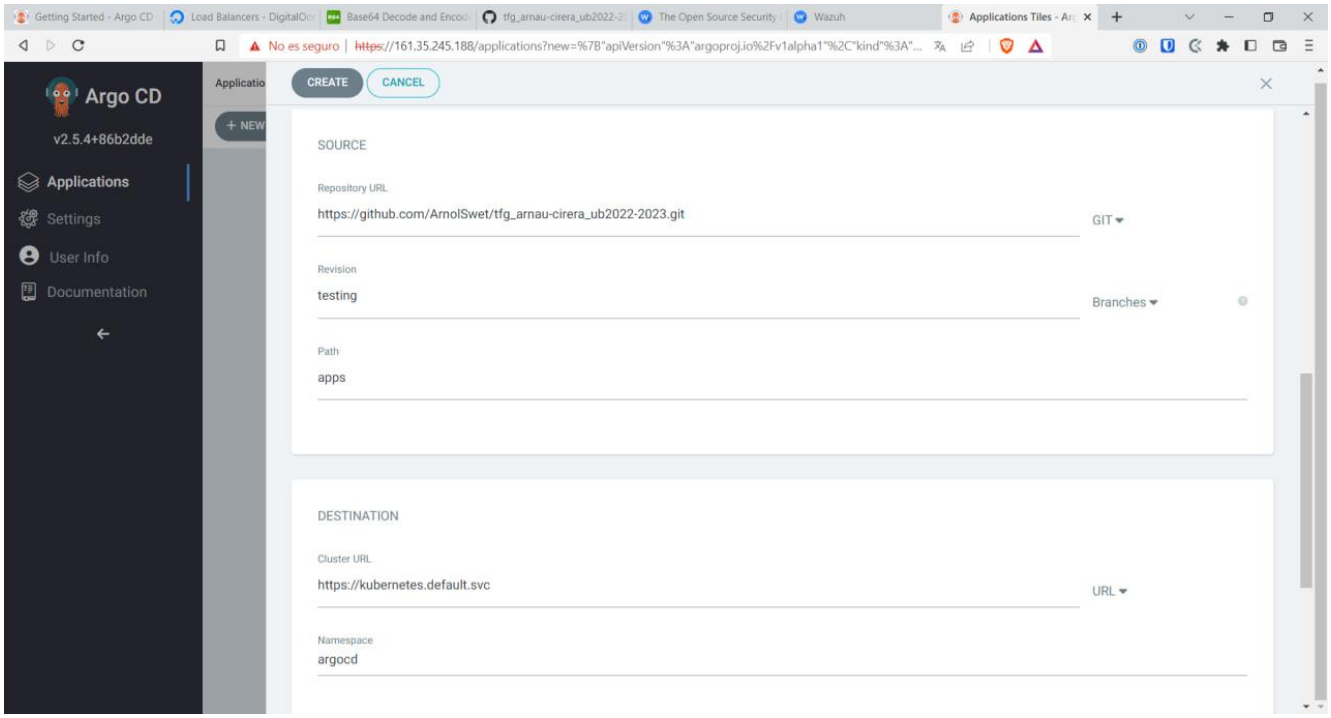
- Manual: La *Application* reconeix els canvis a realitzar i mostra els elements que es canviaran com a no sincronitzats mostrant la diferenciació dels manifests actuals amb els plantejats, però no s'aplicaran els canvis a Kubernetes fins que l'usuari ho aprovi.

- Automàtic: La *Application* reconeix els canvis a realitzar i els aplica directament a Kubernetes sense la intervenció de l'usuari.

Ja que aquesta *Application* només controlarà altres *Applications*, que són objectes no relacionats amb infraestructura del núvol o relacionat amb els *pods*, no afectarà a la infraestructura fins que les altres *Applications* es sincronitzin.

També definirem si volem purgar els recursos al sincronitzar, és a dir, eliminar els recursos que ja no s'utilitzin quan sincronitzem.

Podem definir altres opcions de sincronització que no entrarem en aquest projecte ja que no les utilitzarem però permeten automatitzar més el desplegament, com crear automàticament els *namespaces* on es defineixi recursos de Kubernetes que no estiguin ja creats o aplicar canvis només als recursos que no estiguin sincronitzats.



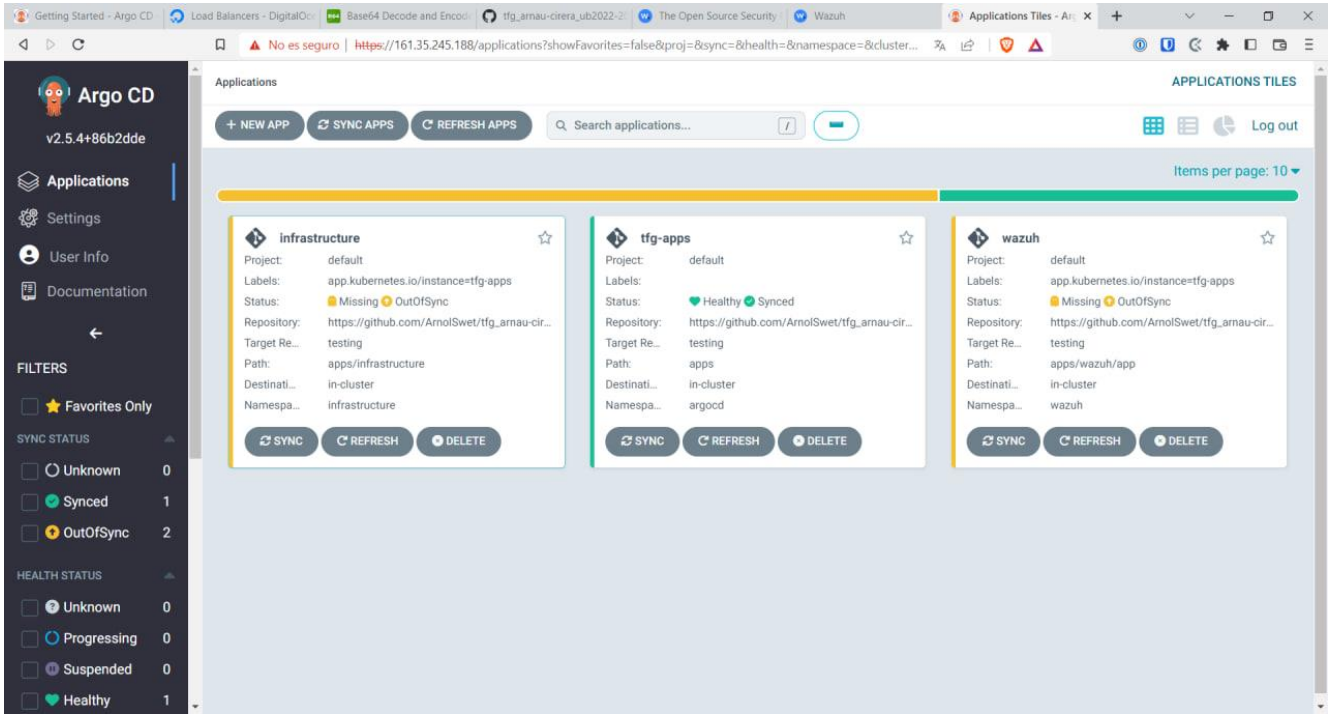
Després indiquem la URL del repositori on tindrem el codi, la branca a la que escoltarà els canvis que es realitzin i el camí a la carpeta que tindrà els arxius a implementar.

Finalment indiquem la URL del clúster destí i el *namespace* on es desplegarà els recursos a implementar. En el nostre cas la URL és kubernetes.default.svc. Aquesta URL és un subdomini que només té el DNS d'aquest clúster de Kubernetes, ja que és un subdomini intern el qual utilitza el clúster per definir els objectes del clúster perquè els recursos desplegats es puguin comunicar amb tota la infraestructura sense la necessitat d'exposar aquests recursos públicament. En aquest cas fa referència al un recurs amb el nom kubernetes, desplegat al *namespace* default i que és de tipus servei (o *service*). Aquest servei exposa l'API de Kubernetes internament per poder gestionar tot el clúster.

En aquest cas s'indica la URL interna ja que tenim desplegat ArgoCD en el mateix clúster que volem gestionar, però ArgoCD permet indicar una URL d'un altre clúster, tant intern en el mateix proveïdor Cloud, com extern en un altre proveïdor, datacenter o CPD. Això dona a l'usuari la possibilitat de desplegar infraestructura en diferents clústers per tot el mon, remotament i automatitzable.

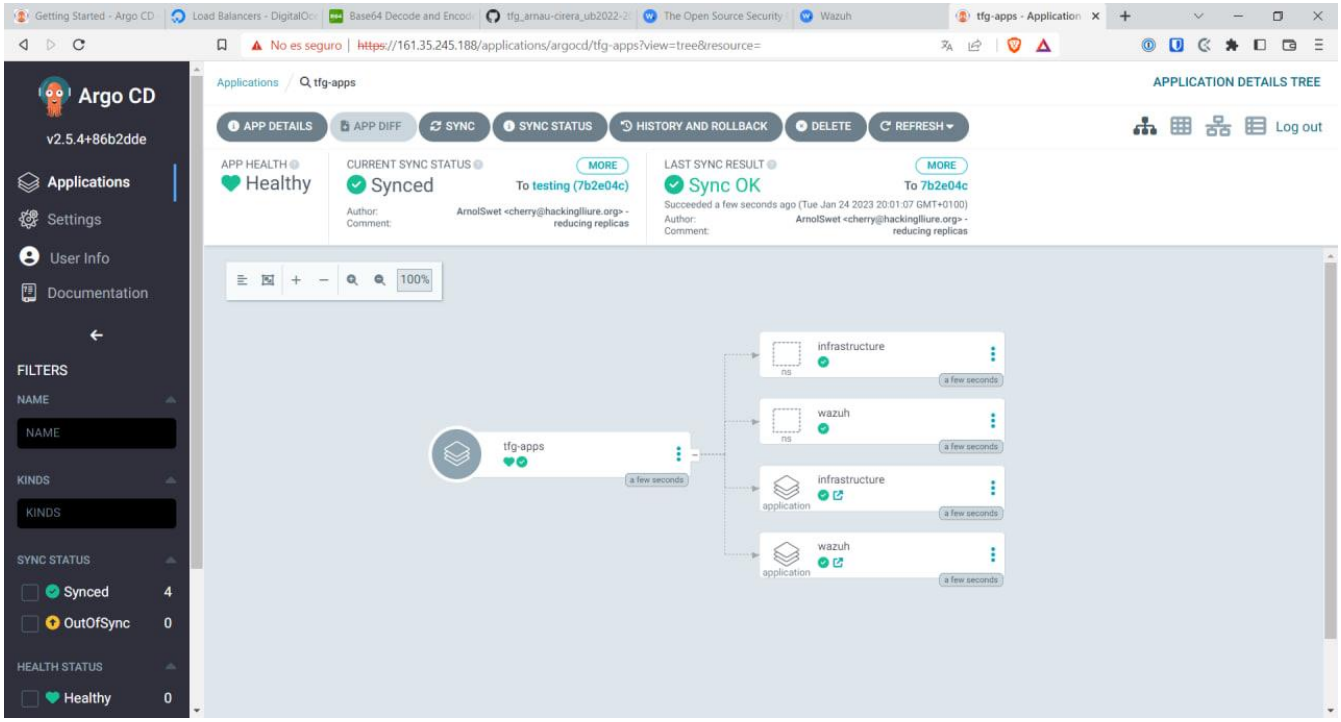
Desplegarem els recursos en el *namespace* argocd ja que amb aquesta *Application* desplegarem altres *Applications* i ArgoCD no permet (per defecte)[4] el desplegament d'*Applications* fora del *namespace* argocd, ja que els *pods* que gestionen el sistema d'Argocd estan en aquest *namespace*, per tant no poden accedir als recursos d'altres *namespaces*.

Un cop creada l'aplicació podrem veure l'*Application* tfg-apps sincronitzada, ja que hem configurat la sincronització automàtica, que ha creat dos *Applications* més, definides dins el repositori amb .yaml.

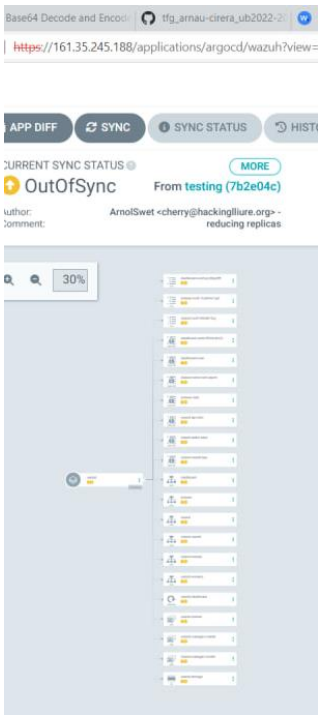


Com podem veure, l'*Application* tfg-apps, ja sincronitzada, es mostra amb color verd i indicant que els recursos estan desplegats correctament (*Healthy)*. I les *Applications* no sincronitzades, en color taronja i indicant que falten recursos a definir.

[4] RBAC (Role-Based Access Control) és un mètode per regular l'accés als recursos informàtics o de xarxa en funció dels rols dels usuaris (usuaris o pods) individuals dins de la vostra organització. Per defecte, ArgoCD dona permisos als recursos d'ArgoCD només al seu propi *namespace*. Es pot modificar el permís associat a l'usuari que utilitzen els pods d'ArgoCD per tal de tenir accès a tot el clúster. Permetent així, poder desplegar *Applications* a qualsevol *namespace*.



Aquí podem veure els recursos desplegats per aquesta *Application* i gestionar tota l'*Application*. Podem veure que s'ha desplegat dos *namespace* (un per cada *Application* que controlarà les solucions) i les dos *Applications*.



Una de les *Application* és la que gestiona els recursos que despleguen la solució de Wazuh. Com es pot veure, crea una gran quantitat de recursos, els quals no entrarem en detall, que s'han extret del repositori que ofereix Wazuh per el desplegament a Kubernetes[5].

Es desplegaran diferents recursos que formaran les diferents aplicacions:

- Wazuh Server: Distribuït en un clúster d'un pod master i 2 pods workers.
- Wazuh Indexer: En aquest cas amb només un pod, però es pot utilizar amb alta disponibilitat amb repliques del pod.
- Wazuh Dashboard

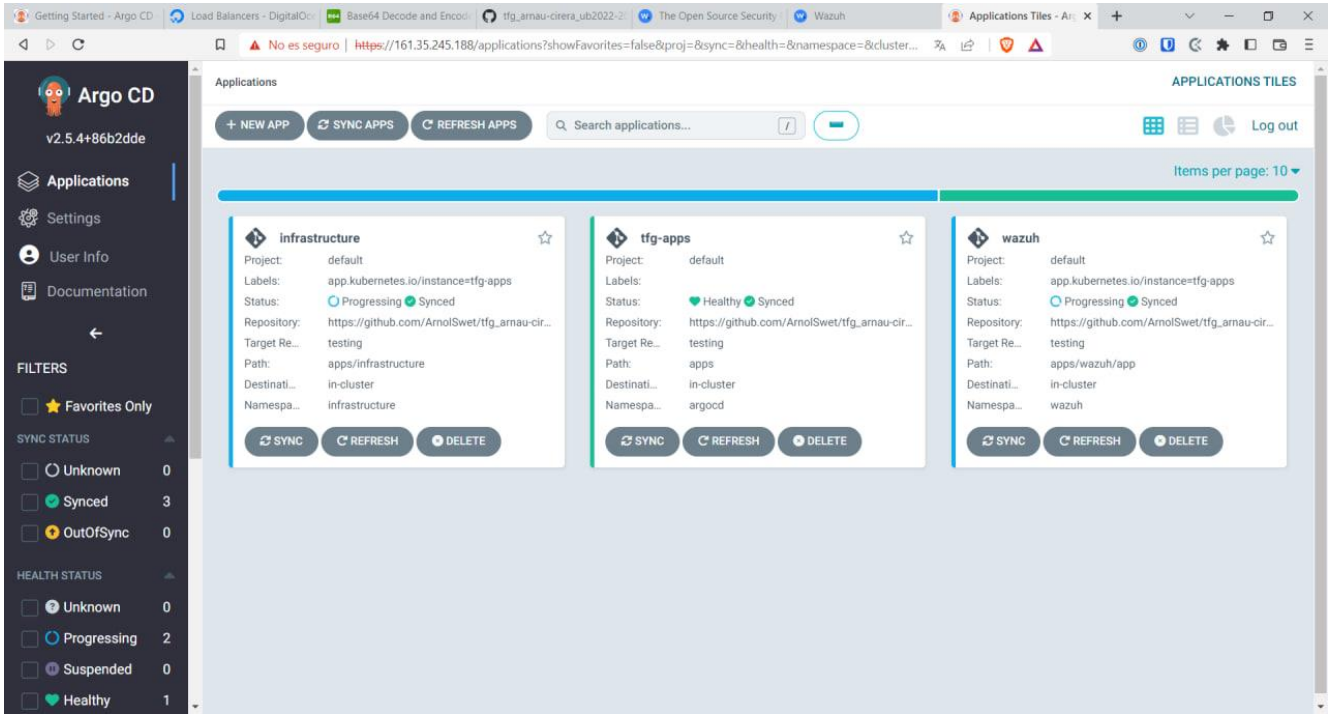
També es crearan els serveis que permetin la connexió d'entrada als pods, dels quals tres d'ells seran balancejadors de càrrega:

- Dashboard: Per tal d'exposar el dashboard de Wazuh públicament perquè els usuaris hi puguin accedir des del navegador.
- Indexer: Per poder connectar l'Indexer amb altres eines externes, ja que al ser un fork d'Elasticsearch, es pot connectar amb qualsevol sistema compatible amb aquest o enviant a través de peticions amb JSON les queries per extreure les dades.
- Wazuh workers: Connexió externa als workers de Wazuh Server, perquè els agents de Wazuh instal·lats a la infraestructura a monitoritzar, puguin enviar els esdeveniments i logs al servidor de Wazuh.

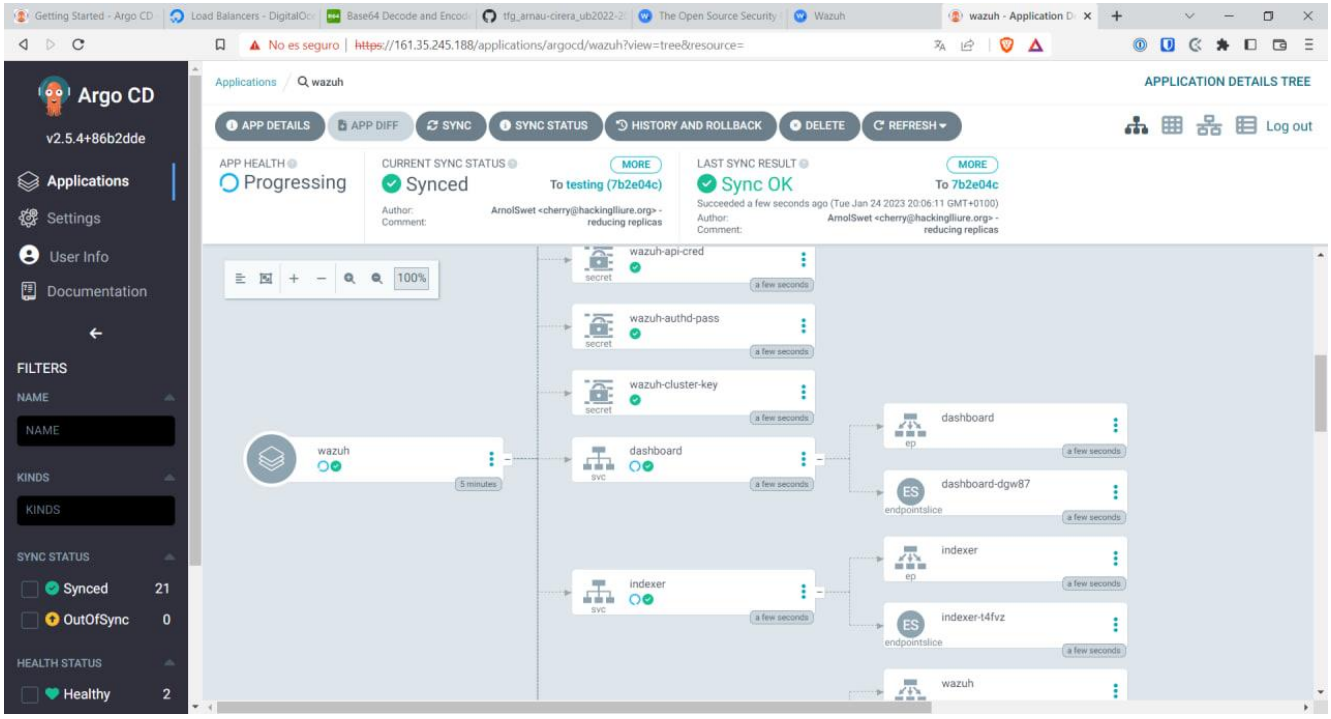
[5] [https://github.com/wazuh/wazuh-kubernetes.git]

<< En el nostre cas, podriem reduir els serveis de l'Indexer i els workers de Wazuh a serveis de tipus *ClusterIP*, ja que la infraestructura a monitorar està al mateix clúster, pel que no cal exposar els workers a internet, i l'Indexer només serà consultat pel propi Dashboard que també es troba al mateix clúster.>>

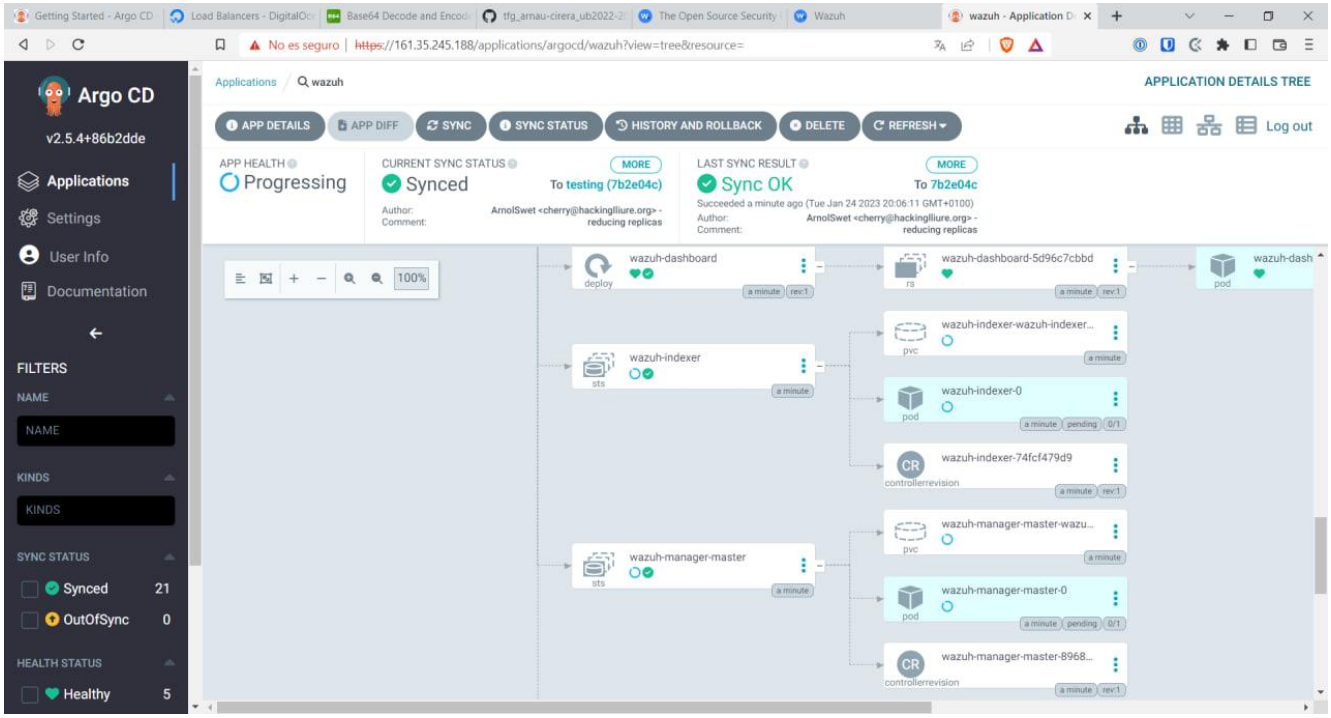
Per desplegar els recursos de les altres *Applications* haurem de sincronitzar-les. Ho podem fer des del menú general d'aplicacions, on podrem seleccionar quines *Applications* volem sincronitzar i com.



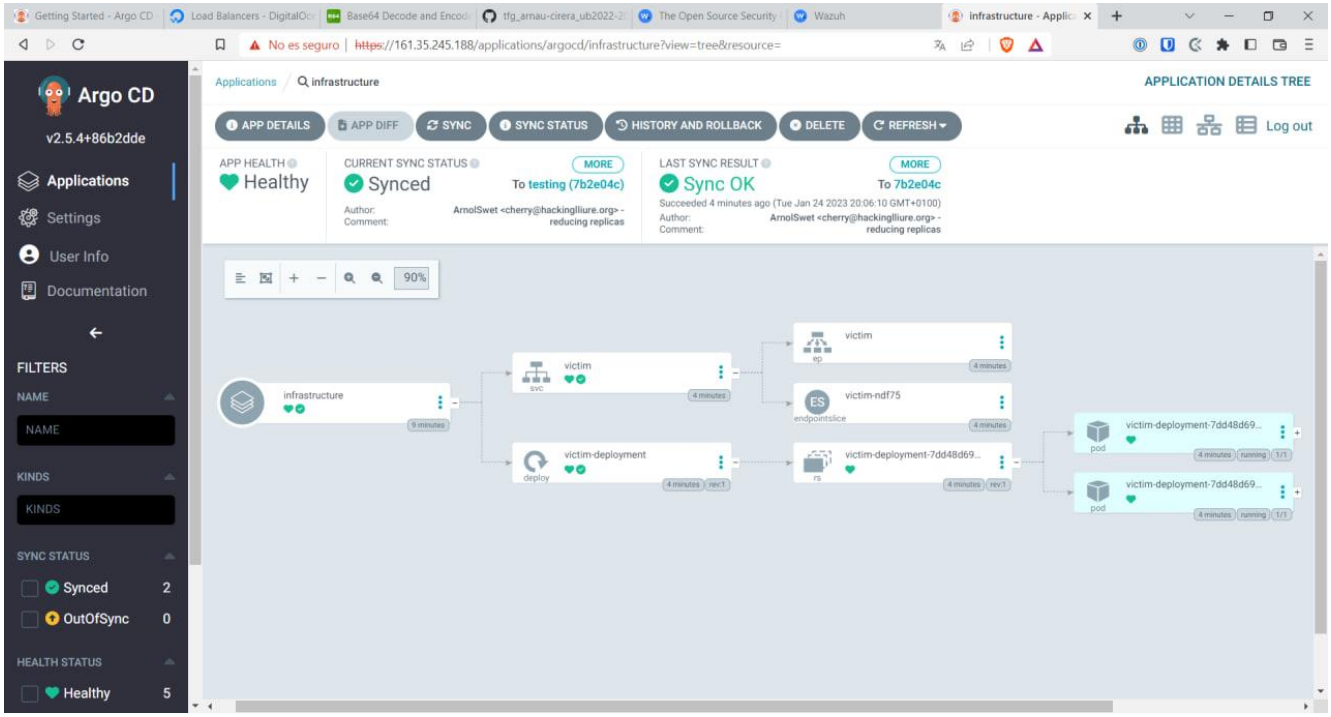
Un cop sincronitzades, veurem com passen a un estat en progrés en el que s'està desplegant els diferents recursos.



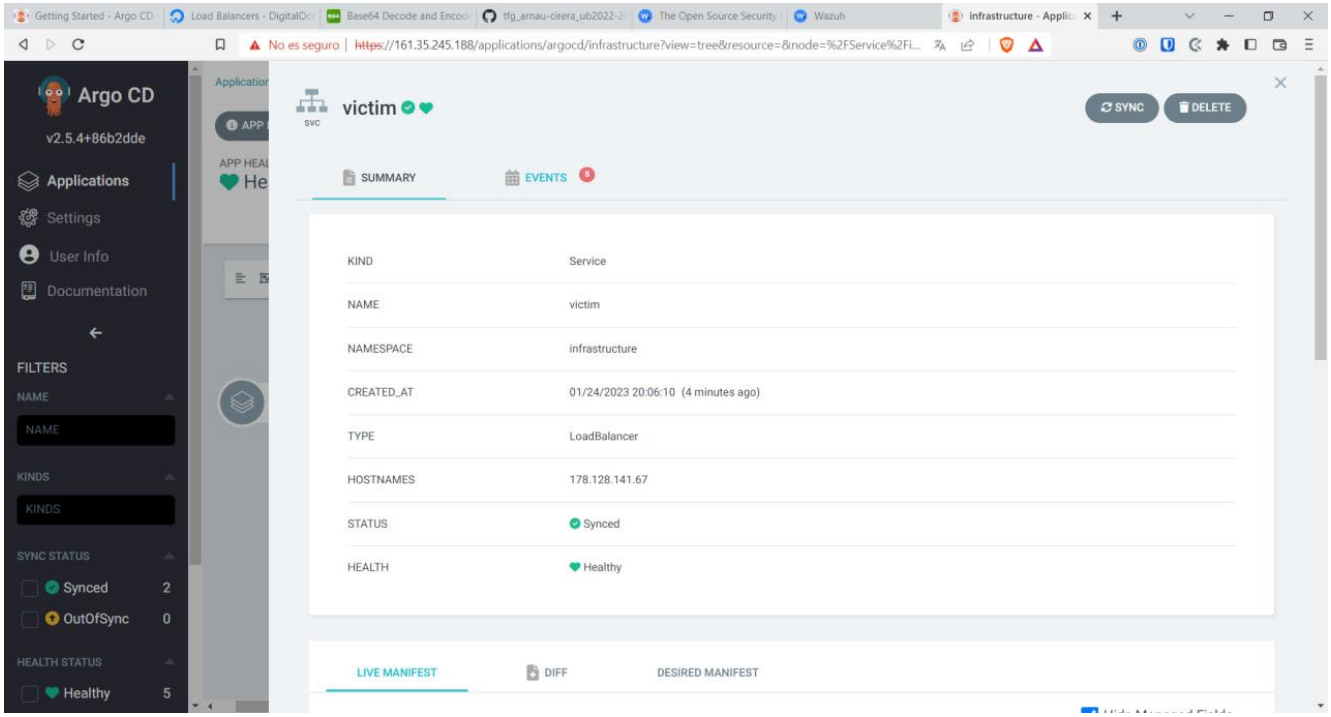
Dins de cada aplicació, ens mostrarà quins recursos ja estan desplegats, i quins recursos estàn en procés.



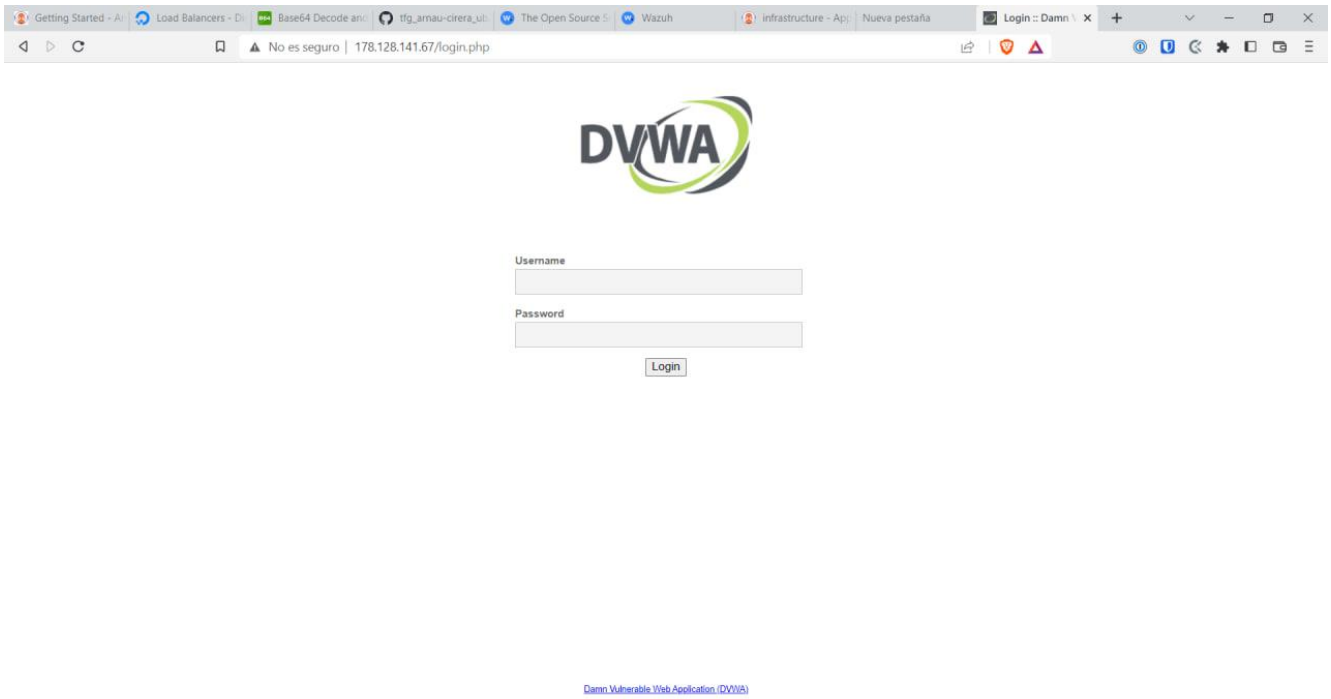
Com es pot veure, ArgoCD mostra els diferents elements que formen una declaració com per exemple els *Statefulsets* com l'Indexer o el Wazuh Server Master, que mostra com es crea el *pod* que executarà el contenidor, el *PVC* que emmagatzemarà les dades del *pod* i un controlador per analitzar la connexió entre el *PVC* i el *pod*.



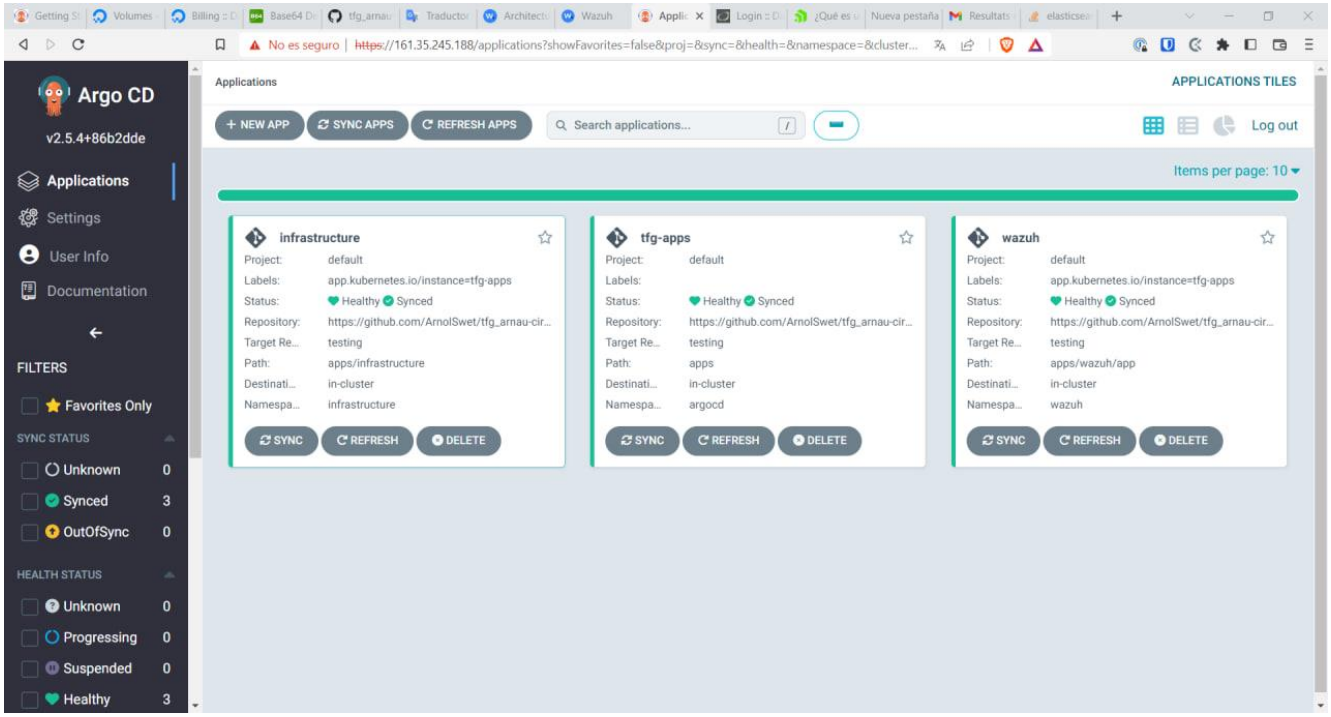
L'altra *Application* engloba els recursos de les dos víctimes on es declara un servei tipus *LoadBalancer*, per poder accedir a la web, i un *Deployment* que desplega un *ReplicaSet* de dos repliques.



Si cliquem a un recurs podem veure el detall d'aquest, com per exemple en el cas d'aquest servei, poder veure la direcció IP la qual ens permetrà des d'un navegador, accedir a la web vulnerable.



Així es mostra el *web server* vulnerable el qual monitoritzarem per controlar la seva ciberseguretat.

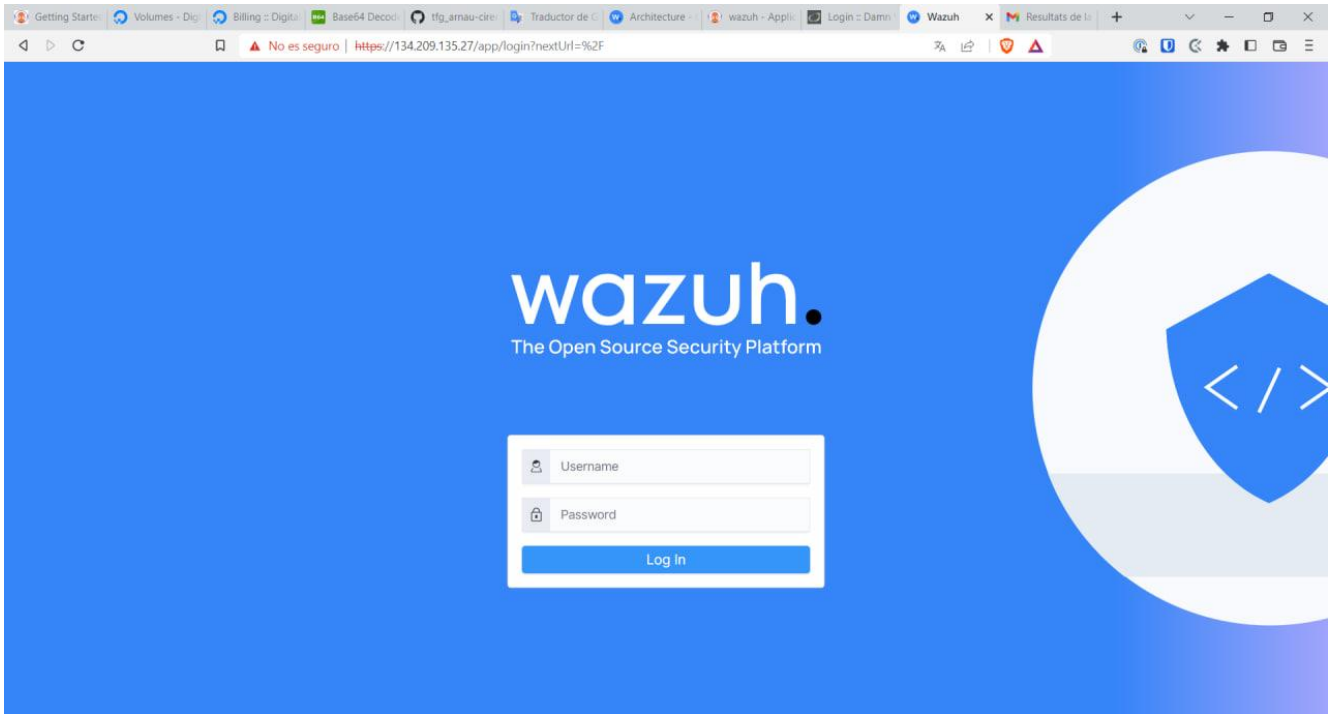


Un cop tinguem totes les *Applications* sincronitzades i ben desplegades sense cap error, ens apareixerà de color verd, finalitzant així, el desplegament de la infraestructura Cloud.

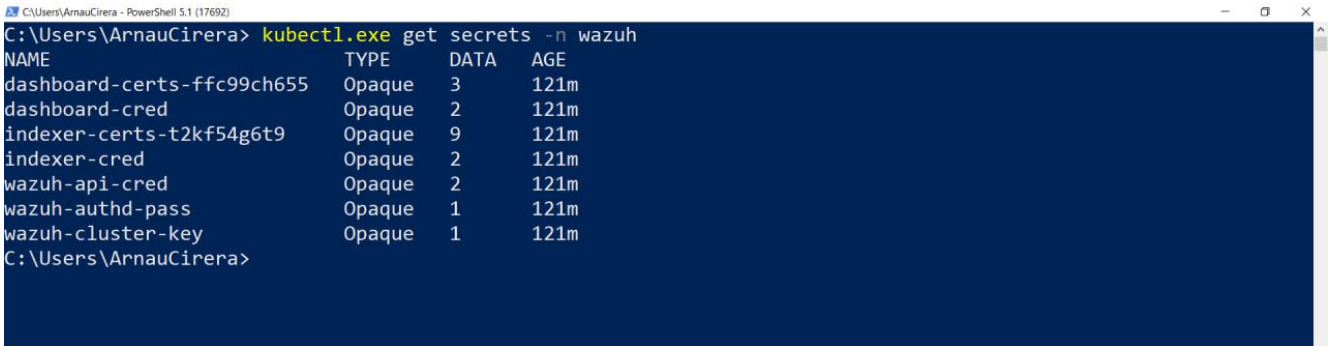
7.3. Configuració de Wazuh un cop desplegat adaptar-lo al projecte

Per integrar els agents de Wazuh al Servidor, primer l'hem de configurar per rebre la connexió dels agents.

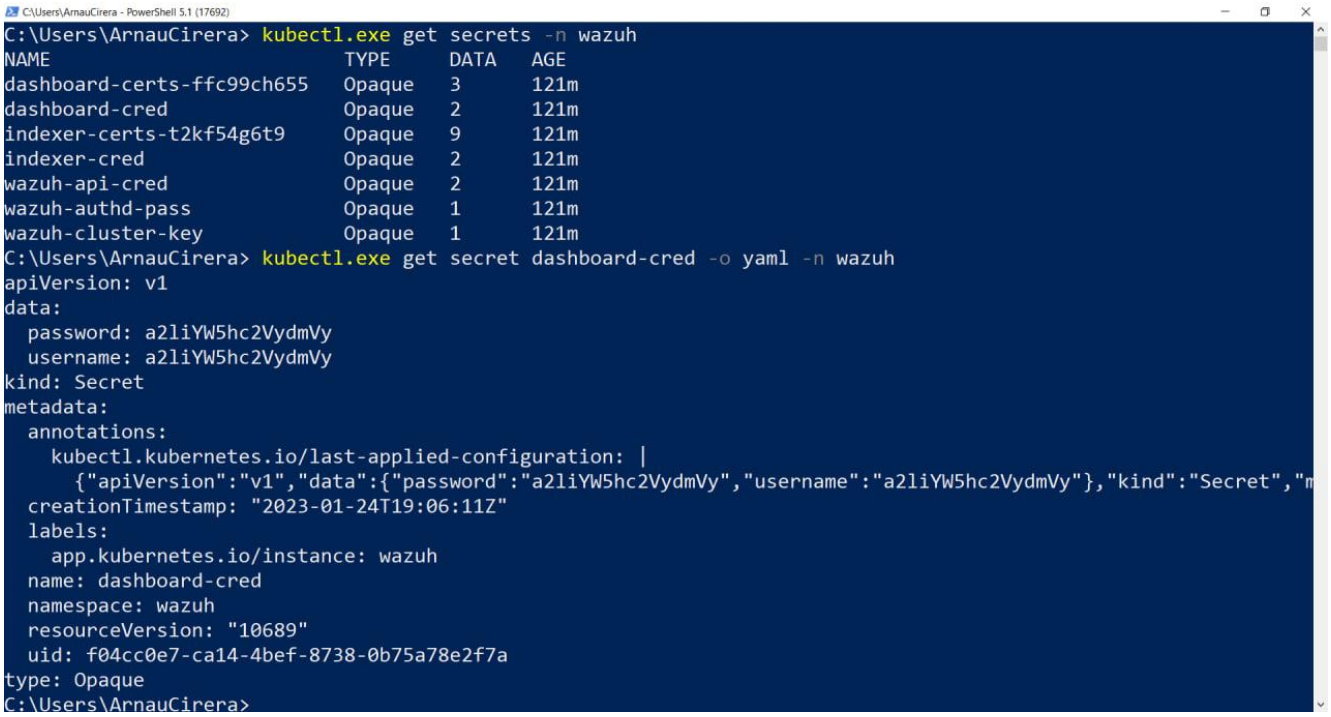
Per això, mirarem amb ArgoCD la IP externa del balancejador de càrrega que exposa el Dashboard a Internet. En aquest cas 134.209.135.27, on trobarem la pàgina d'inici de sessió de Wazuh.



Per saber l'usuari i contrasenya per accedir-hi, hem de buscar el *secret* que Wazuh genera amb les credencials de forma aleatòria. Per això tornarem a la terminal i utilitzarem la comanda `kubectl get secrets -n wazuh` per mostrar els *secrets* que es troben al *namespace* de `wazuh`.



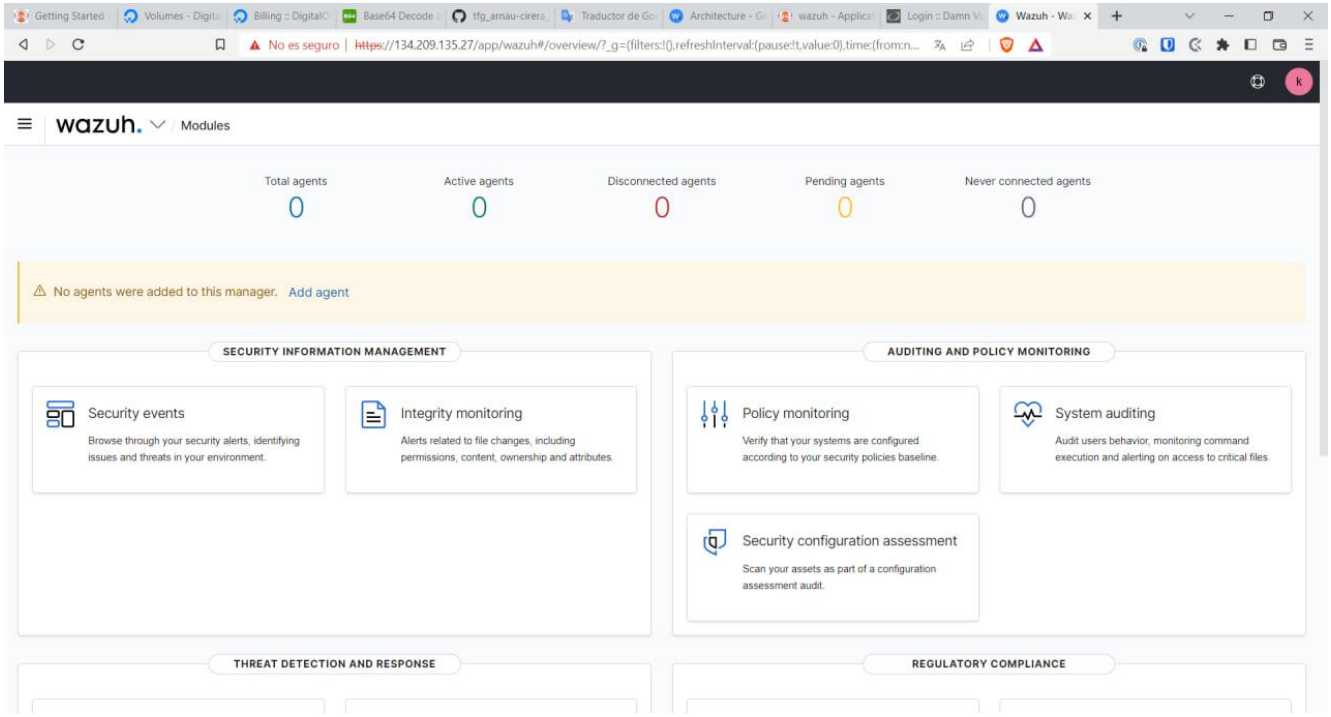
El *secret* que ens interessa és dashboard-cred. Per visualitzar el contingut utilitzem la comanda kubectl get secret dashboard-cred -o yaml -n wazuh per mostrar el *secret* en format .yaml i visualitzar el contingut del manifest.



Com es pot veure, en el camp data: trobem els camps password: i username: que ens mostren l'usuari i la contrasenya amb base64.

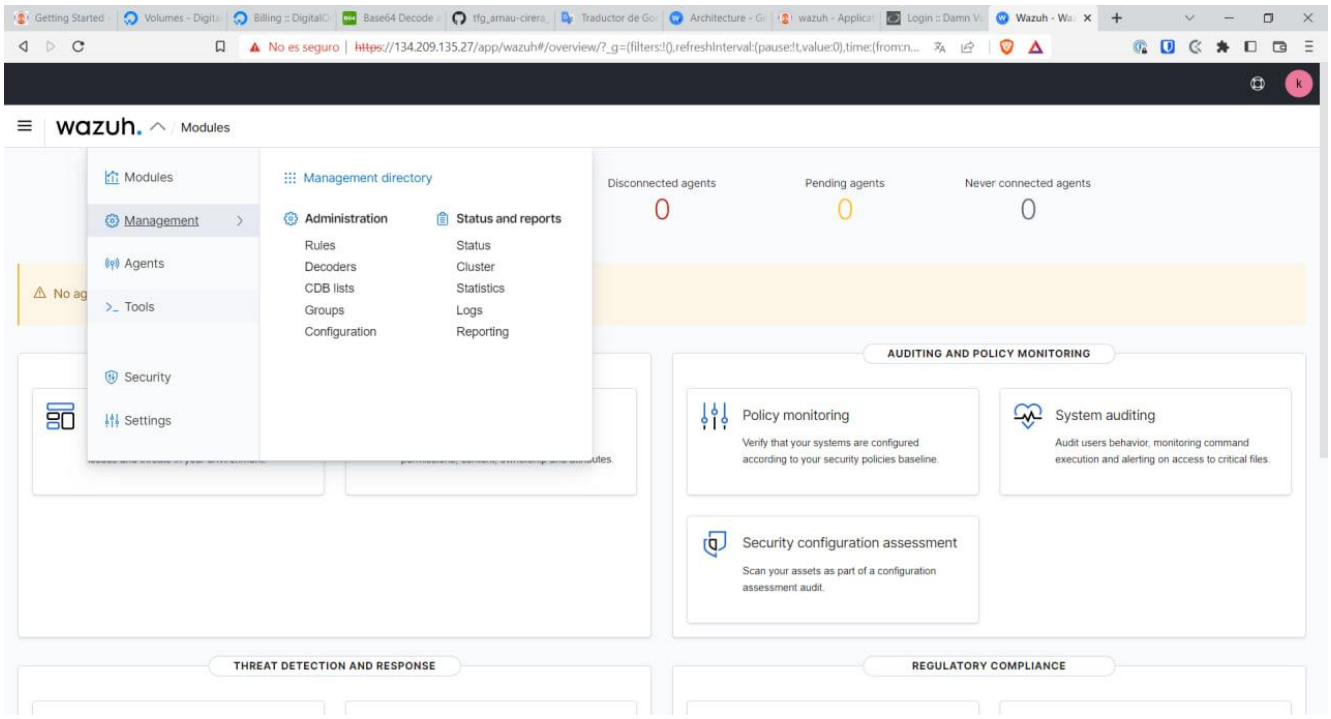
Per visualitzar les credencials, descodificarem els *strings* trobats i veurem que tant l'usuari com la contrasenya és *kibanaserver*.

Un cop iniciat sessió ens trobarem la pantalla inicial la qual ens mostrarà els agents en els seus diferents estats, i els diferents mòduls que conté Wazuh.



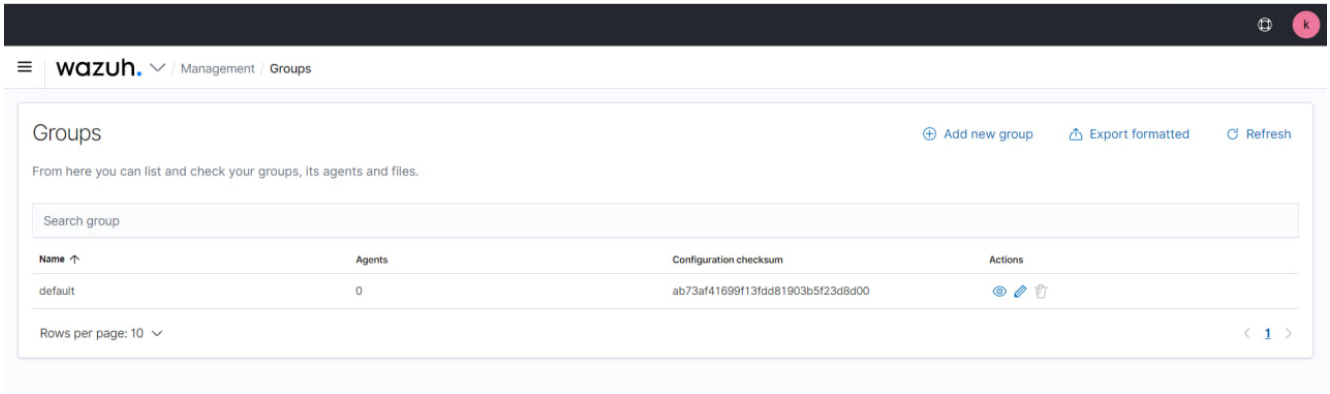
Com es mostra, no hi ha cap agent connectat.

Abans de afegir un agent, ens dirigirem al menú de mòduls on seleccionarem l'apartat de *management* on trobarem el subapartat *Groups* dins de l'Administració.

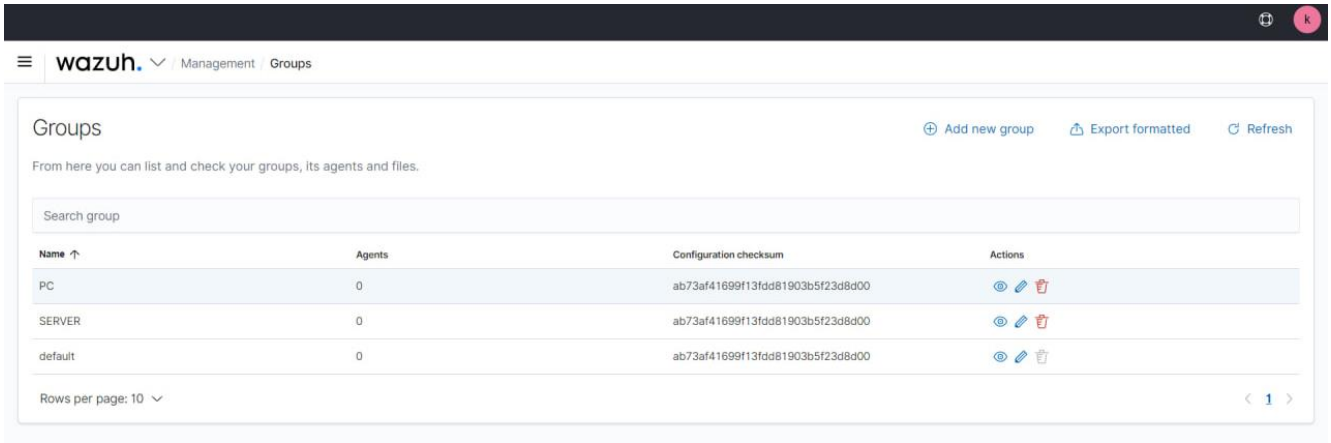


Els grups d'agents permeten separar un conjunt d'agents els quals es pot modificar la configuració simultàniament.

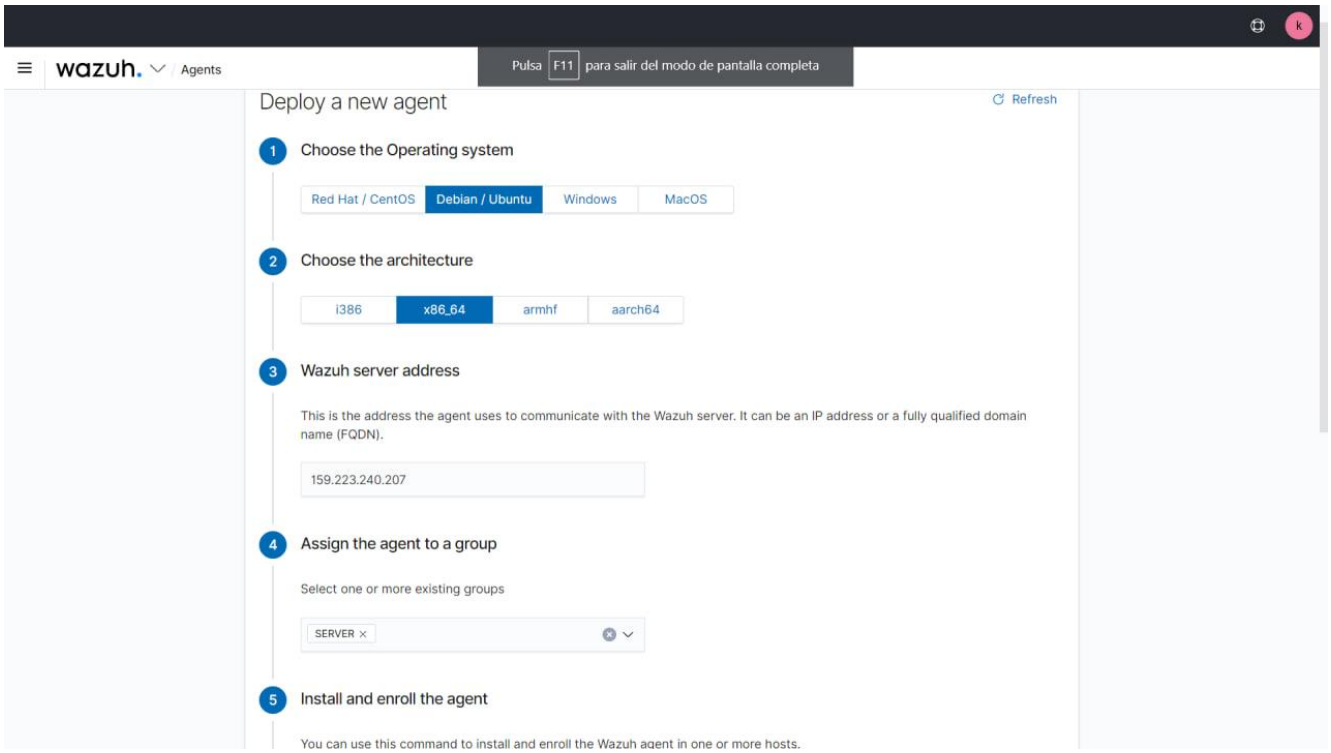
Un cas pràctic per el control de dispositius finals d'una xarxa amb Wazuh, és agrupar els agents amb ordinadors personals (PC) i servidors (SERVER). Ja que en un cas pràctic de control d'una organització d'X treballadors, és l'agrupació que necessitarà unes especificacions més diferencials.



Per afegir un nou grup, només hem d'anar a *Add new group* i indicar un nom.



Un cop creat els grups, ja podem afegir un nou usuari. Tornarem al menú d'agents, i seleccionarem *Add new agent*.

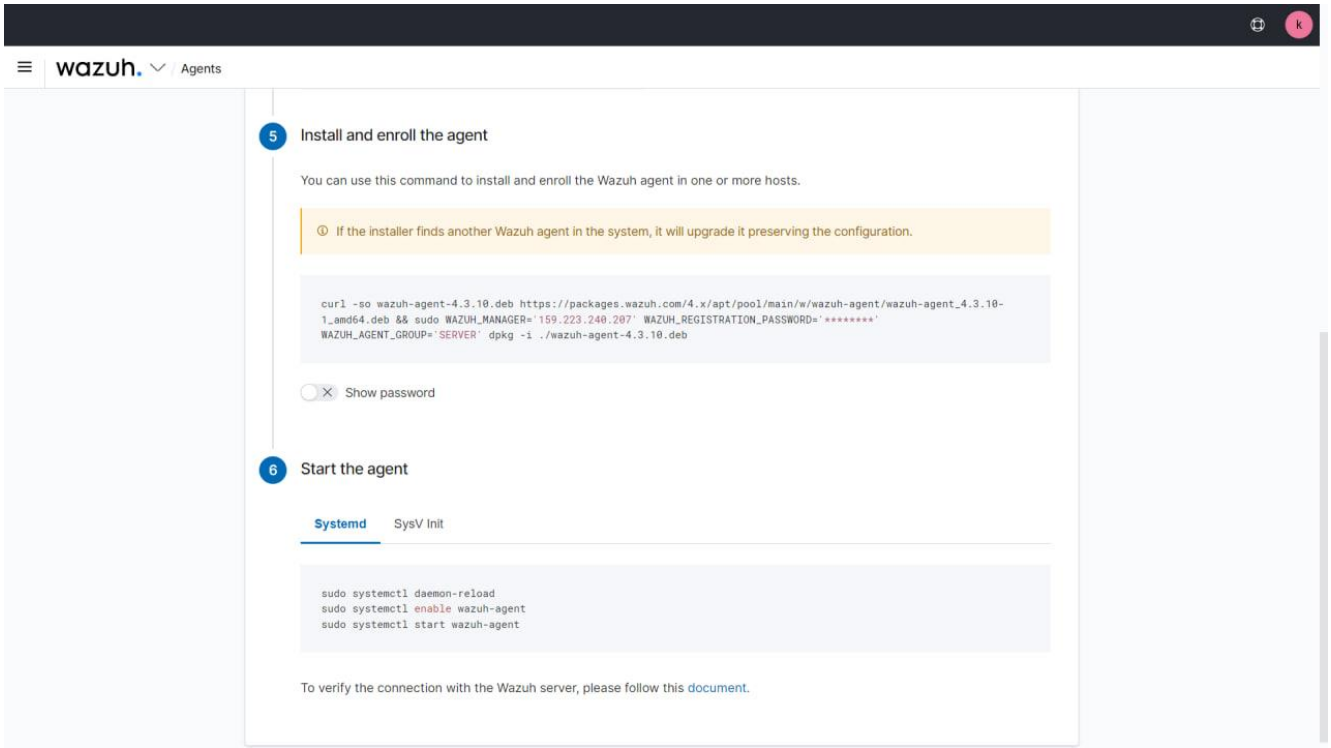


Per afegir un agent, s'ha d'introduïr una comanda a la terminal del dispositiu final per descarregar l'executable i la configuració que requereix per connectar-se al Wazuh Server.

Wazuh ofereix des del Dashboard, una guia per generar aquesta comanda.

Primer triarem el Sistema Operatiu i l'arquitectura del dispositiu que volem instal·lar l'agent. A continuació introduirem l'adreça del servidor. En aquest cas, es podria indicar el subdomini associat al servei que apunta als pods del Wazuh Server, ja que els agents es desplegaran al mateix clúster de Kubernetes. Però com hem utilitzat un balancejador de càrrega per simular, com seria si la infraestructura a monitorar fos externa al clúster de Kubernetes, introduïrem l'adreça IP d'aquest *service*. En aquest cas, 159.223.240.207.

Finalment sel·leccionarem el grup al qual volem afegir l'agent, en aquest cas, a SERVER o PC.



Un cop introduïdes aquestes dades, ens generarà la comanda preparada per instal·lar i connectar l'agent de Wazuh. També ens indicarà com configurar l'agent com un servei, per tornar-se a executar si el dispositiu s'apaga.

Amb això finalitza la configuració del servidor de Wazuh.

8. Instal·lació i configuració dels agents de Wazuh

Per afegir l'agent, ens connectarem al dispositiu tant sigui remotament (RDP o SSH) o físicament i obrirem la terminal per introduir la comanda.

En aquest cas, al tenir el dispositiu en un contenidor, haurem de connectar-nos utilitzat la comanda kubectl exec <<nom del pod>> -it sh -n <<namespace>> que executarà la comanda sh (*Shell*) al contenidor deixant la consola oberta i mostrant-la per pantalla per poder interactuar amb el servidor.

```
C:\Users\ArnauCirera - PowerShell 5.1 (17692)
C:\Users\ArnauCirera> kubectl exec victim-deployment-7dd48d6997-4df95 -it sh -n infrastructure
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
#
```

En aquest contenidor, és possible que no hi hagi paquets necessaris o que estiguin desactualitzats, ja que, per optimitzar recursos, s'ha creat només amb els paquets justos per executar un *Web Server.* Al obrir la terminal, primer de tot, actualitzarem els repositoris de la distribució de Linux, i actualitzarem els paquets desactualitzats.

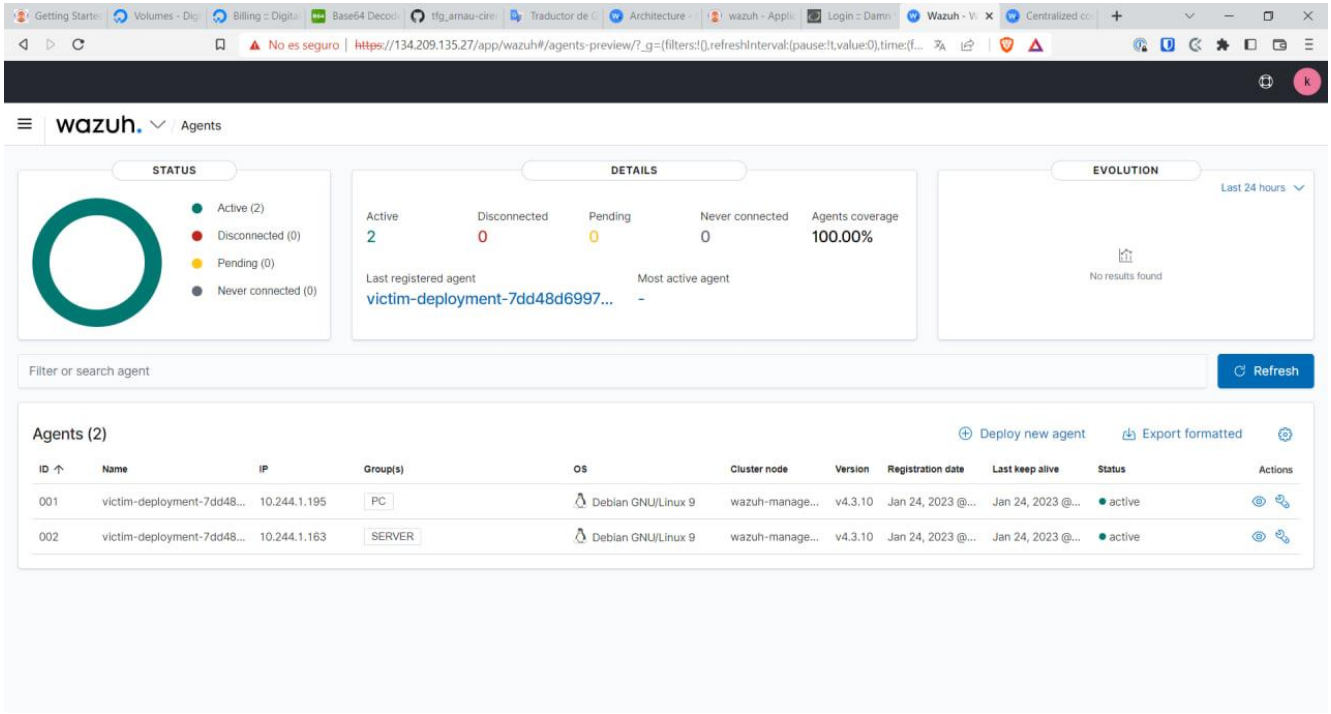
Després introduirem la comanda generada per Wazuh.

```
C:\Users\ArnauCirera - PowerShell 5.1 (17692)
# ^[[A^[[A^[[A^[[A^C
# service wazuh-agent start
Starting Wazuh v4.3.10...
Started wazuh-execd...
Started wazuh-agentd...
Started wazuh-syscheckd...
Started wazuh-logcollector...
Started wazuh-modulesd...
Completed.
#
```

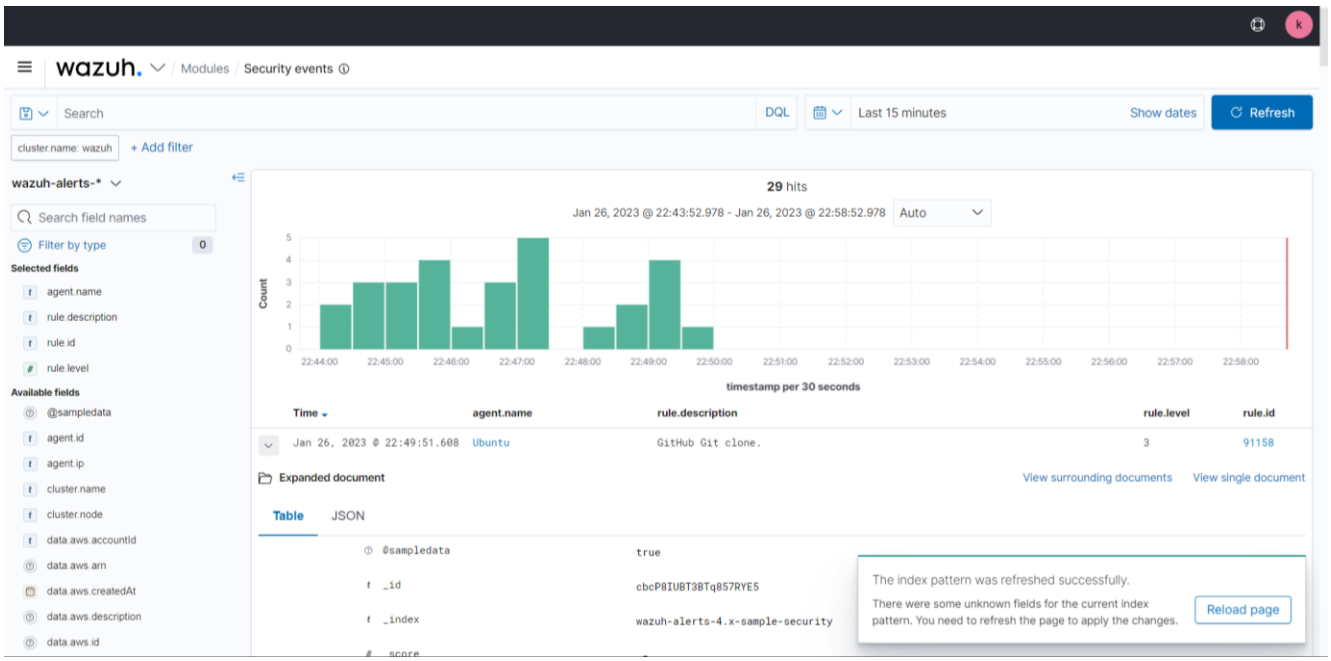
Finalment introduïm la comanda que ens indica Wazuh per crear i executar el servei, i ja tindrem instal·lat Wazuh en el dispositiu final.

Fem el mateix procediment amb l'altre *pod*, un registrant l'agent al grup SERVER i l'altre, al grup PC.

Un cop es connectin, es podran visualitzar a la llista d'agents connectats i començar a visualitzar els esdeveniments.



A continuació es mostra com es visualitzen els esdeveniments que arriben als servidors i el detall d'aquests:



IV. EVALUACIÓ DE RESULTATS

S'ha desenvolupat una solució desplegable automàticament i per tant, escalable, que permet detectar amenaces de ciberseguretat a dispositius finals d'una xarxa i amb un sistema àgil amb control de versions de la infraestructura a desplegar, cumplint així, amb els objectius plantejats in inici del projecte.

V. CONCLUSIONS

Amb la investigació d'aquest projecte he après les diverses tecnologies que es poden trobar al mercat per afrontar diferents necessitats. He pogut veure com la tecnologia d'avui en dia, és capaç de desplegar infraestructura massiva de forma automàtica i amb una gran velocitat, que fa possible la creació de nous serveis i aplicacions de grans dimensions les quals resideixen al Cloud, per poder donar una disponibilitat i facilitat d'ús elevada, utilitzant aplicacions com a clients que requereixen de menys recursos, fent els desplegaments de tecnologies més eficaços.

El valor més important que trec a aquest projecte i a aquesta tecnologia, són les possibilitats infinites de combinació d'aplicacions per donar més valor a la solució i incloure més funcionalitats.

Aquest projecte el segueixo treballant avui en dia amb tot el meu equip en el que integrem noves tecnologies de ciberseguretat per donar més força al producte. Volent arribar al següent nivell de tecnologies, el desenvolupament de mòduls d'aprenentatge automàtic (ML) per gestionar aquestes tecnologies.

REFERÈNCIES

- **Kubernetes. (s.d.). Kubernetes**. Recuperat de [https://kubernetes.io/]
- **Kubernetes. (s.d.). Using RBAC Authorization**. Recuperat de [https://kubernetes.io/docs/reference/access-authn-authz/rbac/]
- **Wazuh. (s.d.). Wazuh - The Open Source Security Platform.** Recuperat de [https://wazuh.com/]
- **Wazuh. (s.d.). Wazuh Documentation.** Recuperat de [https://documentation.wazuh.com/current/index.html]
- **Argo CD. (s.d.). Argo CD - Declarative GitOps CD for Kubernetes.** Recuperat de [https://argo-cd.readthedocs.io/en/stable/]
- **Argo CD. (s.d.). Argo CD.** Recuperat de [https://argoproj.github.io/cd/]
- **Limaye, H. R. (s.d.). Deploying Wazuh on Kubernetes.** Recuperat de [https://medium.com/@hrlimaye/deploying-wazuh-on-kubernetes-b3a231608f35]
- **DigitalOcean. (s.d.). DigitalOcean - Cloud Computing, Simplicity at Scale.** Recuperat de [https://www.digitalocean.com/]
- **Refactorizando. (s.d.). Desplegar aplicaciones con ArgoCD en Kubernetes**. Recuperat de [https://refactorizando.com/desplegar-aplicaciones-argocd-kubernetes/]
- **Elastic. (s.d.). ¿Qué es Elasticsearch?.** Recuperat de [https://www.elastic.co/es/what-is/elasticsearch]

ANNEX 1: KUBERNETES

Què fa de Kubernetes una plataforma?

Tot i que Kubernetes ja ofereix moltes funcionalitats, sempre hi ha nous escenaris que es beneficien de noves característiques. Els fluxes de treball de les aplicacions poden optimitzar-se per accelerar el temps de desenvolupament. Una solució d'orquestració pròpia pot ser suficient al principi, però sol requerir d'una automatització robusta cuan és necessari escalar. Es per això que Kubernetes va ser dissenyada com una plataforma: per poder construir un ecosistema de components i eines que fan més fàcil el desplegar, escalar i administrar aplicacions.

Les etiquetes, o *Labels*, permeten als usuaris organitzar els seus recursos com desitgin, Les anotacions, o Annotations, permeten assignar informació arbitrària a un recurs per facilitar els seus fluxes de treball i fer més fàcil a les eines administratives inspeccionar l'estat.

A més, el Pla de Control de Kubernetes utilitza les mateixes APIs que utilitzen els desenvolupadors i usuaris finals. Els usuaris finals poden escriure els seus propis controladors, com per exemple un planificador o scheduler, utilitzant les seves pròpies APIs des d'una eina de línia de comandes.

Kubernetes està format per 3 principals components: El Pla de control, els components del node i Addons que augmenten les funcionalitats de Kubernetes.

Components del Pla de Control

Els components que formen el pla de control prenen decisions globals sobre el clúster (per exemple la planificació) i detecten i responen a esdeveniments del clúster, com la creació d'un nou *pod* cuan la propietat *repliques* d'un controlador de replicació no es compleix.

Aquests components poden executar-se en qualsevol node del cúster. Tot i així, per simplificar, els scripts d'instal·lació normalment s'inicien en el mateix node de forma exclusiva, sense que s'executin contenidors dels usuaris en aquestes nodes. El pla de control s'executa en diferents nodes per garantitzar l'alta disponibilitat.

- Kube-apiserver: És el component que exposa l'API de Kubernetes. Es tracta del frontend de Kubernetes, rep les peticions i actualitza d'acord a l'estat en etcd.
- Etcd: Emmagatzematge de dades persistent, consistent i distribuït de clau-valor utilitzat per emmagatzemar tota la informació del clúster de Kubernetes.
- Kube-scheduler: Component que està pendent dels *pods* que no tenen cap node assignat i selecciona un on executar-ho. Per això es té en compte diversos factors: requisits de recursos, restriccions de hardware/software/polítiques, afinitat i anti-afinitat, localització de dades dependents, entre altres.
- Kube-controller-manager: Executa els controladors de Kubernetes. Cada controlador és un procés independent, però per reduir la complexitat, tots es compilan en un únic binari i s'executa en un mateix procés. Aquests controladors inclueixen:
    - Controlador de nodes: Responsable de detectar i respondre quan un node deixa de funcionar
    - Controlador de replicació: Responsable de mantenir el número correcte de pods per cada controlador de replicació del sistema.
    - Controlador d'endpoints: contrueixen l'objecte *Endpoints*, es a dir, realitza una unió entre els *serveis* i els *pods.*

    - Controladors de tokens i contes de servei: creen contes i tokens de accés a la API per defecte pels nous *Namespaces.*
- Cloud-controller-manager: Executa controladors que interactuen amb proveïdors del núvol. Només executa cicles de control específics per cada proveïdor Cloud. Permet que el codi de Kubernetes i el del proveïdor, evolucionin independentment. Els següents controladors depenen d'alguna forma d'un proveïdor Cloud:
    - Controlador de nodes
    - Controlador de rutes: Per configurar rutes en la infraestructura de núvol subjacent.
    - Controlador de serveis: Per crear, actualitzar i eliminar balancejadors de càrrega en el núvol.
    - Controlador de volums: Per crear, connectar i muntar volums i interactuar amb el proveïdor Cloud per orquestrar-los.

Components de node

Els components de node corren en cada node, mantenint als pods en funcionament i proporcionant l'entorn d'execució de Kubernetes.

- Kubelet: Agent que s'executa en cada node d'un clúster. S'assegura de que els contenidors estiguin corrent en un *pod*. L'agent pren un conjunt d'especificacions de *Pod*, anomenats *PodSpecs,* que han sigut creat per Kubernetes i garantitza que els contenidors descrits en ells, estiguin funcionant i ne bon estat.
- Kube-proxy: Permet abstreure un servei en Kubernetes mantenint les regles de xarxa en l'anfitrió i fent reenviament de connexions.
- Runtime de contenidors: Software responsable d'executar els contenidors. Kubernetes suporta diferents d'ells: Docker, containerd, cri-o, rktlet i qualsevol implementació de la interfície de runtime de contenidors de Kubernetes, o Kubernetes CRI.

Addons

Els *addons* son *pods* i *serveis* que implementen funcionalitats del clúster. Aquests poden ser administrats per *Deployments, ReplicationControllers* i altres. Els *addons* assignats a un espai de noms es creen en l'espai kube-system.

A continuació es mostren alguns dels principals *addons*.

- DNS: Si bé tots els addons no són estrictament necessaris, tots els cústers de Kubernetes han de tenir un DNS intern del clúster ja que la majoria dels exemples ho requereixen. El DNS intern, és un servidor DNS, addicional als que ja podries tenir en la teva xarxa, que serveix registres DNS als serveis de Kubernetes. Els contenidors que son iniciats per Kubernetes inclueixen automàticament aquest servidor en les seves cerques DNS.
- Interfície web (Dashboard): Permet als usuaris administrar i resoldre problemes que pugin aplicar tant les aplicacions com el clúster.
- Monitor de recursos de contenidors: Emmagatzema de forma centralitzada, series de temps amb mètriques sobre els contenidors, i proveeix una interfície per navegar aquestes dades.
- Registres del clúster: El mecanisme de registres del clúster està a càrrec d'emmagatzemar els registres dels contenidors de forma centralitzada, proporcionant una interfície de cerca i navegació.

Per utilitzar Kubernetes i desplegar la infraestructura, aplicacions i polítiques desitjades, s'utilitzen els recursos de Kubernetes anomenats *Objectes*.

Objectes

Els objectes de Kubernetes son entitats persistents dins del sistema de Kubernetes. Kubernetes utilitza aquestes entitats per representar l'estat del teu clúster. Específicament, poden descriure:

- Quines aplicacions corren en contenidors (i en quins nodes)
- Els recursos disponibles per aquestes aplicacions
- Les polítiques apropen de com aquestes aplicacions es comportan, con les polítiques de reinici, actualització i tolerància a errors

Un objecte de Kubernetes és un "registre d'intenció", un cop creat l'objecte, el sistema de Kubernetes s'encarregarà d'assegurar que l'objecte existeix.

Per treballar amb aquests objectes, s'utilitza l'API de Kubernetes.

Abast i estat d'un objecte

Cada objecte de Kubernetes inclueix dos camps com a objectes que determinen la configuració de l'objecte: el camp *spec* i el camp *status.* El camp *spec*, que és obligatori, descriu l'estat desitjat de l'objecte (les característiques que vols que tingui l'objecte). El camp *status* descriu l'estat actual de l'objecte, i es subministra i actualitza directament pel sistema de Kubernetes. En qualsevol moment, el Pla de Control gestiona de forma activa l'estat actual de l'objecte perquè coincideixi amb l'estat desitjat requerit.

Per exemple, un *Deployment* de Kubernetes és un objecte que pot representar una aplicació del teu clúster. Quan es creii el Deployment, pots especificar en el *spec* del Deployment que vols còrre tres rèpliques de l'aplicació. El sistema de Kubernetes llegueix l'*spec* del Deployment i comença a instanciar rèpliques de la teva aplicació, actualitzant l'estat per conciliar-ho amb el teu *spec*. Si qualsevol de les instàncies falla (un canvi d'estat), el sistema de Kubernetes soluciona la diferencia entre l'*spec* i l'estat realitzant una correcció, en aquest cas, iniciant una altra instàcia de reemplaçament.

Descripció d'un objecte de Kubernetes

Per crear un objecte, es realitza una petició a la API de Kubernetes on s'especifica l'*spec* de l'objecte que descriu el seu estat desitjat, així com la informació bàsica del mateix, en format JSON en el cos de la petició. Normalment, aquesta informació es proporciona a través del client principal de Kubernetes **kubectl** amb un arxiu .yaml. **Kubectl** converteix aquesta informació a JSON quan realitza la crida a la API.

Un exemple d'un manifest .yaml on es declara un *Deployment* seria el següent:

```yaml
apiVersion: apps/v1 # Usa apps/v1beta2 para versiones anteriores a 1.9.0
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # indica al controlador que ejecute 2 pods
  template:    # indica al controlador que executi 2 pods
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Una forma de crear un Deployment utilitzant un arxiu .yaml com l'indicat adalt, es realitzaria executant la comanda **`kubectl apply`** pasant-li l'arxiu .yaml com argument:

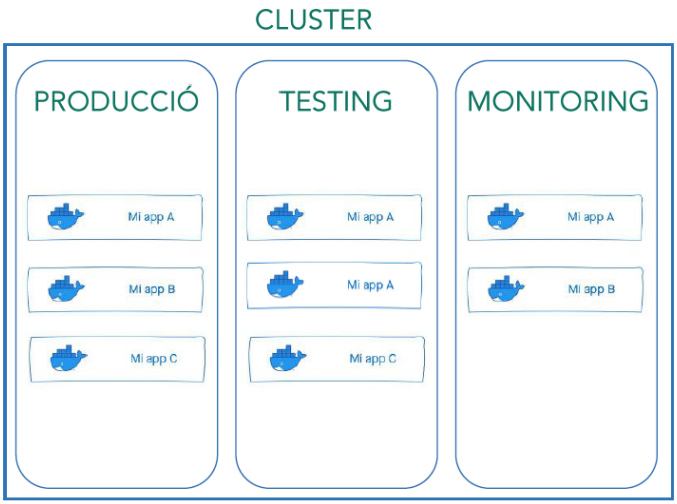
```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml --record
```

Tots els objectes de la API REST de Kubernetes s'identifiquen mitjançant un Nom i un UID.

- Noms: Cadena de caràcters proporcionada pel client que identifica un objecte a la URL d'un recurs, com per exemple, /api/v1/pods/nom-del-objecte. Els noms dels objectes son únics per cada tipus d'objecte. Tot i així, si s'elimina l'objecte, es pot crear un nou amb el mateix nom.
- UIDs: Una cadena de caràcters generada per Kubernetes per identificar objectes de forma única. Cada objecte creat al llarg de tota la vida d'un clúster, té un UID diferent. Està pensat per distingir entre ocurrences històriques d'entitats similars.

Espais de noms (namespaces)

Kubernetes soporta múltiples clústers virtuals recolzats per el mateix clúster físic. Aquests clústers virtuals s'anomenen espais de noms (namespaces).



Quan utilitzar múltiples espais de noms

Els *namespaces* estan pensats per utilitzar-se en entorns amb molts usuaris distribuïts entre múltiples equips o projectes. Proporcionen un camp d'acció pels noms. Els noms dels recursos han de ser únics dins de cada espai de noms, però no entre aquests espais.

Els espais de noms son una forma de dividir els recursos del cúster entre múltiples usuaris.

Kubernetes es crea amb tres espais de noms inicialment:

- Default: L'espai de noms per defecte pels objectes que no especifiquen cap espai de noms.
- Kube-system: L'espai de noms per aquells objectes creats pel sistema de Kubernetes.
- Kube-public: Aquest *namespace* es crea de forma automàtica i es llegible per tots els usuaris (incloent usuaris no autenticats). Es reserva principalment per us intern de clúster, en cas de que alguns recursos necessitin ser visibles i llegibles de forma pública per tot el clúster.

Etiquetes i selectors

Les etiquetes son parells de clau/valor que s'associen als objectes, com els pods. El propòsit de les etiquetes és la de permetre identificar atributs dels objectes que son rellevants i significatius pels usuaris, però que no tenen significat pel sistema principal. Es poden utilitzar les etiquetes per organitzar i seleccionar subconjunts d'objectes.

```
"metadata": {
  "labels": {
    "key1" : "value1",
    "key2" : "value2"
  }
}
```

Les etiquetes permeten que els usuaris mapegin les seves estructures organitzacionals en els propis objectes sense acoplament, sense forçar als clients a emmagatzemar aquests mapejos.

Els desplegaments de serveis i processos en lots solen requerir sovint la gestió d'entitats multi-dimensionals (ex., múltiples particions o desplegaments, múltiples entregues, múltiples capes, múltiples microserveis per capa). Aquesta gestió sovint requereix d'operacions horitzontals

que trenquen la encapsulació de representacions estrictament jeràrquiques, especialment jerarquies determinades per la infraestructura en contes de pels usuaris.

Exemple d'etiquetes son:

- `"release" : "stable", "release" : "canary"`
- `"environment" : "dev", "environment" : "qa", "environment" : "production"`
- `"tier" : "frontend", "tier" : "backend", "tier" : "cache"`
- `"partition" : "customerA", "partition" : "customerB"`
- `"track" : "daily", "track" : "weekly"`

Al contrari que els noms i UIDs, les etiquetes no garantitzen la unitat. En general, s'espera que molts objectes comparteixin etiquetes.

A través del *selector d'etiqueta*, l'usuari pot identificar un conjunt d'objectes. El selector d'etiqueta és la primitiva principal d'agrupació a Kubernetes.

La API actualment soporta dos tipus de selectors:

- Basats en igualtat: Permeten filtrar per clau i valor d'etiqueta. Els objectes coincidents han de complir cada una de les etiquetes indicades, tot i que poden tenir altres etiquetes addicionalment. Es permeten tres clases d'operadors: *=,* == (els dos representen igualtat) i *!*= (representa desigualtat)
- Basats en conjunt: Permeten filtrar les claus en base a un conjunt de valors. Es poden identificar tres tipus d'operadors: *in*, *notin* i *exists*.

Anotacions

Les anotacions permeten adjuntar metadades arbitraries als objectes, de tal forma que clients com eines i llibreries, poden obtenir fàcilment aquestes metadades.

Les etiquetes poden utilitzar-se per seleccionar objectes i per trobar col·leccions d'objectes que cumpleixin certes condicions. Les anotacions en canvi, no s'utilitzen per identificar i seleccionar ibjectes. Les metadades d'una anotació poden ser petites o grans, estructurades o no estructurades, i poden incloure caràcters no permesos en les etiquetes. Les anotacions, al igual que les etiquetes, son mapes de clau/valor:

```
"metadata": {
  "annotations": {
    "key1" : "value1",
    "key2" : "value2"
  }
}
```

Alguns exemples d'informació que es pot indicar com anotació són:

- Camps gestionats per una capa de configuració declarativa, permetent diferenciar-ho dels valors per defecte establerts per clients o servidors, a més dels camps autogenerats i els camps modificats per sistemes d'auto-escalat.
- Informació sobre la construcció, entrega o imatge com marques de dates, IDs d'entrega, branca de Git, funcions hash d'imatges, etc.
- Referències als repositoris de traces, monitorització, analítica o auditoria.

- Informació de la llibreria de client o eina que pot utilitzar-se per depurar el codi: per exemple, nom, versió i informació de construcció.
- Informació d'usuari o procedència d'eina/sistema. Com URLs d'objectes provinents d'altres components de l'ecosistema.
- Metadades per una eina de llançament d'aplicacions: per exemple, configuració o punts de control.
- Numero de telèfon o contacte de les persones a càrreg, o entrades de directori que especifiquen on es pot trobar aquesta informació, com la pàgina web d'un equip de treball.
- Directives de l'usuari final a les implementacions per modificar el comportament o sol·licitar funcionalitats no estàndards.

Una alternativa a les anotacions és emmagatzemar aquest tipus d'informació en una base de dades externes o un directori, però això complicaria la possibilitat de crear llibreries compartides de client, així com eines per el desplegament, gestió, introspecció i similars.

Nodes

Un node és una màquina de treball a Kubernetes. Pot ser una màquina virtual o física, depenent del tipus de clúster. Al utilitzar Kubernetes principalment al Cloud, els nodes solen ser les instàncies o servidors del Cloud en qüestió. Cada node, està gestionat pel component màster i conté els serveis necessaris per executar *pods.* Els serveis en un node inclueixen el container runtime, kubeles i el kube-proxy.

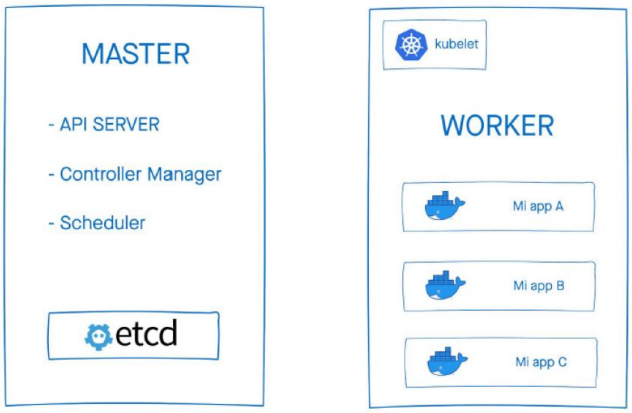
L'estat d'un node compren la seguent informació:

Direccions: Varia depenent del proveidor Cloud i/o de la configuració en màquines locals. Poden ser HostName, ExternalIP, InternalIP.

Estat: Pot ser *OutOfDisk, Ready, MemoryPressure, PIPDPressure, DiskPressure, NetworkUnavailable.*

Capacitat: Descriu els recusos disponibles en el node: CPU, memoria i el nombre màxim de *pods*.

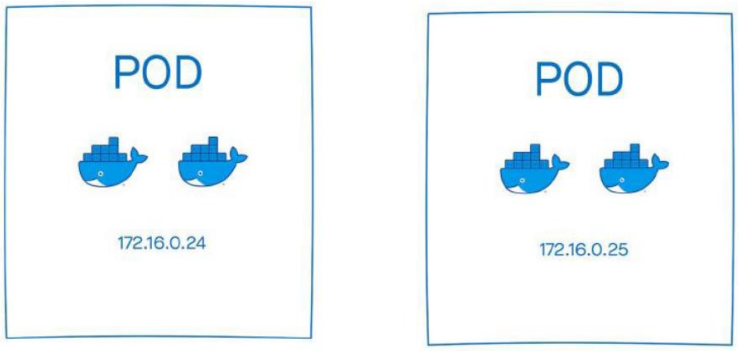
Informació: Informació general sobre el node: versió el kernel,versió de Kubernetes, versió de Docker, etc. Aquesta informació és recollida pel *kubelet* del node en qüestió.



Pods

Els *pods* son la unitat de computació desplegable més petita que es pot crear i gestionar en Kubernetes. És un grup d'un o més contenidors amb emmagatzematge i xarxa compartida, i unes especificacions de com executar els contenidors. Els contenidors d'un *pod* son sempre coubicats, coprogramats i executats en un context compartit.

Els contenidors dins un *pod* comparteixen direcció IP i port, poden trobar-se a través de *localhost*. Les aplicacions dins un *pod* també tenen accés a volums compartits, que es defineixen com part d'un *pod* i estan disponibles per ser muntats en el sistema d'arxius de cada aplicació. Al igual que els contenidors d'aplicacions individuals, els *pods* es consideren entitats relativament efímers. Els *Pods* es creen, s'els assigna un identificador únic (UID) i es planifiquen en nodes on romandran fins la seva finalització o supressió. Quan es diu que quelcom té la mateixa vida útil que un *Pod*, com un volum, vol dir que existeix mentres existeixi aquest *Pod* (amb aquest UID). Si aquest *Pod* s'elimina per qualsevol motiu, inclús si es crea un reemplaçament idèntic, el recurs relacionat (per exemple, el volum) també es destrueix i es crea de nou.



Serveis

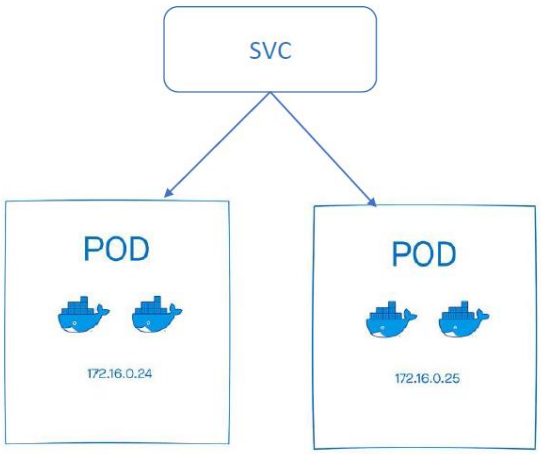
Un servei, és l'objecte de la API de Kubernetes que descriu com s'accedeix a les aplicacions, tal com un conjunt de *Pods*, i que pot descriure ports i balancejadors de càrrega. Amb Kubernetes no necessites modificar la teva aplicació perquè utilitzi un mecanisme de descobriment de serveis desconegut. Kubernetes li dona als *Pods* la seva pròpia direcció IP i un nom DNS per un conjunt de *Pods*, i pot balancejar la càrrega entre ells.

Un servei és una abstracció que defineix un conjunt logic de *Pods* i una política per la qual accedir a ells (semblant al funcionament d'un microservei). El conjunt de *Pods* als que apunta un servei es determina normalment per un *Selector*. Això permet tenir un sol punt d'entrada a un servei que tingui un *backend* d'un o més *Pods* fungibles.

Hi ha 3 tipus de serveis depenent de la connexió que necessiti el conjunt de *Pods* al qual fa referència:

- ClusterIP: Exposa el Servei a una direcció IP interna del clústrer. Aquests només son assolibles des del clúster i s'utilitzen per aplicacions que s'han de connectar amb altres que també estiguin al mateix clúster.
- NodePort: Exposa el servei en cada IP externa del node en un port estàtic. A aquest servei es pot arribar externament amb una petició a `<NodeIP>:<NodePort>`. Automàticament es crea un servei *ClusterIP*, al qual encamina el *NodePort* del servei.

- LoadBalancer: Exposa el servei externament utilitzant un balancejador de càrrega del propi proveïdor del núvol. Es creen automàticament serveis *NodePort* i *ClusterIP*, als quals apuntarà el balancejador extern.



Volums

Els arxius localitzats dins d'un contenidor son efímers, el que presenta problemes per aplicacions no trivials quan s'executa en contenidors. Un segon problema succeeix quan compartim fitxers entre contenidors executant-se junts dins d'un *Pod*. L'abstracció volum de Kubernetes resol els dos problemes.

Un volum és un directori, possiblement amb dades en aquest, que pot ser accessible pels contenidors d'un *Pod*. Com aquest directori es crea, el mitja que el recolza i el contenidor d'aquest es determinen pel tipus de volum utilitzat.

Kubernetes soporta molts tipus de volums. Els tipus de volums efímers tenen el mateix temps de vida que un *Pod,* però els volums persistents existeixen més enllà del temps de vida d'un *Pod*. Aquests volums solen crear un tipus de disc del proveïdor Cloud.

Els volums persistents es solen crear i assignar a un conjunt de *Pods* a través d'un PersistenVolumeClaim(PVC). Un objecte que s'utilitza perquè l'usuari "reclami" un determinat emmagatzematge persistent sense conèixer els detalls de l'entorn del núvol en particular.

ConfigMaps

Un *configmap* és un objecte de la API utilitzat per emmagatzemar dades no confidencials en el format clau-valor. Els *Pods* poden utilitzar els *ConfigMaps* com variables d'entorn, arguments de la línia de comandes o com fitxers de configuració en un volum. S'utilitza un *ConfigMap* per crear una configuració separada del codi de l'aplicació.

Secrets

Els objectes de tipus *Secret* permeten emmagatzemar i administrar informació confidencial, com contrasenyes, tokens OAuth i claus SSH. Conté una petita cantitat de dades confidencials com contrasenyes, un token o una clau. Utilitzant un secret, permet a l'usuari tenim més control sobre com s'utilitza i reduir el risc d'exposició accidental. La informació declarada en un secret , es guarda codificada en Base64.

Controladors

Els controladors permeten desplegar les aplicacions en *Pods* de forma declarativa per poder configurar el desplegament automàtic de diferents rèpliques de *Pods*, volums, ConfigMaps i Secrets. Hi ha diferents tipus de controladors segons la naturalesa que es vol per l'aplicació:

- ReplicaSet: Manté un conjunt estable de rèpliques de Pods executant-se en tot moment, assegurant la disponibilitat d'un número específic de Pods idèntics.
- Deployment: Proporciona actualitzacions declaratives pels Pods i els ReplicaSets.
- StatefulSet: Gestiona el desplegament d'un conjunt de rèpliques de Pods que s'associen a un volum persitent per Pod.
- DaemonSet: Garantitza que tots (o alguns) dels nodes executen una copia d'un Pod. Conforme s'afegeix més nodes al clúster, nous Pods son afegits als mateixos. Si s'eliminen un DaemonSet es netegen tots els Pods que han sigut creats.

ANNEX 2: WAZUH

La plataforma Wazuh ofereix funcions XDR (eXtended Detecion and Response) i SIEM per protegir les càrregues de treball del núvol, contenidors i servidors. Aquests inclouen l'anàlisi de dades de registre, la detecció d'intrusions i programari maliciós, el seguiment de la integritat dels fitxers, l'avaluació de la configuració, la detecció de vulnerabilitats i el suport per al compliment de normativa.

La solució Wazuh es basa en l'agent Wazuh, que es desplega als punts finals supervisats, i en tres components centrals: el servidor Wazuh, l'indexador de Wazuh i el tauler de control de Wazuh.

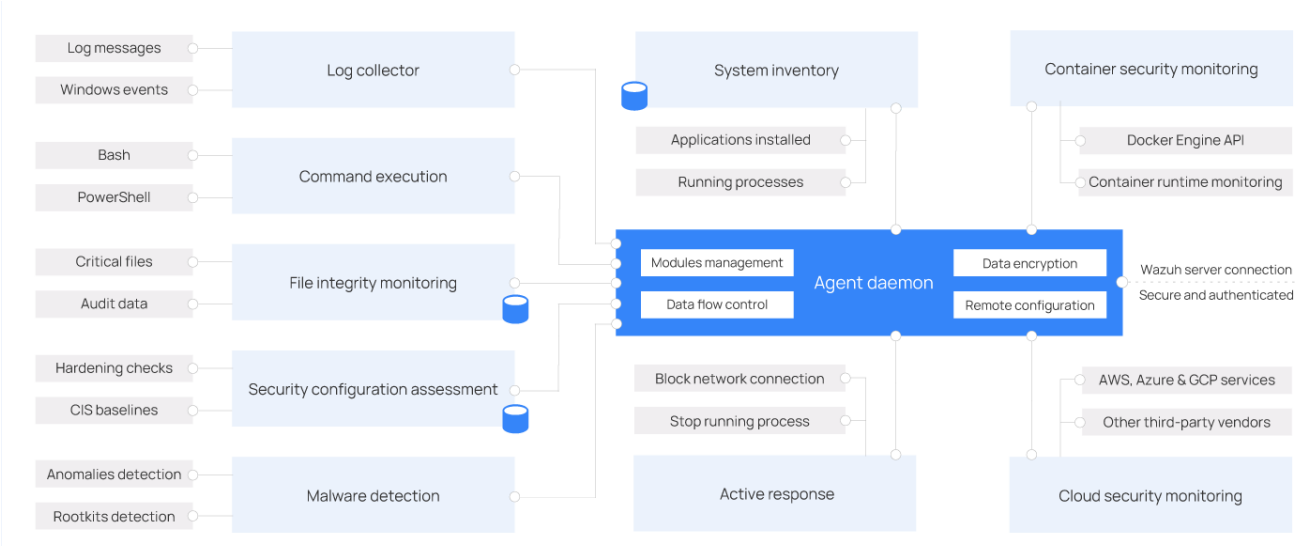
Agent (EDR)

Els agents de Wazuh s'instal·len en punts finals com ara ordinadors portàtils, ordinadors de sobretaula, servidors, instàncies al núvol o màquines virtuals. Proporcionen capacitats de prevenció, detecció i resposta d'amenaces. S'executen en sistemes operatius com Linux, Windows, macOS, Solaris, AIX i HP-UX. L'agent ajuda a protegir el sistema proporcionant capacitats de prevenció, detecció i resposta d'amenaces. També s'utilitza per recollir diferents tipus de dades del sistema i de l'aplicació que reenvia al servidor de Wazuh a través d'un canal encriptat i autenticat.

Arquitectura:

L'agent Wazuh té una arquitectura modular. Cada component s'encarrega de les seves pròpies tasques, inclosa la supervisió del sistema de fitxers, la lectura de missatges de registre, la recollida de dades d'inventari, l'escaneig de la configuració del sistema i la recerca de programari maliciós. Els usuaris poden gestionar els mòduls d'agent mitjançant els paràmetres de configuració, adaptant la solució als seus casos d'ús particulars.

El diagrama següent representa l'arquitectura i els components de l'agent:



Mòduls de l'agent:

Tots els mòduls de l'agent són configurables i realitzen diferents tasques de seguretat. Aquesta arquitectura modular us permet habilitar o desactivar cada component segons les vostres

necessitats de seguretat. A continuació s'especifiquen les diferents finalitats de tots els mòduls d'agent.

- Log collector: Aquest component d'agent pot llegir fitxers de logs plans i esdeveniments de Windows, recopilant missatges de logs d'aplicacions i sistemes operatius. Admet filtres XPath per a esdeveniments de Windows i reconeix formats de diverses línies com els registres d'auditoria de Linux. També pot enriquir esdeveniments JSON amb metadades addicionals.

- Execució d'ordres: Els agents executen ordres autoritzades periòdicament, recopilant la seva sortida i informant-ne al servidor de Wazuh per a una anàlisi posterior. Es pot utilitzar aquest mòdul per a diferents propòsits, com ara controlar l'espai del disc dur que queda o obtenir una llista dels últims usuaris que han iniciat sessió.

- Supervisió de la integritat de fitxers (FIM): Aquest mòdul supervisa el sistema de fitxers i informa quan es creen, s'eliminen o es modifiquen fitxers. Fa un seguiment dels canvis en els atributs del fitxer, els permisos, la propietat i el contingut. Quan es produeix un esdeveniment, captura els detalls de qui, què i quan en temps real. A més, el mòdul FIM (File Integrity Monitoring) crea i manté una base de dades amb l'estat dels fitxers supervisats, permetent que les consultes s'executin de forma remota.

- Avaluació de la configuració de seguretat (SCA): Aquest component proporciona una avaluació contínua de la configuració, utilitzant comprovacions prèvies basades en els punts de referència del Centre de seguretat a Internet (CIS). Els usuaris també poden crear les seves pròpies comprovacions SCA per supervisar i fer complir les seves polítiques de seguretat.

- Inventari del sistema: Aquest mòdul executa periòdicament exploracions, recopilant dades d'inventari com ara la versió del sistema operatiu, les interfícies de xarxa, els processos en execució, les aplicacions instal·lades i una llista de ports oberts. Els resultats de l'escaneig s'emmagatzemen a bases de dades SQLite locals que es poden consultar de forma remota.

- Detecció de programari maliciós: Mitjançant un enfocament no basat en signatures, aquest component és capaç de detectar anomalies i la possible presència de rootkits. A més, busca processos ocults, fitxers ocults i ports ocults mentre supervisa les crides al sistema.

- Resposta activa: Aquest mòdul executa accions automàtiques quan es detecten amenaces, activant respostes per bloquejar una connexió de xarxa, aturar un procés en execució o suprimir un fitxer maliciós. Els usuaris també poden crear respostes personalitzades quan sigui necessari i personalitzar, per exemple, respostes per executar un binari en una caixa de proves, capturar trànsit de xarxa i escanejar un fitxer amb un antivirus.

- Supervisió de seguretat de contenidors: Aquest mòdul d'agent està integrat amb l'API de Docker Engine per supervisar els canvis en un entorn en contenidors. Per exemple, detecta canvis a les imatges del contenidor, la configuració de la xarxa o els volums de dades. A més, alerta sobre els contenidors que s'executen en mode privilegiat i sobre els usuaris que executen ordres en un contenidor en execució.

- Supervisió de seguretat al núvol: aquest component supervisa proveïdors de núvol com Amazon AWS, Microsoft Azure o Google GCP. Es comunica de manera nativa amb les seves API. És capaç de detectar canvis a la infraestructura del núvol (p. ex., quan es crea un nou usuari, quan es modifica un grup de seguretat, quan s'atura una instància del núvol, etc.) i recollir dades de registre de serveis al núvol (p. ex., AWS Cloudtrail, AWS Macie, AWS GuardDuty, Azure Active Directory, etc.)

Comunicació amb el servidor de Wazuh

L'agent de Wazuh es comunica amb el servidor de Wazuh per enviar dades recopilades i esdeveniments relacionats amb la seguretat. A més, l'agent envia dades operatives, informant de la seva configuració i estat. Un cop connectat, l'agent es pot actualitzar, supervisar i configurar de forma remota des del servidor Wazuh.

La comunicació de l'agent amb el servidor es realitza a través d'un canal segur (TCP o UDP), proporcionant encriptació i compressió de dades en temps real. A més, inclou mecanismes de control de flux per evitar inundacions, posar en cua els esdeveniments quan sigui necessari i protegir l'ample de banda de la xarxa.

S'ha de registrar l'agent abans de connectar-lo al servidor per primera vegada. Aquest procés proporciona a l'agent una clau única que s'utilitza per a l'autenticació i el xifratge de dades.

Components centrals

Wazuh Indexer:

L'indexador Wazuh és un motor d'anàlisi i cerca de text complet molt escalable, basat en el conegut motor d'anàlisi i cerca de text Elasticsearch. Aquest component central de Wazuh indexa i emmagatzema les alertes generades pel servidor de Wazuh i proporciona capacitats d'anàlisi i cerca de dades gairebé en temps real. L'indexador Wazuh es pot configurar com a clúster d'un sol node o de diversos nodes, proporcionant escalabilitat i alta disponibilitat.

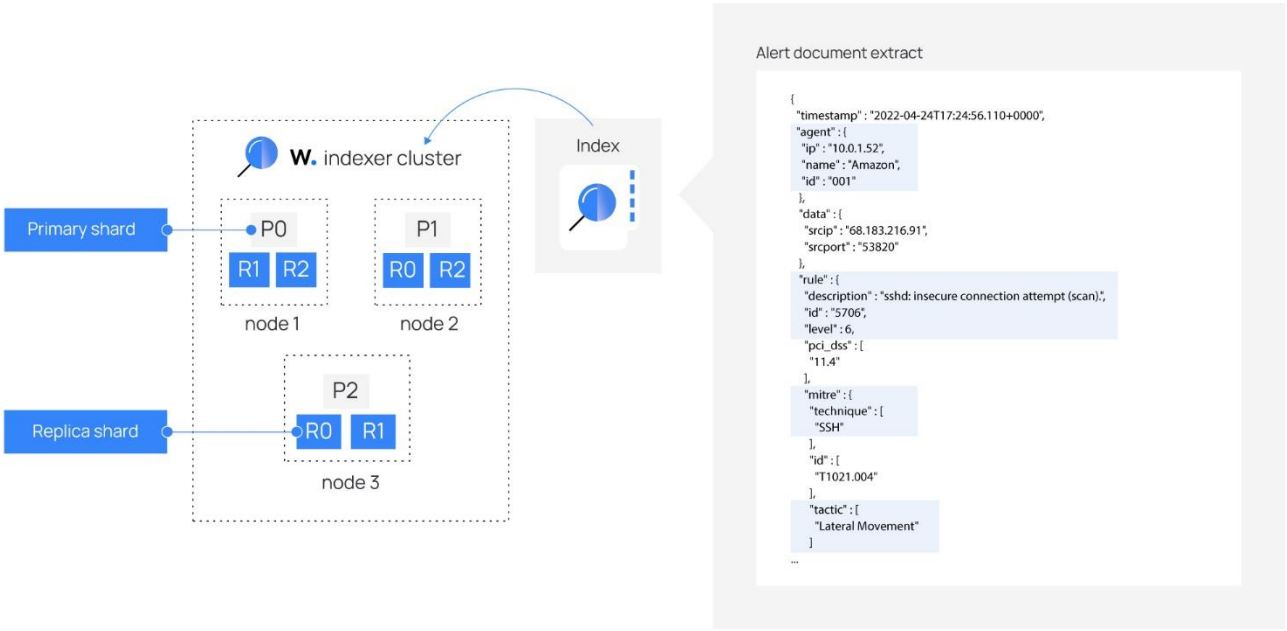
L'indexador de Wazuh emmagatzema dades com a documents JSON. Cada document correlaciona un conjunt de claus, noms de camp o propietats, amb els seus valors corresponents que poden ser cadenes, números, booleans, dates, matrius de valors, geolocalitzacions o altres tipus de dades.

Un índex és una col·lecció de documents relacionats entre si. Els documents emmagatzemats a l'indexador de Wazuh es distribueixen en diferents contenidors coneguts com a *shards*. Mitjançant la distribució dels documents entre diversos fragments i distribuïnt aquests fragments entre diversos nodes, l'indexador de Wazuh pot garantir la redundància. Això protegeix el vostre sistema contra errors de maquinari i augmenta la capacitat de consultes a mesura que s'afegeixen nodes a un clúster.

Wazuh utilitza quatre índexs diferents per emmagatzemar diferents tipus d'esdeveniments:

- Wazuh-alerts: Emmagatzema les alertes generades pel servidor de Wazuh. Es creen cada vegada que un esdeveniment dispara una regla amb una prioritat prou alta (aquest llindar és configurable).
- Wazuh-archives: Emmagatzema tots els esdeveniments (dades d'arxiu) rebuts pel servidor de Wazuh, independentment de si llença una regla o no.

- Wazuh-monitoring: Emmagatzema dades relacionades amb l'estat de l'agent de Wazuh al llarg del temps. La interfície web l'utilitza per representar quan els agents individuals estan o han estat actius, desconnectats o mai connectats.
- Wazuh-statistics: Emmagatzema dades relacionades amb el rendiment del servidor de Wazuh. La interfície web l'utilitza per representar les estadístiques de rendiment.



Es pot interactuar amb el clúster d'indexador de Wazuh mitjançant l'API REST de l'indexador de Wazuh, que ofereix molta flexibilitat. Es pot realitzar cerques, afegir o suprimir documents, modificar índexs i molt més.

Aquí es descriu un exemple d'una consulta a l'indexador de Wazuh que retorna l'última alerta d'un atac de moviment lateral mitjançant la tècnica SSH:

```
GET /wazuh-alerts-4.x-*/_search
{
  "query": {
    "bool": {
      "must": [
        {"term": { "rule.mitre.tactic": "Lateral Movement" } },
        {"term": { "rule.mitre.technique": "SSH" } }
      ]
    }
  },
  "sort": [
    { "timestamp": { "order": "desc" } }
  ],
  "size": 1
}
```

L'indexador de Wazuh és adequat per a casos d'ús sensibles al temps, com ara l'anàlisi de seguretat i el seguiment de la infraestructura, ja que és una plataforma de cerca gairebé en temps real. La latència des que s'indexa un document fins que es pot cercar és molt curta, normalment un segon.

A més de la seva velocitat, escalabilitat i resiliència, l'indexador Wazuh té diverses funcions integrades potents que fan que l'emmagatzematge i la cerca de dades siguin encara més eficients, com ara acumulacions de dades, alertes, detecció d'anomalies i gestió del cicle de vida de l'índex.

Wazuh Server:

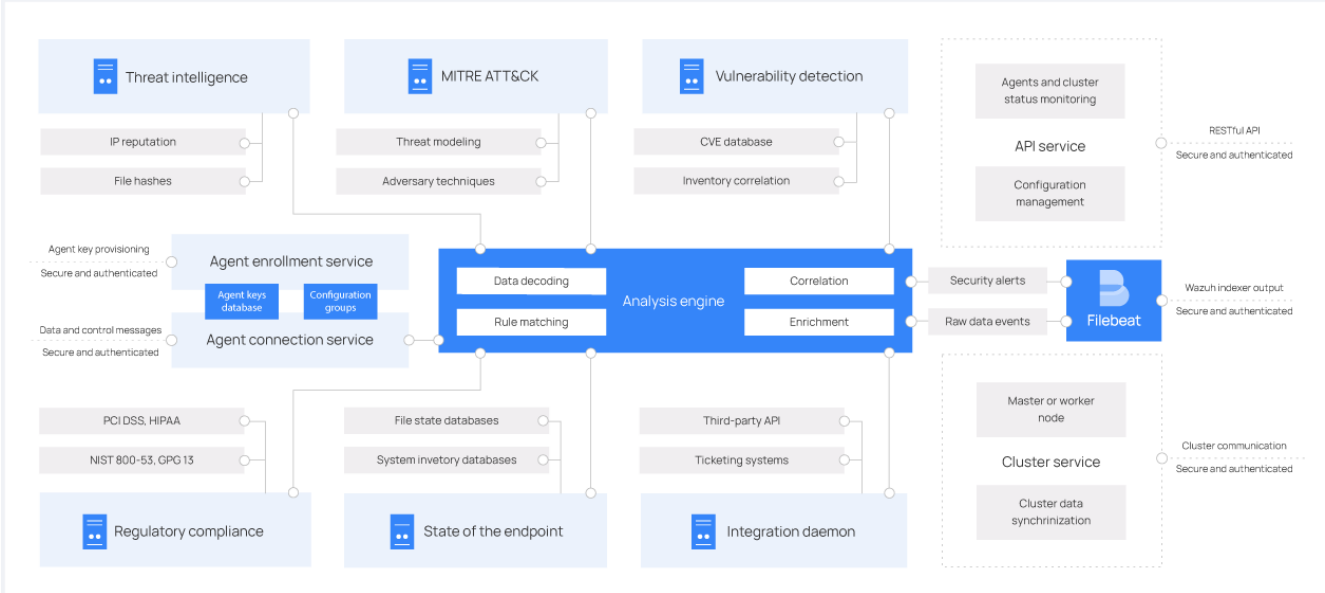
El servidor de Wazuh analitza les dades rebudes dels agents. El processa mitjançant descodificadors i regles, utilitzant la intel·ligència d'amenaces per buscar indicadors de compromís (IOC) coneguts. Un sol servidor pot analitzar dades de centenars o milers d'agents i escalar horitzontalment quan es configura com a clúster. Aquest component central també s'utilitza per gestionar els agents, configurant-los i actualitzant-los de forma remota quan sigui necessari.

El servidor Wazuh utilitza fonts d'intel·ligència d'amenaces per millorar les seves capacitats de detecció. També enriqueix les dades d'alerta utilitzant el marc MITRE ATT&CK i els requisits de compliment normatiu com ara PCI DSS, GDPR, HIPAA, CIS i NIST 800-53, proporcionant un context útil per a l'anàlisi de seguretat.

A més, el servidor Wazuh es pot integrar amb programari extern, inclosos sistemes de *ticketing* com ServiceNow, Jira i PagerDuty, així com plataformes de missatgeria instantània com Slack. Aquestes integracions són convenients per racionalitzar les operacions de seguretat.

El servidor Wazuh executa el motor d'anàlisi, l'API RESTful de Wazuh, el servei d'inscripció d'agents, el servei de connexió d'agents, el dimoni del clúster de Wazuh i Filebeat. El servidor s'instal·la en un sistema operatiu Linux i normalment s'executa en una màquina física autònoma, màquina virtual, contenidor docker o instància del núvol.

El diagrama següent representa l'arquitectura i els components del servidor:



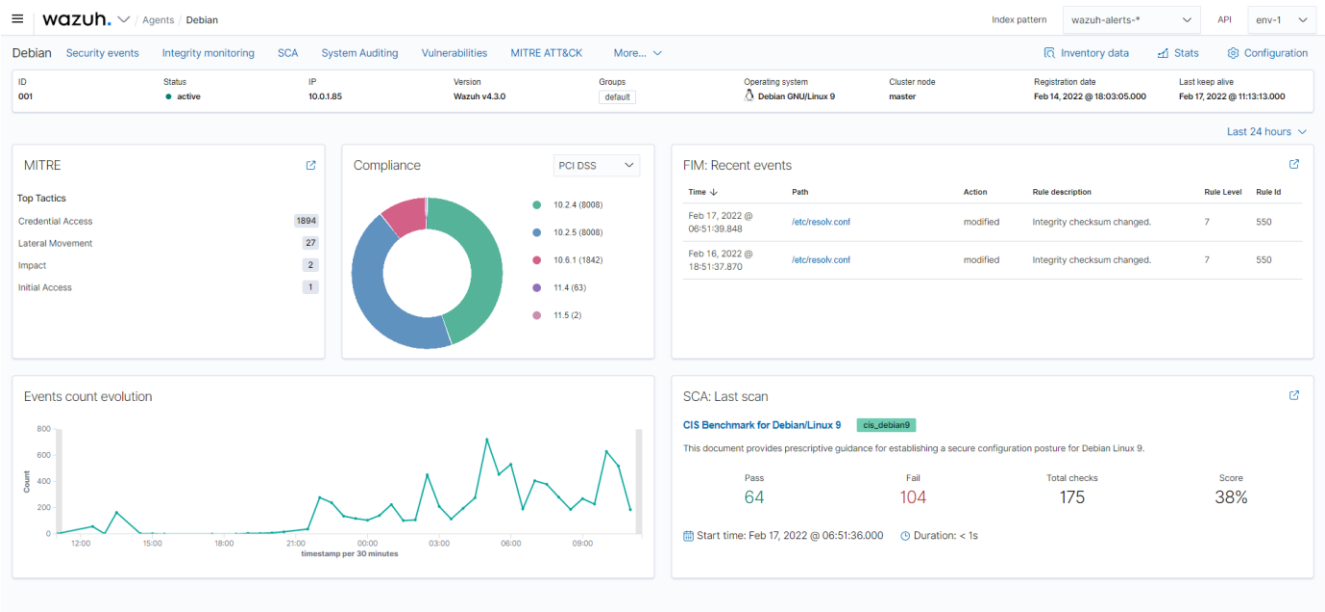
- El servidor de Wazuh inclou diversos components que s'enumeren a continuació que tenen diferents funcions, com ara registrar nous agents, validar la identitat de cada agent i xifrar les comunicacions entre l'agent de Wazuh i el servidor de Wazuh.
- Servei d'enrollment d'agents: S'utilitza per inscriure nous agents. Aquest servei proporciona i distribueix claus d'autenticació úniques a cada agent. El procés s'executa com un servei de xarxa i admet l'autenticació mitjançant certificats TLS/SSL o proporcionant una contrasenya fixa.
- Servei de connexió d'agents: aquest servei rep dades dels agents. Utilitza les claus compartides pel servei d'inscripció per validar la identitat de cada agent i xifrar les

comunicacions entre l'agent de Wazuh i el servidor de Wazuh. A més, aquest servei proporciona una gestió centralitzada de la configuració, que us permeten impulsar nous paràmetres de l'agent de forma remota.

- Motor d'anàlisi: Aquest és el component del servidor que realitza l'anàlisi de dades. Utilitza descodificadors per identificar el tipus d'informació que s'està processant (esdeveniments de Windows, registres SSH, registres del servidor web i altres). Aquests descodificadors també extreuen elements de dades rellevants dels missatges de registre, com ara l'adreça IP d'origen, l'identificador d'esdeveniment o el nom d'usuari. Aleshores, mitjançant regles, el motor identifica patrons específics en els esdeveniments descodificats que podrien activar alertes i possiblement fins i tot demanar contramesures automatitzades (per exemple, prohibir una adreça IP, aturar un procés en execució o eliminar un artefacte de programari maliciós).
- API RESTful de Wazuh: aquest servei proporciona una interfície per interactuar amb la infraestructura de Wazuh. S'utilitza per gestionar els paràmetres de configuració d'agents i servidors, supervisar l'estat de la infraestructura i la salut general, gestionar i editar decodificadors i regles de Wazuh i consultar l'estat dels punts finals supervisats. El tauler de control de Wazuh també l'utilitza.
- Daemon de clúster de Wazuh: aquest servei s'utilitza per escalar els servidors de Wazuh horitzontalment, desplegant-los com a clúster. Aquest tipus de configuració, combinada amb un equilibrador de càrrega de xarxa, proporciona una alta disponibilitat i equilibri de càrrega. El dimoni del clúster de Wazuh és el que utilitzen els servidors de Wazuh per comunicar-se entre ells i mantenir-se sincronitzats.
- Filebeat: s'utilitza per enviar esdeveniments i alertes a l'indexador de Wazuh. Llegeix la sortida del motor d'anàlisi Wazuh i envia esdeveniments en temps real. També proporciona equilibri de càrrega quan es connecta a un clúster d'indexadors Wazuh de diversos nodes.

Wazuh dashboard:

El tauler de control de Wazuh és la interfície d'usuari web per a la visualització i l'anàlisi de dades. Inclou quadres de comandament predefinits per a esdeveniments de seguretat, compliment normatiu (p. ex., PCI DSS, GDPR, CIS, HIPAA, NIST 800-53), aplicacions vulnerables detectades, dades de supervisió de la integritat dels fitxers, resultats de l'avaluació de la configuració, monitorització de la infraestructura del núvol. esdeveniments, i altres. També s'utilitza per gestionar la configuració de Wazuh i controlar el seu estat.



A més de les capacitats de monitorització basades en agents, la plataforma Wazuh pot supervisar dispositius sense agent com ara tallafocs, commutadors, encaminadors o IDS de xarxa, entre d'altres. Per exemple, les dades d'un registre del sistema es poden recollir a través de Syslog i la seva configuració es pot controlar mitjançant un sondeig periòdic de les seves dades, mitjançant SSH o mitjançant una API.

El diagrama següent representa els components i el flux de dades de tots els components de Wazuh.