

SECURING K8S CLUSTER ON AWS WITH WAZUH

DORRA ELBOUKARI

Acknowledgement

I am perpetually thankful to my God ALLAH the almighty, who enlightened my brain with knowledge, and my heart with faith.

A dedication to dear parents, for their encouraging words.

A dedication to the legendary sisters that I have, the best treasure a person could hope for, who rose brilliantly to my impossible challenges and who gave me a moral support with their unflagging efforts.

A debt of gratitude to Mr. Nabil Tabbane for being my number one source of information on countless topics, and to every teacher in the Higher School of Telecommunications Sup'Com for their important role in building my knowledge.

My greatest thanks to my dear companion who gave me love and support, and who strengthened my hope and my ambition for adventure and science.

Contents

Liste of figures	ix
Liste of tables	x
Liste of Acronyms	xi
1 Securing Containerized Infrastructure On The Cloud	3
1.1 Infrastructure	3
1.1.1 On-Premises	3
1.1.2 Private Cloud	3
1.1.3 Public Cloud	4
1.1.4 Hybrid cloud	4
1.1.5 On-Premises VS Cloud	4
1.2 Containers	5
1.2.1 Container Engine	5
1.2.2 Why Docker	6
1.2.3 Container Orchestration	6
1.3 Security Information and Event Management(SIEM)	7
1.3.1 What is a SIEM?	7
1.3.2 Why Wazuh?	7
2 Platform Description	8
2.1 Kubernetes	8
2.1.1 Kubernetes deep dive	8
2.1.2 Kubernetes Architecture	9
2.2 Implementation on a hybrid infrastructure	9
2.2.1 Kubernetes implementation options	9
2.2.1.1 Kubernetes From Scratch	9
2.2.1.2 Amazon EKS	10

2.3	Deploying Containerized Applications on Kubernetes cluster	10
2.3.1	Types of Applications	10
2.3.2	Sample of Multi-tier to Use	11
2.4	Wazuh	11
2.4.1	What is Wazuh ?	11
2.4.1.1	What Does Wazuh Use?	11
2.4.2	What Are The Features Provided By Wazuh	12
2.4.3	Types of Wazuh implementation	13
2.4.4	Wazuh Components	15
3	Installation and configuration of the implementation	17
3.1	Building Amazon EKS with AWS CloudFormation (IaaS) with eksctl tool	17
3.1.1	What is eksctl	17
3.1.2	Installing eksctl	17
3.1.3	Create IAM user account in AWS	18
3.1.4	Connect eksctl to AWS IAM user Account	20
3.1.5	Installing the Cluster	22
3.1.6	Some Troubleshooting:	25
3.2	Deploying multi-tier application on Amazon EKS	28
3.3	Installing Wazuh server and connect to the agent	30
3.3.1	Installing Wazuh Repository	30
3.3.2	Installing the Wazuh manager	32
3.3.3	Installing Elasticsearch	33
3.3.4	Certificates Creation	33
3.3.5	Installing Filebeat	35
3.3.6	Installing Kibana	37
3.3.7	Secure Wazuh Access on the Browser	39
3.3.8	Connect Wazuh server To the agent	41
3.3.8.1	Expose our local Wazuh server to the internet	42
3.3.8.2	Expose Frontend tier of our Application to the Wazuh server through the internet	47
3.3.8.3	Establish the connection between Wazuh server and Fron- tend Tier Worker Node	51

4	Platform Vulnerability, Attack Scenarios, and recommendations	56
4.1	Vulnerability detection	56
4.1.1	How It Works?	56
4.1.1.1	What is a CVE?	57
4.1.2	Scan Types:	57
4.2	DoS attack scenario	58
4.2.1	What is a DoS attack	58
4.2.2	DoS Attack scenario	58
4.2.3	Wazuh Outcome:	62
4.3	Network Scanning	63
5	Conclusion et perspectives	65
5.1	Limitations	65
5.1.1	Ngrok is good, but....	65
5.1.2	Whitelisting is nearly impossible	65
5.1.3	Wazuh is an excellent HIDS , but	65
	General Conclusion	71

List of Figures

1.1	Magic Quadrant for Cloud Infrastructure and Platform Services [2]	5
2.1	Kubernetes Cluster Architecture [1]	9
2.2	Visualize the Conceptual Difference between the Monolithic Architecture and the Microservice Architecture	10
2.3	Components of Monolithic Architecture Vs Components of Microservice Architecture	11
2.4	tree of installation possibilities	14
2.5	Types of Wazuh implementation[3]	14
2.6	Wazuh Amazon AMI in AWS Marketplace	15
2.7	Wazuh Amazon AMI prices	16
2.8	Component required to Install Wazuh server all-in-one	16
3.1	Find-out Kali Distribution	17
3.2	1. Download and extract the latest release of eksctl with the following command	18
3.3	2. Move the extracted binary to /usr/local/bin	18
3.4	3. Test that your installation was successful with the version attribute	18
3.5	1. From AWS services Menu , select IAM	18
3.6	2. Select Users and then add users	18
3.7	3. Give a User name for your IAM User account and select the AWS access type	19
3.8	4.Set the permissions for your account , in our case, KubeAdministrator will belong to admins group and gain their attached policies	19
3.9	5. Add Tags , this step is optional. But is useful for root account to differentiate the other users and to know their purpose of creation.	19
3.10	6. Download the .csv file that contains the Credentials related to the user: User name and Login link	20
3.11	7. You can see clearly that the new IAM user account is created successfully.	20
3.12	1. To get the necessary access credentials, go to Security credentials, then lick on create access key	20

3.13	2. At this point, download the .csv file that contain your Access key ID and your Secret access key.	21
3.14	3. Connect your IAM account to your local machine.	21
3.15	4. Test the connection by viewing the list of IAM users	21
3.16	You can see the account credentials in the following file : .aws/credentials	21
3.17	What The Credentials file will look like	21
3.18	The region in which we will deploy our architecture is defined in the file /.aws/config	22
3.19	Searching for Amazon EKS on AWS Management Console	22
3.20	Amazon EKS Service Dashboard	22
3.21	Available Amazon EKS Clusters	22
3.22	1. Explore the options given by the eksctl tool. We can see clearly that we can use eksctl create ***	23
3.23	2. Integrate the wanted parameters into the cluster creation command just as following	23
3.24	CloudFormation Starts building the Amazon EKS cluster	24
3.25	Here you can see clearly that the deployment succeeded	24
3.26	You can view the progress of your cluster with AWS CloudFormation . . .	24
3.27	Install kubectl	24
3.28	Inspect Available pods on the Amazon EKS Clusters	25
3.29	Troubleshooting 'CREATE_FAILED'	25
3.30	Some required IAM permissions for the user who is building the Amazon EKS with eksctl	26
3.31	Troubleshooting 'Error: invalid version'	26
3.32	Available Amazon EKS Kubernetes versions[4]	27
3.33	Deleting Amazon EKS cluster with one command	27
3.34	Troubleshooting: Unable to build an Amazon EKS Cluster with the same name of a previously deleted one	28
3.35	Implemented Architecture of the a Hybrid environment	29
3.36	1. Install the necessary packages for the installation:	30
3.37	2. apt-key is deprecated	30
3.38	3. Install the GPG key with wget command	30
3.39	4. Inspect the downloaded	31
3.40	5. Create New GPG Keyring	31
3.41	6. Verify and move GPG Keyring to trusted config directory	31
3.42	7. Add the repository	31
3.43	1. Install the Wazuh manager package	32
3.44	2. Enable and start the Wazuh manager service	32

3.45	Run this command to check if the Wazuh manager is active	32
3.46	1. Install Elasticsearch OSS and Open Distro for Elasticsearch	33
3.47	2. Download the configuration file, Users and roles for Elasticsearch	33
3.48	1. Manage certificates	33
3.49	2.Mitigate Log4j2 vulnerability:	34
3.50	3.Enable and start the Elasticsearch service	34
3.51	4. Run the Elasticsearch securityadmin script to load the new certificates information and start the cluster	34
3.52	Test the installation	35
3.53	5.delete The Open Distro for Elasticsearch performance analyzer plugin	35
3.54	1. Install the Filebeat package	35
3.55	2. Download the preconfigured Filebeat configuration file	36
3.56	Copy the Elasticsearch certificates into /etc/filebeat/certs	36
3.57	3. Enable and start the Filebeat service	36
3.58	4. Test id the installation of Filebeat was successful	37
3.59	1. Install the Kibana package:	37
3.60	2. Download the Kibana configuration file and Create the /usr/share/kibana/data directory	37
3.61	3. Install the Wazuh Kibana plugin. The installation of the plugin must be done from the Kibana home directory	38
3.62	4. Create the /etc/kibana/certs directory and ownership and Copy the Elasticsearch certificates into /etc/kibana/certs	38
3.63	5. Link Kibana socket to privileged port 443	38
3.64	6. Enable and start the Kibana service	38
3.65	1. Access the Wazuh login interface on the browser localhost	39
3.66	2. Connection is not secure	39
3.67	3. Notice the Root Certification Authority certificate that we have previ- ously downloaded	39
3.68	4. Go to the Web Browser Settings (We are using Chrome for this project).Select Manage certificates:	40
3.69	5. Select Authorities section and then Import	40
3.70	6.Select the root-ca.pem certificate that we have already downloaded	41
3.71	7.Check 'Trust this certificate for identifying websites'	41
3.72	Sign-in to Ngrok	42
3.73	Download Ngrok Installation file for Kali linux	43
3.74	Verify Ngrok file download	43
3.75	Extract Ngrok installation file	43
3.76	Authenticate our Ngrok agent	44

3.77 Inspect Ngrok file	44
3.78 Verify if the required ports	44
3.79 Discover possible commands by Ngrok with <code>-help</code> attribute	45
3.80 expose our https local Wazuh server	45
3.81 Display of Public server Information	46
3.82 Login to our Internet exposed Wazuh local server	46
3.83 Local server is exposed successfully to the internet	47
3.84 Application Frontend Tier is an EC2 instance	47
3.85 Security Group	48
3.86 the Security Group attached to the Frontend tier EC2 instance	48
3.87 Inbound Rules	49
3.88 Add the rules	49
3.89 Start by adding rules	50
3.90 Outbound Rules	50
3.91 Successful Update of Security group corresponding to the Frontend Tier EC2 instance	50
3.92 Go to the Session Manager of the Frontend Tier Worker Node (E2)	51
3.93 Establish connection to the Frontend Tier Worker Node (E2)	51
3.94 Go to the Session Manager of the Frontend Tier Worker Node (E2)	52
3.95 Linux Wazuh agent Os characteristics	52
3.96 ngrok Session Status	52
3.97 Fill in the agent parameters 1.0	53
3.98 Fill in the agent parameters 1.0	53
3.99 Fill in the agent parameters 1.1	54
3.100Run the command on FrontEnd Tier worker node	54
3.101Restart Daemon and Wazuh-agent service	55
3.102Connected Agent	55
4.1 Repositories[5]	56
4.2 Vulnerability detection with Feeds from CVE	58
4.3 The Exposed service of frontend tier which we will attack	59
4.4 Service DO NOT accept ICMP protocol as protection measure	59
4.5 Display of our application on the browser	60
4.6 nmap scan did not provide any information about open Ports	60
4.7 nmap deep scan shows an open port	60
4.8 Start Metasploit on kali Linux	61
4.9 Search for the location of synflood auxiliary	61

4.10	use the SYNflood auxiliary	61
4.11	Show and select the SYNflood option to use	62
4.12	Set A spoofable source address, set the targeted IP and Port as well	62
4.13	Start flooding the application with the command exploit	62
4.14	The DOS attack Wasn't identified	62
4.15	The Common asked Question	63
4.16	Professional Response on the Expected outcome	63
4.17	nmap deep scan shows an open port	63
4.18	Port scan is detected by Wazuh	64
5.1	The Frontend MongoExpress Deployment Yaml Definition file	67
5.2	The Backend MongoDB Deployment Yaml Definition file	68
5.3	Internal service Yaml Definition File	68
5.4	External Service Node IP Service Definition File	69
5.5	Secret Yaml Definition File	69
5.6	ConfigMap Yaml Definition File	70

List of Tables

1.1	Comparative Table Between On-premises and Public Cloud Infrastructure	6
-----	---	---

List of Acronyms

AWS: Amazon Web Services

K8S: Kubernetes

EC2: Elastic Compute Cloud

CapEX: Capital Expenditure

OpEX: Operational Expenditure

IaaS: Infrastructure As A Service

PaaS: Platform As A Service

SaaS: Software As A Service

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

Amazon EKS: Amazon Elastic Kubernetes Services

SIEM: Security information and event management

AWS IAM : Amazon Web Services Identity and Access Management (IAM)

CVE : Common Vulnerabilities and Exposures

General Introduction

In this era of exponential competitive markets, organizations are in an endless pursuit of software development approaches that improve productivity at a fast pace. Reaching the best results while minimizing time and effort is the main ingredient in the recipe of successful business. This is crucial for companies to gain popularity and to build a recognizable reputation.

DevOps is the key solution to this issue. Few years ago, this concept emerged to boost software development process and to establish the intended harmony between development and operational teams. DevOps strategies are usually implemented by using a variety of tool that improve the capacity of a business to provide applications and services at a high rate of production.

One of the most reputable DevOps tools is Kubernetes. It's an open source orchestrator that helps to deploy and manage containerized applications. Kubernetes provides as well a beneficial variety of key features such as self-healing for containers, scaling, service discovery, storage management and cluster optimization. All these assets made Kubernetes widely acknowledged by organizations thanks to the big competitive advantage it provides compared to the traditional deployment of applications.

Although the undeniable convenience provided by Kubernetes, this technology faces a myriad of security challenges that need to be controlled. In fact, we live in a world where both internal and external actors are attempting to obtain data for nefarious purposes.

Isn't primordial then, to search for a method to detect and mitigate threats? This project stands for securing Kubernetes Cluster as it is an unquestionable gateway to the immunity of businesses and an open window on an indisputable evolution of their performance. In fact, we will shed the light on Kubernetes Pod's security where the application tiers are running.

Our ultimate goal is to detect and act against malicious attacks on micro services' application deployed on kubernetes distributed system by using Intrusion Detection Systems (IDS) and Security information and event management (SIEM). In this work, we highlight at the first chapter the Bench-marking which precise the various used components and justifies the selected choices. So, we will go through a detailed discussion that explains the profitability of each element, starting by the used infrastructure, passing by the container run-time engine, until the most convenient IDS or SIEM.

In the second chapter, we dive deep in each component infrastructure. Thus, we go deeper to underline the contribution of each granular element and we explain how all the components interact with each other.

Since we explained all the required elements, now, we can apply, thirdly, a specific implementation that will be used later, in chapter four, for several attack scenarios. The output of these attacks on the Wazuh dashboard will be a crystal clear evidence on the major contribution of SIEM in threat prevention.

To crown all, it's obvious that using Wazuh to detect and protect kubernetes pods has a major importance in protecting containerized applications. This includes an active detection of threats and a considerable mitigation of their lethal impact. Unfortunately, this action is not sufficient to protect multi-tier application from external and internal actors. It cannot guarantee an optimal immune status for the deployed application.

This can be fulfilled while considering a complementary security solution that takes in consideration all the granular components of Kubernetes cluster.

Chapter 1

Securing Containerized Infrastructure On The Cloud

Introduction

There are always a huge variety of technologies that can be applied to reach a specific result. In this project, it's critical to make wise choices about the infrastructure on which the project will be built as well as the used Container Engine and the most efficient tool to detect and mitigate security threat. That's the reason why benchmarking is crucial to get the most convenient results.

1.1 Infrastructure

The infrastructure on which the application will run is an important element . It holds the networking devices , computational servers as well as storage solutions. The infrastructure that we will apply can be either on-premises, on the cloud or in hybrid environment.

1.1.1 On-Premises

An On-Premises infrastructure , also called on-site solution is the infrastructure that is established on- site as a data-center. The implementation of this solution makes the owner of the DC as the first and only responsible for its security . The headquarters solution is considered as immune , under personal control, and physically supervised by Network Video Recording (NVR) systems . On-premises infrastructure solution requires capital expenditure CapEX, which means considerable funds to buy, update and maintain the equipment. In other words, a company needs to spends considerable amounts of money not only to acquire devices but to ensure that the environment obeys to the compliance norms related to cooling system, fire alarms, etc.

1.1.2 Private Cloud

A private cloud is a term that refers to a cloud computing infrastructure devoted to a single user group. A private cloud can be hosted in an organization's own data center, in a third-party colocation facility, or through a private cloud provider that specializes in private cloud hosting and may or may not also provide typical public shared multi-tenant cloud architecture.

1.1.3 Public Cloud

Instead of running software on the local devices in your own data-center, cloud gives you the opportunity to take advantage of resources given by Public cloud providers (PCP). As a customer to PCP, you will be introduced to three main types of cloud services:

- **Infrastructure As A Service (IaaS):** Provides infrastructure elements on the cloud which are networking resources, essential compute power and data storage
- **Platform As A Service (PaaS):** is mainly dedicated for developers. It provides a cloud-based development and deployment environment with tools to help you produce everything from simple cloud-based application to powerful, innovative, cloud-enabled business solutions.
- **Software As A Service (SaaS):** is a software distribution paradigm in which a cloud provider hosts and makes programs available and ready to use for end users through the internet.

Nowadays, there is a considerable of Public cloud providers such as: Amazon Web Services (AWS) by Amazon, Azure cloud by Microsoft, Google cloud, etc.

AWS Cloud: To make a good choice of cloud provider, the customer needs to rely on some very precise metrics to make a wise choice. One important metric is the Magic Quadrant for cloud Infrastructure and platform services given by Gartner, every year, view the figure below. A classification of cloud providers is given while relying on several concepts such as the Ability to execute, Completeness of Vision, etc.

No one can deny the fact that AWS is an innovation leader in cloud field. It uses its entire engineering capacity to innovate in new areas. Furthermore, Amazon aims to be Earth's most customer-centric company. Amazonians are Customer obsessed, which means they do their best to fill the customer needs as they start with his requirements and work backwards. All those assets make to use AWS cloud infrastructure above any other cloud provider.

1.1.4 Hybrid cloud

Hybrid cloud is a solution in which an organization takes advantage of the services given by public cloud provider along with the on-premises infrastructure or a hybrid cloud solution. In this project, we will take advantage of this type of cloud infrastructure. In fact, we will establish our Kubernetes cluster on AWS Cloud and we will connect it to the personal computer which is the Wazuh server that will handle monitoring Kubernetes cluster.

1.1.5 On-Premises VS Cloud

To make a wise choice between the to implementation environment, it's important to make a comparative study that highlights the benefits as well as the drawbacks of each solution.

Figure 1: Magic Quadrant for Cloud Infrastructure and Platform Services



Figure 1.1: Magic Quadrant for Cloud Infrastructure and Platform Services [2]

1.2 Containers

Among the myriad of software that we select for the project, we need to make a clear decision about the container Engine.

1.2.1 Container Engine

A container engine is the key element that helps to build, deploy, run and manage containers. In fact, on the same operating system kernel, container engines can run multiple, isolated instances, known as containers. Containers provide a controlled, readily manageable environment for executing programs and dependencies by performing virtualization at the operating system level. By segregating programs, apps, and code from other applications operating on the same physical host, container isolation can improve security.

The Open Container Initiative (OCI) container image format is used by the majority of current container engines. OCI container images are a representation of a container and the software that should execute inside it, allowing for predictable and repeatable container creation.

The container runtime, which connects with the operating system kernel to conduct the containerization process and define access and security policies for running containers, is an important component of a container engine.

There are many possible container engines that can be used such as: Docker, CoreOS rkt, Containerd, CRI-O, etc. In our project, we will use Docker as our main container engine.

	Benefits	Limitations
On-premises	<ul style="list-style-type: none"> • Generally greater protection that with cloud 	<ul style="list-style-type: none"> • Not easy to scale • CapEx is always higher than OpEx • The owner of infrastructure is responsible for all the maintenance, power supply, patches, • Need to hire experts to deal with maintenance
Public Cloud	<ul style="list-style-type: none"> • Shared responsibility for the security of services • Highly scalable • Easier to build architecture • Pay-As-You-Go for on- demand services reduces spent money • No need to pay for physical maintenance of devices. 	<ul style="list-style-type: none"> • Security is a Hot Topic because public cloud is exposed to the internet

Table 1.1: Comparative Table Between On-premises and Public Cloud Infrastructure

1.2.2 Why Docker

The Docker container system has a comprehensive set of capabilities and is available in both free and paid versions, making it the most popular container technology. With its own image specifications, command line interface, and container image building service, it is Kubernetes' default container runtime.

Docker provides a RESTful API for controlling container state. The command-line client uses the API to develop, deploy, and manage container images, and it runs as a daemon on each node. Windows, Linux, and Mac all support it.

1.2.3 Container Orchestration

While running multi-tier applications, with a considerable number of containers, managing all the running containers manually will be a challenging task. Nowadays, organizations need to run many containerized applications with hundreds of microservices to manage. This task becomes more and more complicated and even impossible. The best way to manage all these containers efficiently is to use a **container orchestration solution**.

The deployment, administration, scaling, and networking of containers are all automated via container orchestration. It considerably decreases the work and complexity of administering a large containerized application estate by simplifying the provisioning of containerized apps. Furthermore, orchestration facilitates an agile or DevOps strategy by automating operations, allowing teams to build and deploy new features and capabilities in quick, iterative cycles (Continuous Integration / Continuous Delivery).

1.3 Security Information and Event Management (SIEM)

Another element to choose carefully is the used SIEM . In this paragraph we will explain accurately our choices regarding this component.

1.3.1 What is a SIEM?

SIEM (Security Information and Event Management) is a security system that assists enterprises in identifying possible security threats and vulnerabilities before they interrupt business operations. It detects anomalous user activity and may use artificial intelligence to automate many of the manual procedures involved with threat detection and incident response. It has become a standard in modern security operation centers (SOCs) for security and compliance management use cases.

1.3.2 Why Wazuh?

Wazuh is one of the most reputable open source SIEM for it's stability and it's accuracy. In fact, Wazuh is a A comprehensive SIEM solution.It is a data collection, analysis, and correlation tool that may also be used for threat detection, compliance management, and incident response. It may be used on-premises, in hybrid cloud setups, or on the cloud. This compatibility with the aimed in- frastructure of this project makes it a very convenient tool to exploit. Furthermore, Wazuh is highly flexible and scalable. It has free license and no vendor lock-in. Theses benefits will help us to reach the required results while saving money.

Conclusion

To sum up , implementing our kubernetes cluster on AWS infrastructure and monitoring it with an Open Source SIEM such as Wazuh is the most cost effective solution.

Chapter 2

Platform Description

Introduction

After selecting the technologies that we will use in the previous chapter, we will focus on the characteristics of each component as well as its architecture and its role in our implementation.

2.1 Kubernetes

Kubernetes, also known "K8s," is an open source project that was originally developed by Google. It is an orchestrator that coordinates, manages and organizes the execution of containerized applications over a cluster of servers. Using on-premises infrastructure or public cloud platforms such as AWS, the K8s solution simplifies the deployment and administration of cloud native apps. It automates dynamic container networking and distributes application workloads throughout a Kubernetes cluster. Kubernetes enables as well resiliency by allocating storage and persistent volumes to running containers, offering automated scaling, and continually working to preserve the intended state of applications.

2.1.1 Kubernetes deep dive

Just like any other cluster that you have in mind, Kubernetes is just a set of machines assembled together. They cooperate to run applications smoothly and conveniently. These machines can be physical servers, virtual hosts or cloud instances (EC2 instances for AWS Cloud). Each machine is called a node.

Kubernetes has two types of node :

- Control Plane:

previously called Control Node or Master Node. It implements the most sophisticated features in the cluster. It schedules tasks, keeps track of the state of the cluster and helps in the remediation, recovery and scalability of the applications.

- Worker Node :

It is the node on which the containers are running. This element receives orders from the control plane to delete, run or re-create a container.

2.1.2 Kubernetes Architecture

The architecture of the kubernetes cluster appears as follows:

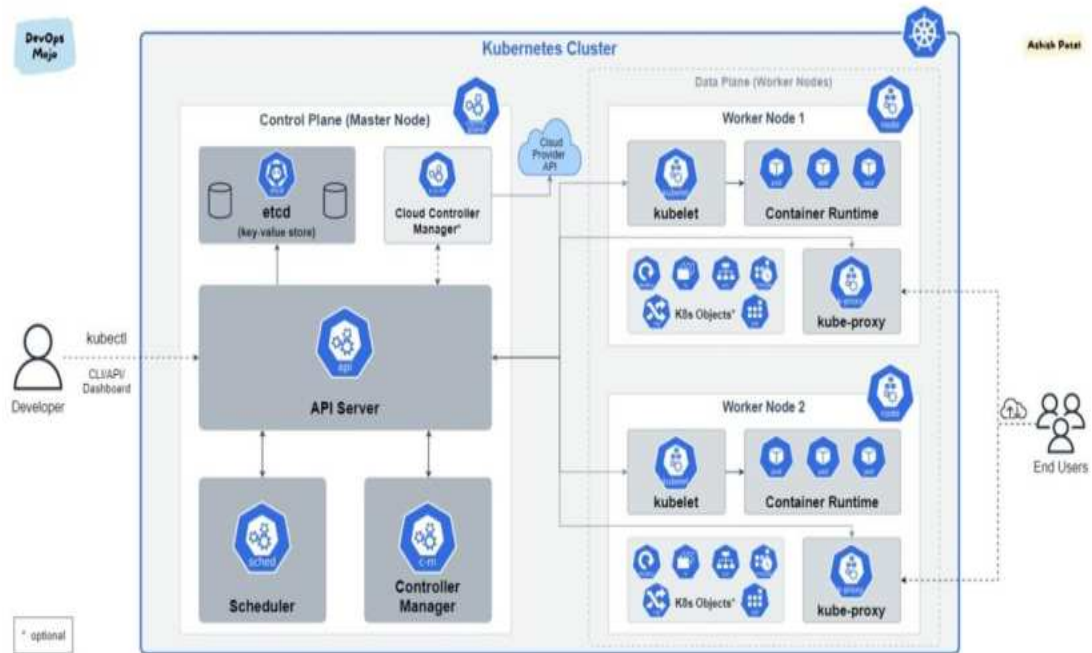


Figure 2.1: Kubernetes Cluster Architecture [1]

2.2 Implementation on a hybrid infrastructure

In this chapter ,we will precise how to implement each element on our hybrid environment. Starting by the cloud hosted elements which are the supervised K8s cluster and the inner deployed application until reaching the on-premises hosted Wazuh server.

2.2.1 Kubernetes implementation options

To run our kubernetes container on AWS cloud, we have two main approaches:

- Building the cluster the hard way from scratch,
- Building the Cluster by using AWS managed service Amazon EKS

2.2.1.1 Kubernetes From Scratch

This type of setup is also called "the hard way". Having a healthy cluster requires installing all the components of the cluster by hand, one by one and then configure them to cooperate together.

2.2.1.2 Amazon EKS

AWS Cloud provides a managed container service called Amazon EKS that helps to run kubernetes and deploy worker nodes as AWS instances EC2 (Elastic Compute Cloud) or deploy them as serverless containers with AWS Fargate.

Amazon EKS dashboard will be viewed on AWS Console, where you can view and explore running Kubernetes applications .

2.3 Deploying Containerized Applications on Kubernetes cluster

The main purpose of building kubernetes is to manage applications while taking advantage of k8s scaling feature as well as the automated deployment. So , what are the application types that can be deployed on kubernetes cluster.

2.3.1 Types of Applications

According to the architecture of the application, we can have two main types to deploy:

- Monolithic Architecture:

It is regarded as an old method of developing applications. A monolithic application is made up of a single, indivisible piece of software. A client-side user interface, a server-side program, and a database are often included in such a system. It is unified, with all functions handled and served from a single location. Monolithic applications are characterized by a single big code base and a lack of modularity. Developers use the same code base when they wish to update or replace something. As a result, they make modifications to the entire stack at the same time.

- Microservices Architecture:

Unlike the monolithic application which is a single cohesive entity, a microservices design divides it into smaller autonomous components. Every application procedure is handled by these units as a distinct service. As a result, each service has its own logic and database, as well as the ability to execute specialized operations.

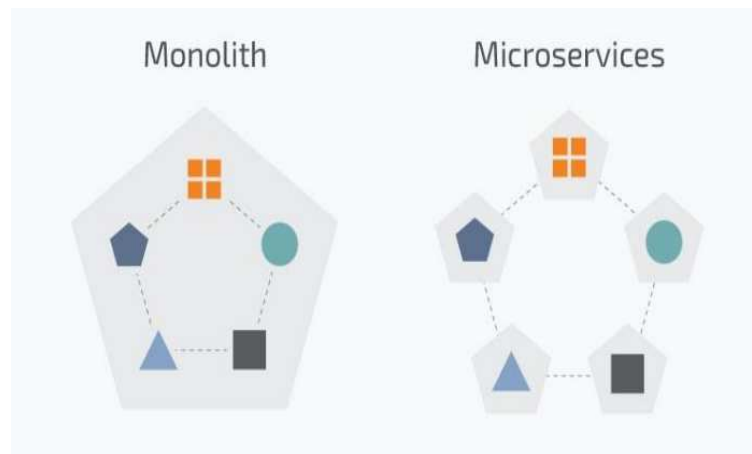


Figure 2.2: Visualize the Conceptual Difference between the Monolithic Architecture and the Microservice Architecture

2.3.2 Sample of Multi-tier to Use

As we have already mentioned ,unlike the Monolithic applications , Microservices applications puts each of the User interface, the business logic and the data access layer in independent tiers called microservices. All the tier work in coordination with each other to give a healthy application thanks to orchestrator .

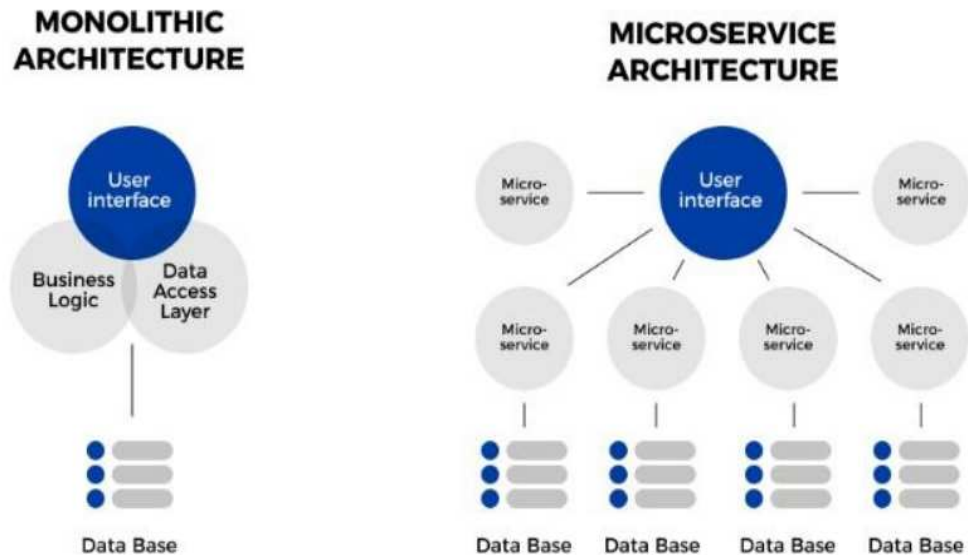


Figure 2.3: Components of Monolithic Architecture Vs Components of Microservice Architecture

In our project, we will use a very simple multi-tier application to provide a proof of concept. Our sample contains a database tier which is MongoDB and a frontend tier which is MongoExpress. MongoExpress contains a graphical interface that is used to Create, Read Update and Delete (CRUD operations) on the DB tier.

2.4 Wazuh

To inspect and supervise the security of kubernetes cluster and the application deployed on it, we will need a security platform that detects and displays all the susceptible threats such as Wazuh.

2.4.1 What is Wazuh ?

Wazuh is a threat detection, integrity monitoring, incident response, and compliance security monitoring system that is free, open source, and enterprise-ready. Wazuh is a flexible, scalable platform with no vendor lock-in or license fees. It offers expert assistance, consulting service and training.

2.4.1.1 What Does Wazuh Use?

We can also consider Wazuh server as a stack that contains Wazuh Manager, Elasticsearch, Kibana (as a Wazuh Kibana plugin) and FileBeat. You will install by hand those elements

, one by one

Wazuh Manager Wazuh began as a derivative of OSSEC, and according to the official documentation, it was designed to be more dependable and scalable. Wazuh detects rootkits using anomaly and signature detection methods, as well as log analysis, integrity checking, Windows registry monitoring, and active response.

OSSEC: OSSEC is a host-based intrusion detection system that is available for free and open-source. Log analysis, integrity checking, Windows registry monitoring, rootkit identification, time-based alerting, and active response are among the functions it provides.

ElasticSearch: Elasticsearch is an Apache Lucene-based distributed search and analytics engine. Elasticsearch has swiftly become the most popular search engine since its introduction in 2010, and it is frequently used for log analytics, full-text search, security intelligence, business analytics, and operational intelligence.

Kibana: Kibana is a free and open-source frontend tool that sits atop the Elastic Stack, allowing users to search and visualize data indexed in Elasticsearch.

FileBeat: Is the element that takes the place of Logstash. It's a lightweight shipper than Fluentd. It's dedicated for forwarding and centralizing log data. It's an open source data collector that allows you to integrate data gathering and consumption from various resources for improved data utilization and comprehension.

2.4.2 What Are The Features Provided By Wazuh

Wazuh is a rich open-source project. It contains many features, such as:

- Containers Security:

Wazuh monitors the activity of your Docker hosts and containers, identifying threats, vulnerabilities, and abnormalities. Users may monitor images, volumes, network settings, and running containers using the Wazuh agent, which has native Docker integration. Wazuh gathers and analyzes extensive runtime data on a continual basis. For instance, alerting for privileged mode containers, vulnerable apps, a shell operating in a container, modifications to persistent volumes or images, and other potential dangers.

- Cloud Security:

Wazuh helps monitoring cloud infrastructure at an API level, using integration modules that are able to pull security data from well known cloud providers, such as Amazon AWS, Azure or Google Cloud. In addition, Wazuh provides rules to assess the configuration of your cloud environment, easily spotting weaknesses. In addition, Wazuh light-weight and multi-platform agents are commonly used to monitor cloud environments at the instance level.

- Security Analytics:

Wazuh is a security data collection, aggregation, indexing, and analysis tool that aids businesses in detecting intrusions, threats, and anomalous activity. As cyber attacks

become increasingly complex, real-time monitoring and security analysis are required to detect and remediate problems quickly. As a result, our light-weight agent performs the required monitoring and response functions, while our server component offers security intelligence and data analysis.

- **Intrusion Detection:**

Wazuh agents search for malware, rootkits, and suspicious abnormalities on the systems they monitor. Hidden files, cloaked processes, and unregistered network listeners, as well as discrepancies in system call answers, can all be detected by them. The server component, in addition to agent capabilities, utilizes a signature-based approach to intrusion detection, analyzing gathered log data and looking for evidence of penetration using its regular expression engine.

- **Log Data Analysis:**

Wazuh agents scan and securely transfer operating system and application logs to a central manager for rule-based analysis and storage. Wazuh rules alert you to application or system problems, misconfigurations, attempted and/or successful harmful operations, policy violations, and other security and operational concerns.

- **File Integrity Monitoring:**

Wazuh keeps an eye on the file system, detecting changes in file content, permissions, ownership, and characteristics that you should be aware of. It also recognizes people and apps that were used to create or change files natively. To discover threats or compromised hosts, file integrity monitoring capabilities can be used with threat intelligence. Moreover, it is required by a number of regulatory compliance requirements, including the PCI DSS.

- **Vulnerability Detection:**

Wazuh agents collect software inventory data and send it to the server, where it is compared to constantly updated CVE (Common Vulnerabilities and Exposures) databases to detect known susceptible software. Automated vulnerability assessment enables you to identify weak spots in your important assets and take remedial action before attackers use them to destroy your business or steal sensitive information.

- **Incident Response:**

When specific conditions are satisfied, Wazuh provides out-of-the-box active reactions to execute different countermeasures to combat active threats, such as limiting access to a system from the threat source. Wazuh may also be used to execute live forensics or incident response duties by remotely running commands or system queries, recognizing indications of compromise (IOCs), and assisting with other live forensics or incident response chores.

- **Regulatory Compliance:**

Wazuh provides some of the necessary security controls to become compliant with industry standards and regulations. These features, combined with its scalability and multi-platform support help organizations meet technical compliance requirements.

2.4.3 Types of Wazuh implementation

To monitor endpoints with Wazuh, we have to consider this tree of installation possibilities with which we can install Wazuh server along with Filebeat, Elasticsearch and kibana.

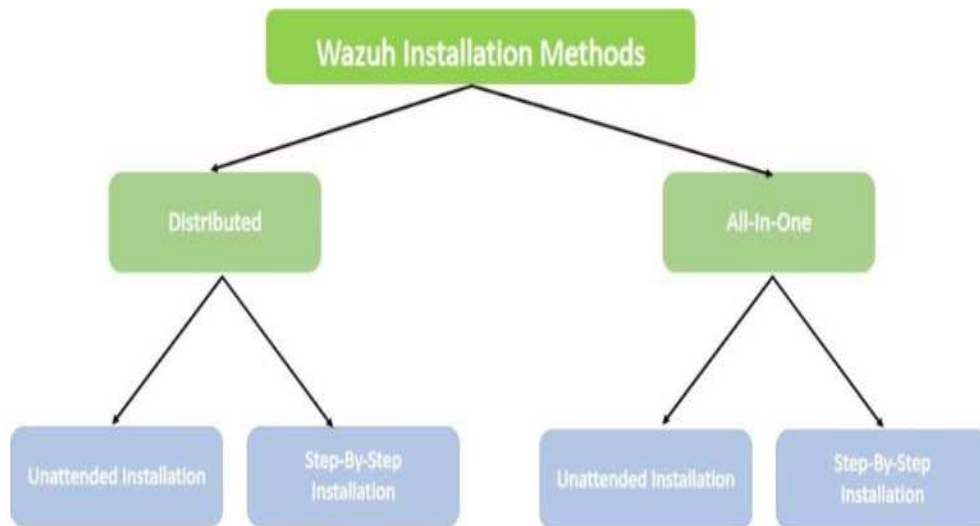


Figure 2.4: tree of installation possibilities

There are two main different approaches to deploy Wazuh Server.

- Distributed:

Each component is installed as a single-node or multi-node cluster on a different host. This sort of deployment ensures the product's high availability and scalability, and it's ideal for big work-groups.

- All-in-one:

The Wazuh server and Elastic Stack are both installed on your system on the same host. You may also use our AMI to launch an EC2 Instance or download our ready-to-use OVA.

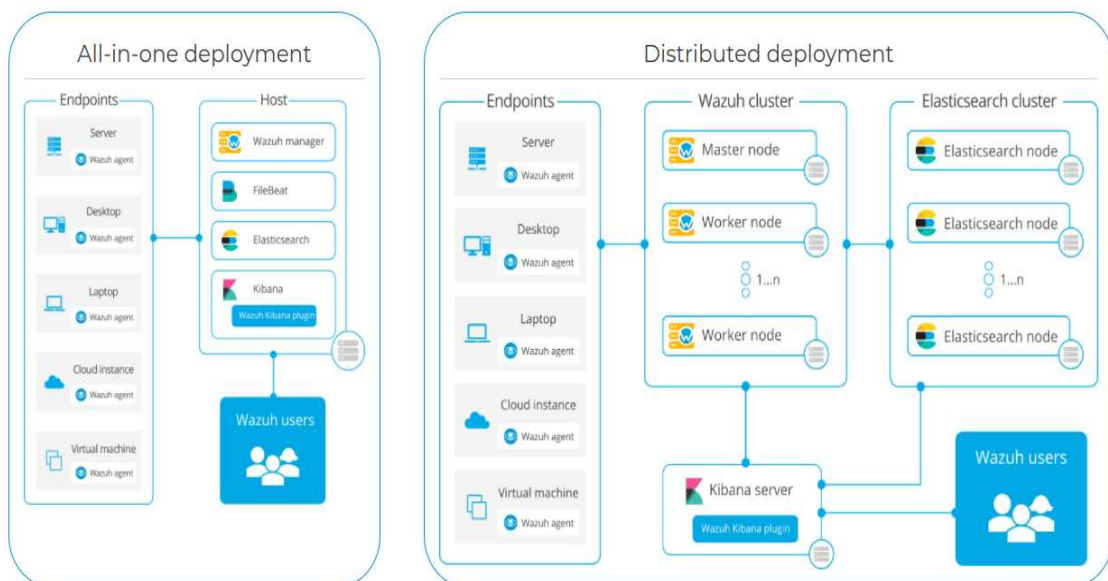


Figure 2.5: Types of Wazuh implementation[3]

2.4.4 Wazuh Components

Wazuh platform contains many features that helps you to monitor and audit many types of endpoints .To fulfill this objective , the three components of wazuh work in harmony for an optimal output. Those three elements are:

- Wazuh Server
- Wazuh agent
- Elastic Stack

In this projet, we will avoid the huge consumption of resources on AWS.

We can easily install Wazuh server and Elastic Stack on an AWS EC2 thanks to the ready-to-use Amazon Machine Image (AMI) But this instance is expected to consume more money than we want to spend. So, we will not deploy Wazuh on EC2 instance.

Anyways, we don't have a huge number of nodes and pods deployed on our the Kubernetes cluster that we will inspect. This is why we will install Wazuh on one an on-site physical server to audit and monitor the Amazon EKS Cluster remotely. So our installation method is All-in-one. Now, we have to select among the next two options:

- Unattended installation:

This method relies on the use of scripts to automate the installation of Wazuh components

- Step-by-step installation :

This method installs Wazuh components one by one. Along with the certificates need to secure the communication.

In our project, we will go in the installation **Step-by-step** to explore more each component.

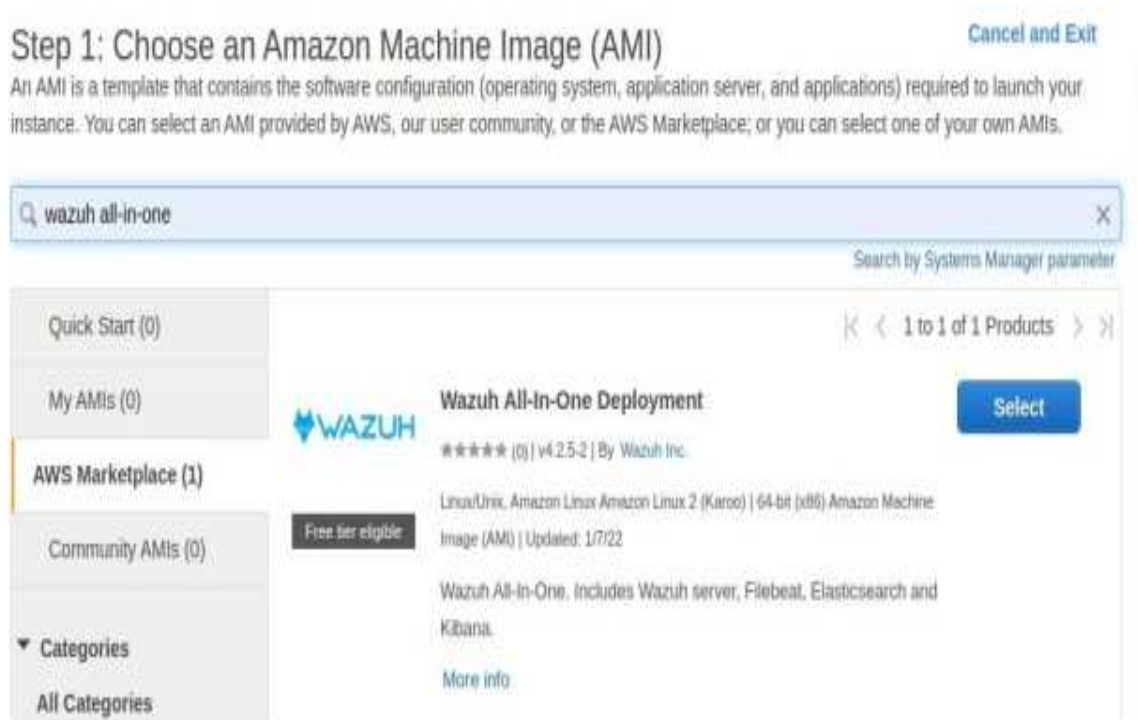


Figure 2.6: Wazuh Amazon AMI in AWS Marketplace

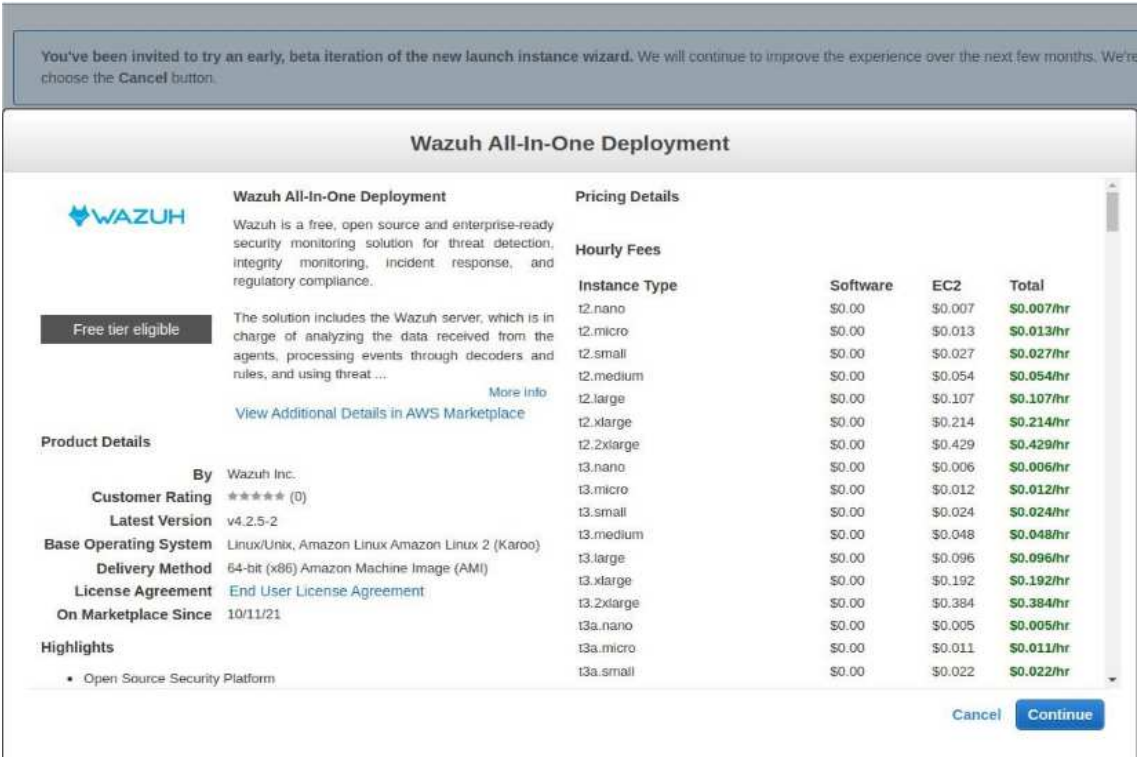


Figure 2.7: Wazuh Amazon AMI prices

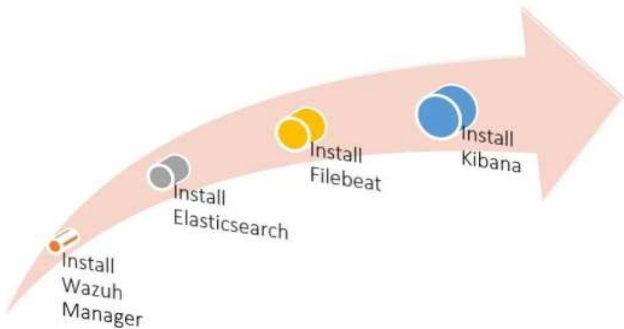


Figure 2.8: Component required to Install Wazuh server all-in-one

Conclusion

Through this chapter, we went through a deep dive into all the granular concepts that we will need throughout this project. We have highlighted the architecture of Kubernetes ,as well as the architecture of Wazuh and the Multi-tier deployed application.

Chapter 3

Installation and configuration of the implementation

Introduction

After understanding the components of all the architecture and the main role of each element, it's time to start the installation of all the component to build a consistent sample of which we can perform our attacks later. First, let's start with building kubernetes cluster on AWS .Second, let's deploy a sample application on Amazon EKS .Last but not least, let's install our Wazuh server from scratch and connect it to our kubernetes targeted worker node.

3.1 Building Amazon EKS with AWS CloudFormation (IaaS) with eksctl tool

3.1.1 What is eksctl

eksctl is an open-source CLI utility for building and managing clusters using EKS. It was built by Weaverworks in Go programming language .It leverages CloudFormation to build the required cluster.

3.1.2 Installing eksctl

We are working on a Kali distribution :



```
(geekone@geekone)-[~]  
$ uname -a  
Linux geekone 5.14.0-kali4-amd64 #1 SMP Debian 5.14.16-1kali1 (2021-11-05) x86_64 GNU/Linux
```

Figure 3.1: Find-out Kali Distribution

This is why we are going to use the dedicated installation guide for our distribution in AWS Documentation entitled '**Installing or upgrading eksctl on Linux using curl**'

The installation of the eksctl will happen just like the following steps:

```
(geekone@geekone)-[~]
$ curl -silent -location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

Figure 3.2: 1. Download and extract the latest release of eksctl with the following command

```
(geekone@geekone)-[~]
$ sudo mv /tmp/eksctl /usr/local/bin
```

Figure 3.3: 2. Move the extracted binary to /usr/local/bin

```
(geekone@geekone)-[~]
$ eksctl version
0.86.0
```

Figure 3.4: 3. Test that your installation was successful with the version attribute

3.1.3 Create IAM user account in AWS

Creating an IAM user account will be doing by following steps:



Figure 3.5: 1. From AWS services Menu , select IAM

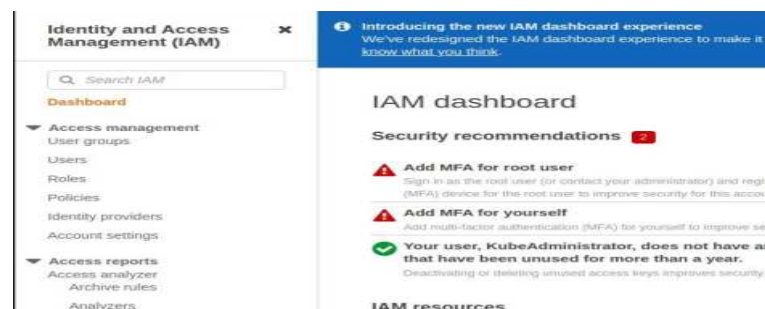


Figure 3.6: 2. Select Users and then add users

The screenshot shows the 'Add user' console in AWS IAM, specifically step 3: 'Set user details'. At the top, there are four numbered steps: 1 (Add user), 2 (Set user details), 3 (Set permissions), and 4 (Add tags). The 'User name' field is filled with 'KubeAdministrator'. Below it, there's a link to 'Add another user'. The 'Select AWS access type' section has two options: 'Access key - Programmatic access' (unchecked) and 'Password - AWS Management Console access' (checked). The 'Console password' section has two options: 'Autogenerated password' (unchecked) and 'Custom password' (checked). A password field is visible with masked characters. The 'Require password reset' checkbox is checked. At the bottom, there are 'Cancel' and 'Next: Permissions' buttons.

Figure 3.7: 3. Give a User name for your IAM User account and select the AWS access type

The screenshot shows the 'Set permissions' console in AWS IAM, specifically step 4: 'Set permissions'. At the top, there are three options: 'Add user to group' (selected), 'Copy permissions from existing user', and 'Attach existing policies directly'. Below these, there's a section 'Add user to group' with a search bar and a table of groups. The 'admins' group is selected, and its attached policy 'AdministratorAccess' is listed. At the bottom, there's a link to 'Set permissions boundary'.

Figure 3.8: 4.Set the permissions for your account , in our case, KubeAdministrator will belong to admins group and gain their attached policies

The screenshot shows the 'Add tags (optional)' console in AWS IAM, specifically step 5: 'Add tags'. It explains that IAM tags are key-value pairs that can be added to a user. Below this, there's a table with two columns: 'Key' and 'Value (optional)'. The first row has 'Kubernetes' as the key and 'kubernetes Cluster' as the value. There's a link to 'Add new key' and a note at the bottom stating 'You can add 49 more tags.'

Figure 3.9: 5. Add Tags , this step is optional. But is useful for root account to differentiate the other users and to know their purpose of creation.

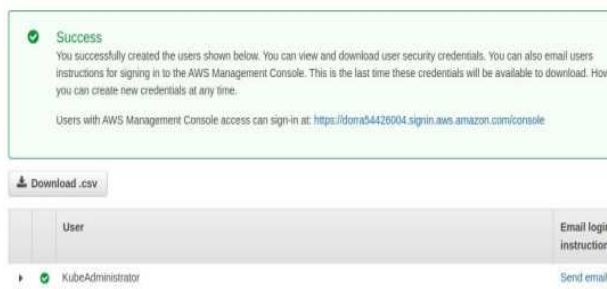


Figure 3.10: 6. Download the .csv file that contains the Credentials related to the user: User name and Login link

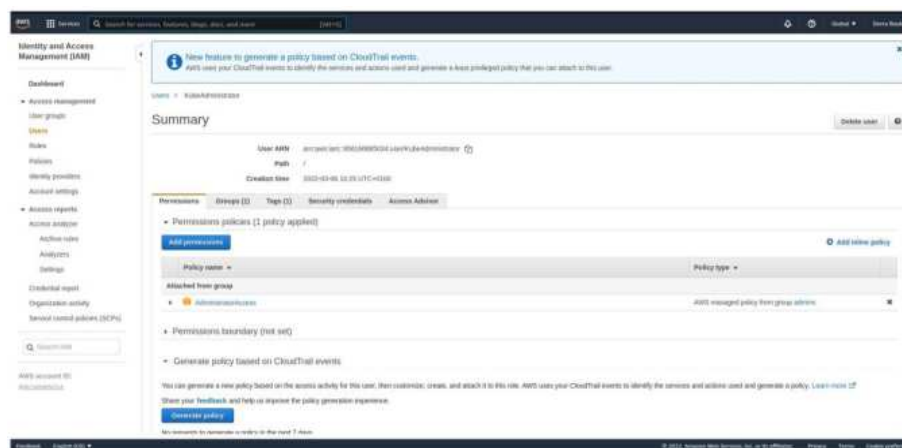


Figure 3.11: 7. You can see clearly that the new IAM user account is created successfully.

3.1.4 Connect eksctl to AWS IAM user Account

Before using eksctl, you have to get your aws IAM user credentials (created in the previous section) locally in the path `/.aws`. You can use this <https://docs.aws.amazon.com/cli/latest/userguide/configure-files.html> for more details.

So let's Integrate our IAM account credentials by applying these following steps:

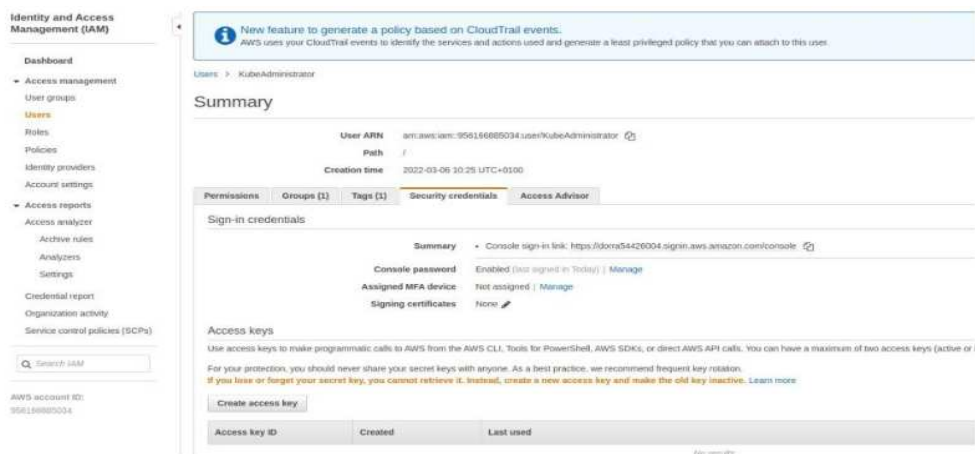


Figure 3.12: 1. To get the necessary access credentials, go to Security credentials, then lick on create access key

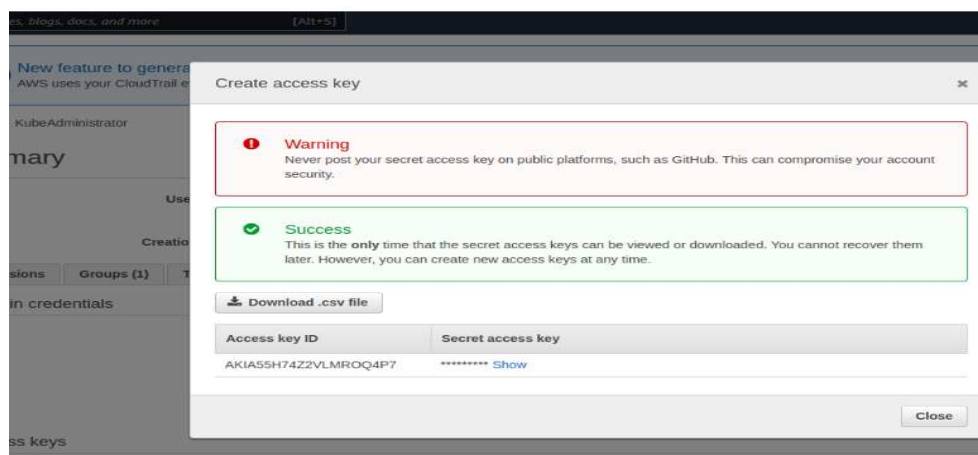


Figure 3.13: 2. At this point, download the .csv file that contain your Access key ID and your Secret access key.

PS: Make sure to keep this file out of reach, encrypted and never post its content on social media.

```
(geekone@geekone)-[~]
$ aws configure
AWS Access Key ID [None]: AKIA55H74Z2VLMROQ4P7
AWS Secret Access Key [None]: 4rqEPGdpPkbC1YPxXZN23fGStSerquijFgPri3z7d
Default region name [None]: us-east-1
Default output format [None]:
```

Figure 3.14: 3. Connect your IAM account to your local machine.

```
(geekone@geekone)-[~]
$ aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "KubeAdministrator",
      "UserId": "AIDA55H74Z2VC54VKCXIB",
      "Arn": "arn:aws:iam::956166885034:user/KubeAdministrator",
      "CreateDate": "2022-03-06T09:25:08Z",
      "PasswordLastUsed": "2022-03-06T09:52:00Z"
    }
  ]
}
```

Figure 3.15: 4. Test the connection by viewing the list of IAM users

```
(geekone@geekone)-[~]
$ vim .aws/credentials
```

Figure 3.16: You can see the account credentials in the following file : `.aws/credentials`

```
[default]
aws_access_key_id = 
aws_secret_access_key =
```

Figure 3.17: What The Credentials file will look like

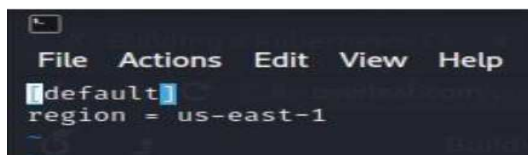


Figure 3.18: The region in which we will deploy our architecture is defined in the file `/.aws/config`

3.1.5 Installing the Cluster

Let's start with exploring the Amazon EKS service on the graphical console management interface. To do so, go to the research bar and search for eks .Select 'EKS Kubernetes Service' that appears among the research results.

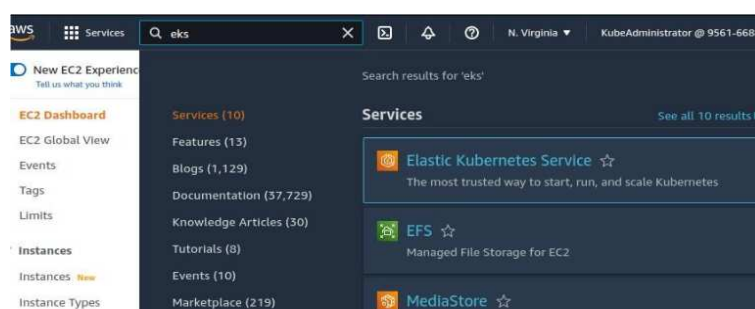


Figure 3.19: Searching for Amazon EKS on AWS Management Console

The service dashboard appears in front of you with a brief explanation of the service and a side navigation bar:



Figure 3.20: Amazon EKS Service Dashboard

In the side navigation bar, click on clusters to see the available eks clusters. If you don't have any deployed clusters on your account , you will see your AWS management console EKS service Dashboard like this:

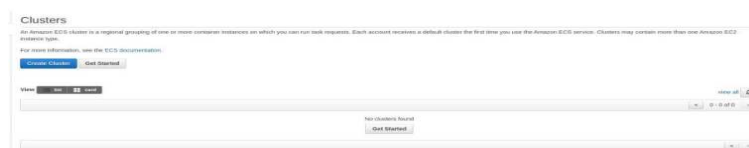


Figure 3.21: Available Amazon EKS Clusters

Now it's time to build our cluster with `eksctl` tool from weaveworks . Our cluster will contain:

- **Name of the cluster:** dorra-eks-cluster
- **Region where the cluster will be deployed:** us-east-1 (Virginia)
- **Kubernetes version:** 1.21
- **Number of worker nodes :** 3 worker nodes Managed by The customer but the Control Node Managed by AWS and is in an independent VPC.
- **Type of used EC2 instances:** t2.xlarge
- **Name of node group:** dorra-nodes

PS: Those are the value we decided to give after troubleshooting while installing the cluster



```

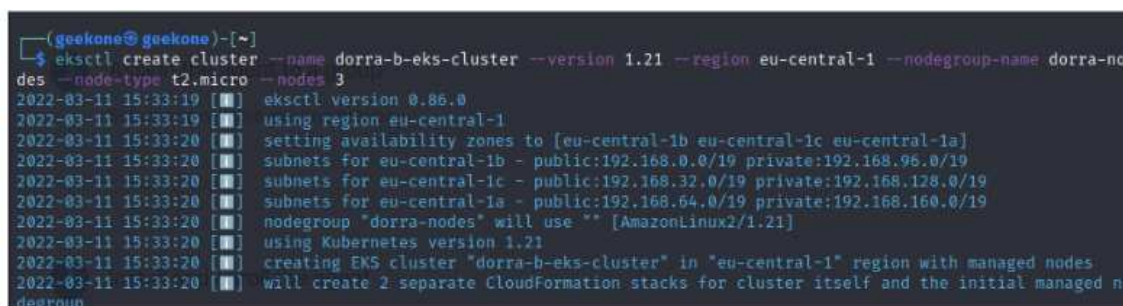
$ eksctl --help
The official CLI for Amazon EKS

Usage: eksctl [command] [flags]

Commands:
  eksctl anywhere           EKS anywhere
  eksctl associate          Associate resources with a cluster
  eksctl completion         Generates shell completion scripts for bash, zsh or fish
  eksctl create             Create resource(s)
  eksctl delete            Delete resource(s)
  eksctl deregister         Deregister a non-EKS cluster
  eksctl disassociate       Disassociate resources from a cluster
  eksctl drain             Drain resource(s)
  eksctl enable            Enable features in a cluster
  eksctl get               Get resource(s)
  eksctl help              Help about any command
  eksctl info              Output the version of eksctl, kubectl and OS info
  eksctl register          Register a non-EKS cluster
  eksctl scale             Scale resources(s)
  eksctl set               Set values
  eksctl unset            Unset values
  eksctl update            Update resource(s)
  eksctl upgrade           Upgrade resource(s)
  eksctl utils             Various utils
  eksctl version           Output the version of eksctl

```

Figure 3.22: 1. Explore the options given by the eksctl tool. We can see clearly that we can use eksctl create ***



```

$ eksctl create cluster --name dorra-b-eks-cluster --version 1.21 --region eu-central-1 --nodegroup-name dorra-no
des --node-type t2.micro --nodes 3
2022-03-11 15:33:19 [I] eksctl version 0.86.0
2022-03-11 15:33:19 [I] using region eu-central-1
2022-03-11 15:33:20 [I] setting availability zones to [eu-central-1b eu-central-1c eu-central-1a]
2022-03-11 15:33:20 [I] subnets for eu-central-1b - public:192.168.0.0/19 private:192.168.96.0/19
2022-03-11 15:33:20 [I] subnets for eu-central-1c - public:192.168.32.0/19 private:192.168.128.0/19
2022-03-11 15:33:20 [I] subnets for eu-central-1a - public:192.168.64.0/19 private:192.168.160.0/19
2022-03-11 15:33:20 [I] nodegroup "dorra-nodes" will use "" [AmazonLinux2/1.21]
2022-03-11 15:33:20 [I] using Kubernetes version 1.21
2022-03-11 15:33:20 [I] creating EKS cluster "dorra-b-eks-cluster" in "eu-central-1" region with managed nodes
2022-03-11 15:33:20 [I] will create 2 separate CloudFormation stacks for cluster itself and the initial managed n
odegroup

```

Figure 3.23: 2. Integrate the wanted parameters into the cluster creation command just as following

Once you execute this command, the kubernetes cluster will be created by using an IaaS (Infrastructure as a Code) service provided by AWS called CloudFormation. This operation take between 14-20 minutes to build a healthy cluster of 3 worker nodes.

```

2022-03-06 15:09:26 [ ] building cluster stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:09:27 [ ] deploying stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:09:57 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:10:29 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:11:30 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:12:47 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:13:52 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:14:56 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:16:02 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:17:03 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:18:10 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:19:19 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"
2022-03-06 15:20:40 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-cluster"

```

Figure 3.24: CloudFormation Starts building the Amazon EKS cluster

```

2022-03-06 15:27:02 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-nodegroup-dorra-nodes"
2022-03-06 15:27:32 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-nodegroup-dorra-nodes"
2022-03-06 15:27:53 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-nodegroup-dorra-nodes"
2022-03-06 15:28:13 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-nodegroup-dorra-nodes"
2022-03-06 15:28:45 [ ] waiting for CloudFormation stack "eksctl-dorra-eks-cluster-nodegroup-dorra-nodes"
2022-03-06 15:28:52 [ ] waiting for the control plane availability...
2022-03-06 15:28:52 [✓] saved kubeconfig as "/home/geekone/.kube/config"
2022-03-06 15:28:52 [ ] no tasks
2022-03-06 15:28:52 [✓] all EKS cluster resources for "dorra-eks-cluster" have been created
Error: listing nodes: Get "https://CD3DEE2EC999C92172D3279D21F14990.gr7.us-east-1.eks.amazonaws.com/api/v1/nodes?labelSelector=alpha.eksctl.io%2Fnodegroup-name%3Ddorra-nodes": http2: client connection lost

```

Figure 3.25: Here you can see clearly that the deployment succeeded

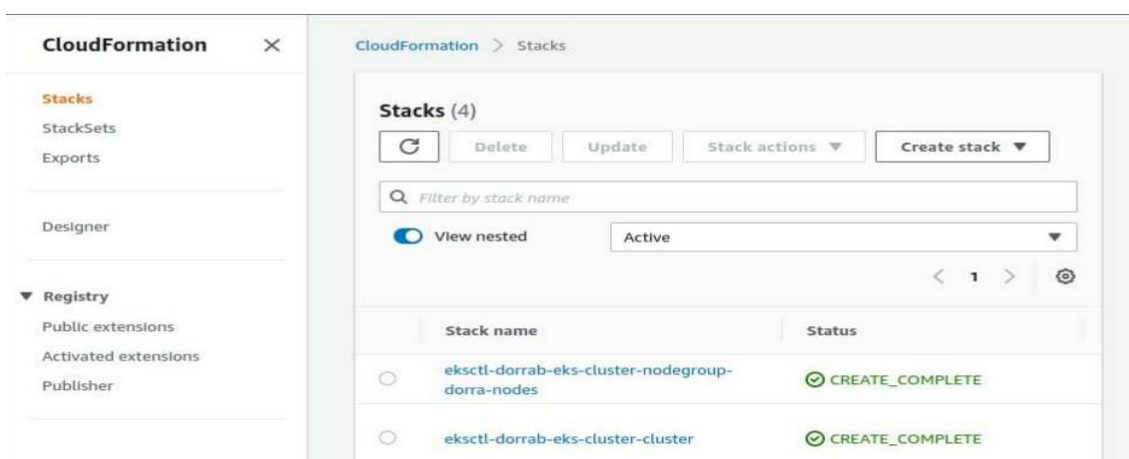


Figure 3.26: You can view the progress of your cluster with AWS CloudFormation

If you don't have **kubectl CLI** installed on your host, you can use the command 'sudo apt-get install kubernetes-client -y' to get in.

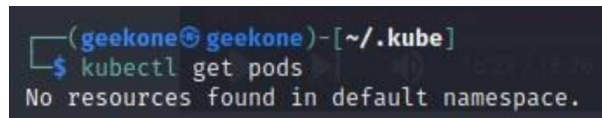
```

[geekone@geekone] ~/ .kube
$ kubectl get pods
Command 'kubectl' not found, but can be installed with:
sudo apt install kubernetes-client
Do you want to install it? (N/y)y
sudo apt install kubernetes-client
[sudo] password for geekone:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libvpx6 python3-twisted-bin ruby-atomic
  ruby-thread-safe
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  kubernetes-client
0 upgraded, 1 newly installed, 0 to remove and 132 not upgraded.
Need to get 8,382 kB of archives.
After this operation, 41.4 MB of additional disk space will be used.
0% [Connecting to mirror.serverius.net]

```

Figure 3.27: Install kubectl

Now , let's see what are the pod that we have on the cluster. Our cluster was just created so we should not expect any pods:



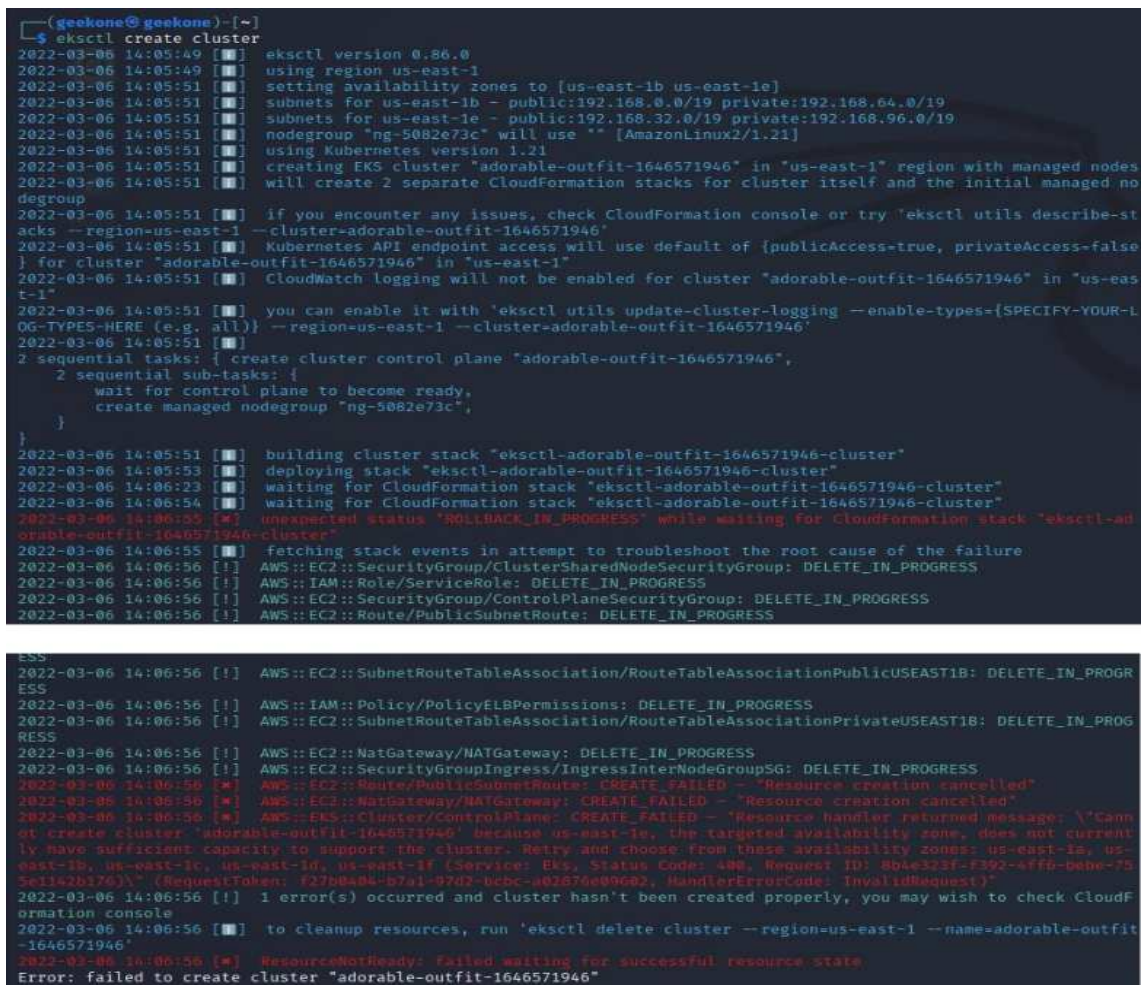
```
(geekone@geekone)~[~/.kube]
$ kubectl get pods
No resources found in default namespace.
```

Figure 3.28: Inspect Available pods on the Amazon EKS Clusters

3.1.6 Some Troubleshooting:

In this paragraph ,we will highlight some issue that you can face and how to solve them .

- **Unexpected status 'ROLLBACK_IN_PROGRESS' while waiting for CloudFormation stack.... ,CREATE_FAILED: Ressource Creation Cancelled :** This error means that there is something that stops cloudformation from building the cluster ressources. One main and obvious reason could be the insufficient permissions given to the IAM user who is currently building the cluster. To build an Amazon EKS cluster, the user must have permission to use CloudFormation as well as EC2 instance, along with other services .



```
(geekone@geekone)~[~]
$ eksctl create cluster
2022-03-06 14:05:49 [i] eksctl version 0.86.0
2022-03-06 14:05:49 [i] using region us-east-1
2022-03-06 14:05:51 [i] setting availability zones to [us-east-1b us-east-1e]
2022-03-06 14:05:51 [i] subnets for us-east-1b - public:192.168.0.0/19 private:192.168.64.0/19
2022-03-06 14:05:51 [i] subnets for us-east-1e - public:192.168.32.0/19 private:192.168.96.0/19
2022-03-06 14:05:51 [i] nodegroup "ng-5082e73c" will use "" [AmazonLinux2/1.21]
2022-03-06 14:05:51 [i] using Kubernetes version 1.21
2022-03-06 14:05:51 [i] creating EKS cluster "adorable-outfit-1646571946" in "us-east-1" region with managed nodes
2022-03-06 14:05:51 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2022-03-06 14:05:51 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region-us-east-1 --cluster-adorable-outfit-1646571946'
2022-03-06 14:05:51 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "adorable-outfit-1646571946" in "us-east-1"
2022-03-06 14:05:51 [i] CloudWatch logging will not be enabled for cluster "adorable-outfit-1646571946" in "us-east-1"
2022-03-06 14:05:51 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region-us-east-1 --cluster-adorable-outfit-1646571946'
2022-03-06 14:05:51 [i]
2 sequential tasks: { create cluster control plane "adorable-outfit-1646571946",
2 sequential sub-tasks: {
wait for control plane to become ready,
create managed nodegroup "ng-5082e73c",
}
}
2022-03-06 14:05:51 [i] building cluster stack "eksctl-adorable-outfit-1646571946-cluster"
2022-03-06 14:05:53 [i] deploying stack "eksctl-adorable-outfit-1646571946-cluster"
2022-03-06 14:06:23 [i] waiting for CloudFormation stack "eksctl-adorable-outfit-1646571946-cluster"
2022-03-06 14:06:54 [i] waiting for CloudFormation stack "eksctl-adorable-outfit-1646571946-cluster"
2022-03-06 14:06:55 [!] unexpected status "ROLLBACK_IN_PROGRESS" while waiting for CloudFormation stack "eksctl-adorable-outfit-1646571946-cluster"
2022-03-06 14:06:55 [i] fetching stack events in attempt to troubleshoot the root cause of the failure
2022-03-06 14:06:56 [!] AWS::EC2::SecurityGroup/ClusterSharedNodeSecurityGroup: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::IAM::Role/ServiceRole: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::SecurityGroup/ControlPlaneSecurityGroup: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::Route/PublicSubnetRoute: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::SubnetRouteTableAssociation/RouteTableAssociationPublicUSEAST1B: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::IAM::Policy/PolicyELBPermissions: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::SubnetRouteTableAssociation/RouteTableAssociationPrivateUSEAST1B: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::NatGateway/NATGateway: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::SecurityGroupIngress/IngressInterNodeGroupSG: DELETE_IN_PROGRESS
2022-03-06 14:06:56 [!] AWS::EC2::Route/PublicSubnetRoute: CREATE_FAILED - "Resource creation cancelled"
2022-03-06 14:06:56 [!] AWS::EC2::NatGateway/NATGateway: CREATE_FAILED - "Resource creation cancelled"
2022-03-06 14:06:56 [!] AWS::EKS::Cluster/ControlPlane: CREATE_FAILED - "Resource handler returned message: 'Cannot create cluster "adorable-outfit-1646571946" because us-east-1a, the targeted availability zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these availability zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f (Service: EKS, Status Code: 400, Request ID: 801e323f-f392-4ffb-b0be-755e11621761)' (RequestToken: f270a884-b7d1-9702-bbcb-402674e00602, HandlerErrorCode: InvalidRequest)"
2022-03-06 14:06:56 [!] 1 error(s) occurred and cluster hasn't been created properly, you may wish to check CloudFormation console
2022-03-06 14:06:56 [i] to cleanup resources, run 'eksctl delete cluster --region-us-east-1 --name-adorable-outfit-1646571946'
2022-03-06 14:06:56 [!] ResourceNotReady: Failed waiting for successful resource state
Error: failed to create cluster "adorable-outfit-1646571946"
```

Figure 3.29: Troubleshooting 'CREATE_FAILED'

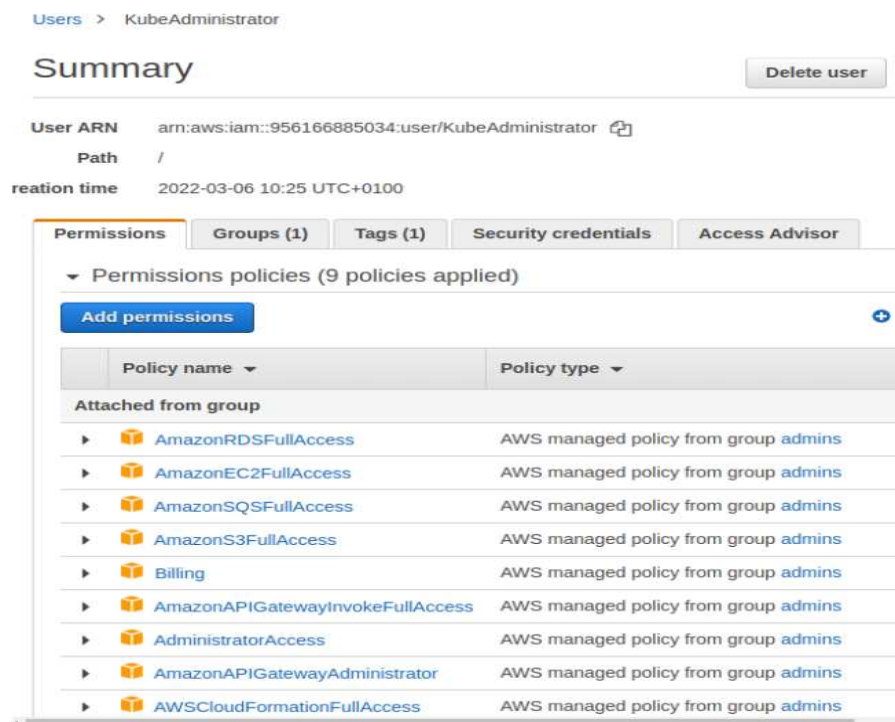


Figure 3.30: Some required IAM permissions for the user who is building the Amazon EKS with eksctl

- **'Error: invalid version '**: the main issue here is that the Amazon EKS kubernetes version required in the command is no longer available (deprecated). AWS works continuously on evaluating their projects , this is why old versions with limited features are sometimes unavailable.

```
(geekone@geekone)~$ eksctl create cluster --name dorra-eks-cluster --version 1.17 --region us-east-1 --nodegroup-name dorra-nodes --
node-type t2.xlarge --nodes 3
2022-03-06 15:06:38 [I] eksctl version 0.86.0
2022-03-06 15:06:38 [I] using region us-east-1
Error: invalid version, 1.17 is no longer supported, supported values: 1.18, 1.19, 1.20, 1.21
see also: https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html
```

Figure 3.31: Troubleshooting 'Error: invalid version '

The solution for this issue is to check continuously the available versions of Amazon EKS Kubernetes and to select the currently available ones.

For example on 24th April,2022 the available versions are the following ones:

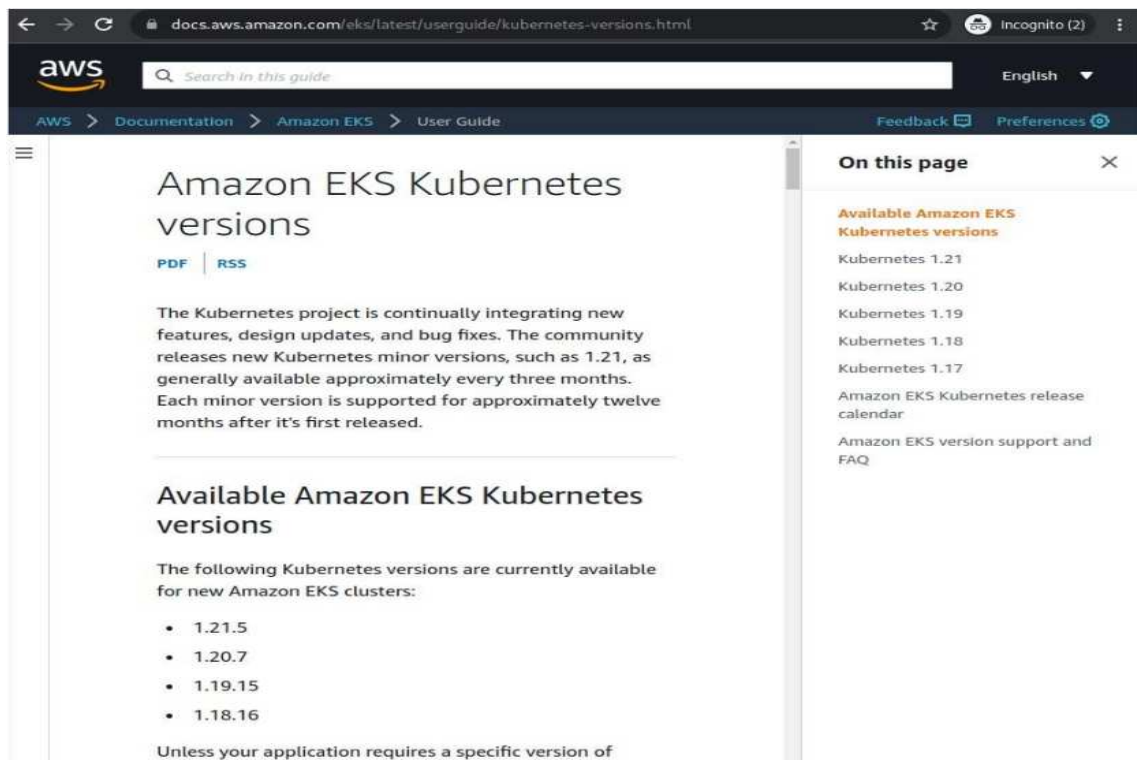


Figure 3.32: Available Amazon EKS Kubernetes versions[4]

After deleting an Amazon EKS created cluster named 'dorra-b-test' for example in the region 'eu-central-1' by using the command 'eksctl delete cluster', you will not be allowed to build another Amazon EKS cluster with the same name in the same region

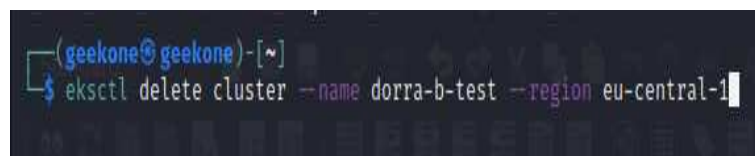


Figure 3.33: Deleting Amazon EKS cluster with one command

While building the next cluster, you will face an error, the ultimate solution is to modify the name of the next built cluster or to change the region itself.

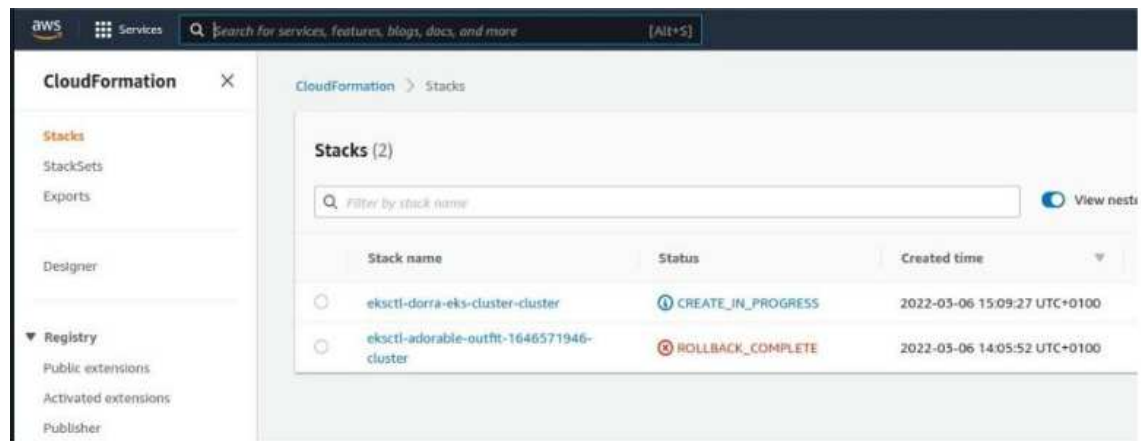


Figure 3.34: Troubleshooting: Unable to build an Amazon EKS Cluster with the same name of a previously deleted one

3.2 Deploying multi-tier application on Amazon EKS

Our multi-tier application will be deployed on Amazon EKS just like the following figure shows. We have a frontend Tier that is already in the publi subnet and which will be exposed to the EndUsers with a Node IP service, and we have 2 clustered database tiers which are in different Availability zones and in private subnets.

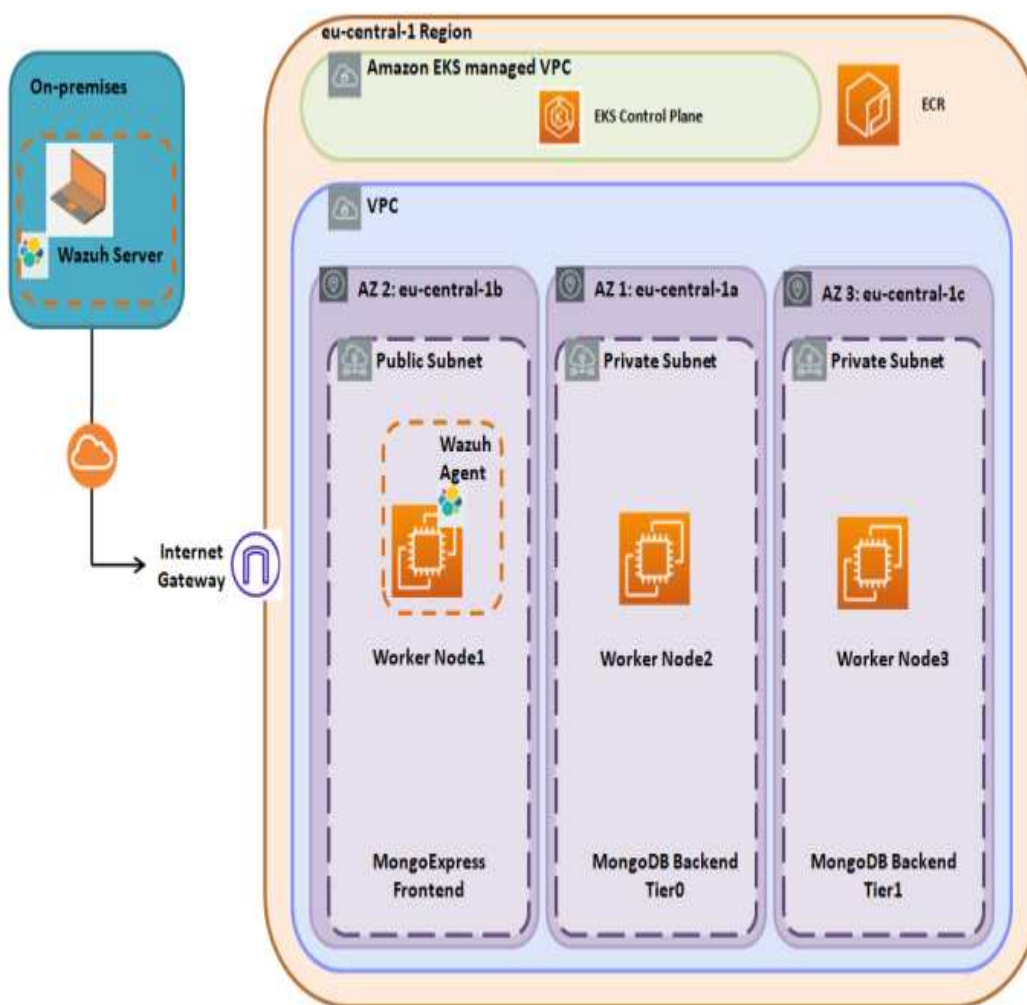


Figure 3.35: Implemented Architecture of the a Hybrid environment

- **Step1:** Create the MongoDB deployment or pod.
 - **Step2:** Create an Internal service (clusterIP service) for it in order to allow only pods from the same cluster to access it . No external requests are allowed
 - **Step3:** Create MongoExpress Deployment by using MongoExpress.yaml definition file. MongoExpress needs the DB URL as well as the Credentials to login to the database. Those informations are stored in Environment Variables. We will define the Mongo database URL in ConfigMap and login credentials in Secret and reference both parameters in MongoExpress.yaml under Environment Variables.
 - **Step4:** Expose MongoExpress to the browser with a NodePort Service
- PS:** You can see the yaml scripts used to completely deploy this application in the Annex A. Those yaml scripts correspond to :
- MongoDB deployment
 - the internal service that lets MongoExpres Access MongoDB

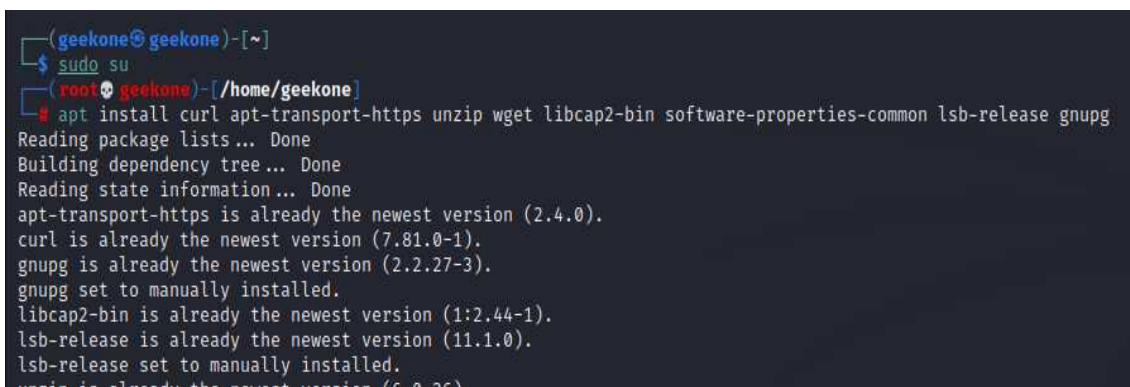
- MongoExpress deployment
- the external service that exposes FrontEnd Tier to the internet
- ConfigMap
- Secret

3.3 Installing Wazuh server and connect to the agent

To be able to supervise a specific element, we have first to install Wazuh server and deploy an agent on the supervised element.

3.3.1 Installing Wazuh Repository

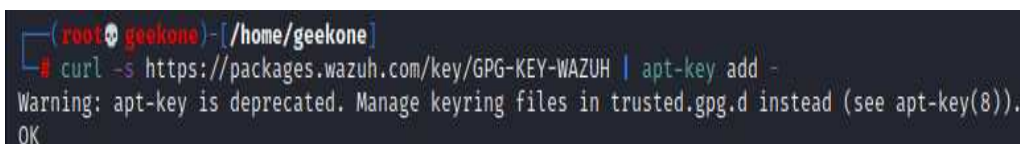
1. First, we have a set of required packages that have to be installed before going any further.



```
(geekone@geekone)-[~]
$ sudo su
(root@geekone)-[/home/geekone]
# apt install curl apt-transport-https unzip wget libcap2-bin software-properties-common lsb-release gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.4.0).
curl is already the newest version (7.81.0-1).
gnupg is already the newest version (2.2.27-3).
gnupg set to manually installed.
libcap2-bin is already the newest version (1:2.44-1).
lsb-release is already the newest version (11.1.0).
lsb-release set to manually installed.
unzip is already the newest version (6.0-26).
```

Figure 3.36: 1. Install the necessary packages for the installation:

2. Install the GPG key:

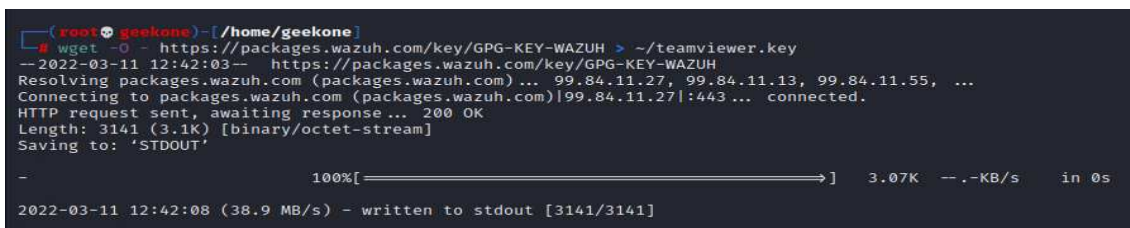


```
(root@geekone)-[/home/geekone]
# curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

Figure 3.37: 2. apt-key is deprecated

You can see that the command apt-key is deprecated.

3. So , let's install the GPG key while using wget command



```
(root@geekone)-[/home/geekone]
# wget -O - https://packages.wazuh.com/key/GPG-KEY-WAZUH > ~/teamviewer.key
--2022-03-11 12:42:03-- https://packages.wazuh.com/key/GPG-KEY-WAZUH
Resolving packages.wazuh.com (packages.wazuh.com)... 99.84.11.27, 99.84.11.13, 99.84.11.55, ...
Connecting to packages.wazuh.com (packages.wazuh.com)|99.84.11.27|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3141 (3.1K) [binary/octet-stream]
Saving to: 'STDOUT'

- 100%[=====>] 3.07K --.-KB/s in 0s
2022-03-11 12:42:08 (38.9 MB/s) - written to stdout [3141/3141]
```

Figure 3.38: 3. Install the GPG key with wget command

4. Now let's inspect the downloaded file and make sure it's a public key

```
(root@geekone)-[/home/geekone]
# file ~/teamviewer.key
/root/teamviewer.key: PGP public key block Public-Key (old)
```

Figure 3.39: 4. Inspect the downloaded

In our context, we need to get the GPG key Wazuh, but the downloaded one is a PGP. GnuPrivacy Guard (GPG) encrypts files securely so that only the intended receiver may decode them. GPG, in particular, adheres to the OpenPGP standard. It is based on the Pretty Good Privacy software (PGP).

PS: The `--no-default-keyring` means do not add the default keyrings to the list of keyrings.

```
(root@geekone)-[/home/geekone]
# gpg --no-default-keyring --keyring ./teamviewer_keyring.gpg --import ~/teamviewer.key
gpg: directory '/root/.gnupg' created
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 96B3EE5F29111145: public key "Wazuh.com (Wazuh Signing Key) <support@wazuh.com>" imported
gpg: Total number processed: 1
gpg:         imported: 1
```

Figure 3.40: 5. Create New GPG Keyring

Now, you can display the list of files and folders in the current directory. You can see that the GPG keyring is created successfully and is already available among the list. To inspect the file type, you can use the command `'file teamviewer_keyring.gpg'`. Now let's move this key to the directory that contains the config related to apt trusted keys.

```
(root@geekone)-[/home/geekone]
# gpg --no-default-keyring --keyring ./teamviewer_keyring.gpg --export > ./teamviewer.gpg

(root@geekone)-[/home/geekone]
# ls
CTF      google-chrome-stable_current_amd64.deb  Pictures      teamviewer_keyring.gpg~
Desktop  mongodb-define.yaml                     Public        Templates
Documents mongodb-secret.yaml                     teamviewer.gpg  Videos
Downloads Music                                   teamviewer_keyring.gpg

(root@geekone)-[/home/geekone]
# sudo mv ./teamviewer.gpg /etc/apt/trusted.gpg.d/
```

Figure 3.41: 6. Verify and move GPG Keyring to trusted config directory

```
(root@geekone)-[/home/geekone]
# echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list
deb https://packages.wazuh.com/4.x/apt/ stable main
```

Figure 3.42: 7. Add the repository

Now, for a better functioning, you have to update your system by using the command: `'sudo apt-get update'`

3.3.2 Installing the Wazuh manager

```
(root@geekone)~[/home/geekone]
# apt-get install wazuh-manager
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libvpx6 libxatracker2 python3-twisted-bin
  ruby-atomic ruby-thread-safe
```

Figure 3.43: 1. Install the Wazuh manager package

```
(root@geekone)~[/home/geekone]
# systemctl daemon-reload

(root@geekone)~[/home/geekone]
# systemctl enable wazuh-manager.service
Synchronizing state of wazuh-manager.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable wazuh-manager
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-manager.service → /lib/systemd/system/wazuh-manager.service.

(root@geekone)~[/home/geekone]
# systemctl start wazuh-manager.service
```

Figure 3.44: 2. Enable and start the Wazuh manager service

```
(root@geekone)~[/home/geekone]
# systemctl status wazuh-manager.service
● wazuh-manager.service - Wazuh manager
   Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-03-11 12:55:49 CET; 58s ago
     Process: 56860 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
    Tasks: 168 (limit: 18738)
   Memory: 1.0G
      CPU: 36.829s
   CGroup: /system.slice/wazuh-manager.service
           └─56934 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             └─56973 /var/ossec/bin/wazuh-authd
               └─56986 /var/ossec/bin/wazuh-db
                 └─57014 /var/ossec/bin/wazuh-execd
                   └─57032 /var/ossec/bin/wazuh-analysisd
                     └─57133 /var/ossec/bin/wazuh-syscheckd
                       └─57159 /var/ossec/bin/wazuh-remoted
                         └─57194 /var/ossec/bin/wazuh-logcollector
                           └─57220 /var/ossec/bin/wazuh-monitord
                             └─57291 /var/ossec/bin/wazuh-modulesd

Mar 11 12:55:39 geekone env[56860]: Started wazuh-db ...
Mar 11 12:55:40 geekone env[56860]: Started wazuh-execd ...
Mar 11 12:55:41 geekone env[56860]: Started wazuh-analysisd ...
Mar 11 12:55:42 geekone env[56860]: Started wazuh-syscheckd ...
Mar 11 12:55:43 geekone env[56860]: Started wazuh-remoted ...
Mar 11 12:55:44 geekone env[56860]: Started wazuh-logcollector ...
Mar 11 12:55:46 geekone env[56860]: Started wazuh-monitord ...
Mar 11 12:55:47 geekone env[56860]: Started wazuh-modulesd ...
Mar 11 12:55:49 geekone env[56860]: Completed.
Mar 11 12:55:49 geekone systemd[1]: Started Wazuh manager.
```

Figure 3.45: Run this command to check if the Wazuh manager is active

3.3.3 Installing Elasticsearch

```
(root@geekone)~[/home/geekone]
# apt install elasticsearch-oss opendistroforelasticsearch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libvpx6 libxatracker2 python3-twisted-bin
  ruby-atomic ruby-thread-safe
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  opendistro-alerting opendistro-anomaly-detection opendistro-asynchronous-search opendistro-index-management
  opendistro-job-scheduler opendistro-knn opendistro-knnlib opendistro-performance-analyzer
  opendistro-reports-scheduler opendistro-security opendistro-sql
The following NEW packages will be installed:
  elasticsearch-oss opendistro-alerting opendistro-anomaly-detection opendistro-asynchronous-search
  opendistro-index-management opendistro-job-scheduler opendistro-knn opendistro-knnlib
  opendistro-performance-analyzer opendistro-reports-scheduler opendistro-security opendistro-sql
```

Figure 3.46: 1. Install Elasticsearch OSS and Open Distro for Elasticsearch

```
(root@geekone)~[/home/geekone]
# curl -sO /usr/share/elasticsearch/plugins/opendistro_security/securityconfig/roles.yml https://packages.wazuh.com/resources/4.2/open-distro/elasticsearch/roles/roles.yml

(root@geekone)~[/home/geekone]
# curl -sO /usr/share/elasticsearch/plugins/opendistro_security/securityconfig/roles_mapping.yml https://packages.wazuh.com/resources/4.2/open-distro/elasticsearch/roles/roles_mapping.yml

(root@geekone)~[/home/geekone]
# curl -sO /usr/share/elasticsearch/plugins/opendistro_security/securityconfig/internal_users.yml https://packages.wazuh.com/resources/4.2/open-distro/elasticsearch/roles/internal_users.yml
```

Figure 3.47: 2. Download the configuration file, Users and roles for Elasticsearch

3.3.4 Certificates Creation

Remove the demo certificates, Generate and deploy the certificates with `wazuh-cert-tool.sh` tool, and then move them to their proper directory:

```
(root@geekone)~[/home/geekone]
# rm /etc/elasticsearch/esnode-key.pem /etc/elasticsearch/esnode.pem /etc/elasticsearch/kirk-key.pem /etc/elasticsearch/kirk.pem /etc/elasticsearch/root-ca.pem -f

(root@geekone)~[/home/geekone]
# curl -sO ~/wazuh-cert-tool.sh https://packages.wazuh.com/resources/4.2/open-distro/tools/certificate-utility/wazuh-cert-tool.sh

(root@geekone)~[/home/geekone]
# curl -sO ~/instances.yml https://packages.wazuh.com/resources/4.2/open-distro/tools/certificate-utility/instances.yml

(root@geekone)~[/home/geekone]
# bash ~/wazuh-cert-tool.sh
03/11/2022 13:05:12 INFO: Configuration file found. Creating certificates ...
03/11/2022 13:05:12 INFO: Creating the Elasticsearch certificates ...
03/11/2022 13:05:12 INFO: Creating Wazuh server certificates ...
03/11/2022 13:05:12 INFO: Creating Kibana certificate ...
03/11/2022 13:05:12 INFO: Certificates creation finished. They can be found in ~/certs.

(root@geekone)~[/home/geekone]
# mkdir /etc/elasticsearch/certs/

(root@geekone)~[/home/geekone]
# mv ~/certs/elasticsearch* /etc/elasticsearch/certs/

(root@geekone)~[/home/geekone]
# mv ~/certs/admin* /etc/elasticsearch/certs/

(root@geekone)~[/home/geekone]
# cp ~/certs/root-ca* /etc/elasticsearch/certs/
```

Figure 3.48: 1. Manage certificates

Add the following configuration to mitigate Apache Log4j2 Remote Code Execution (RCE) vulnerability - CVE-2021-44228 - ESA-2021-31.

```
(root@geekone)~[/home/geekone]
# mkdir -p /etc/elasticsearch/jvm.options.d

(root@geekone)~[/home/geekone]
# echo '-Dlog4j2.formatMsgNoLookups=true' > /etc/elasticsearch/jvm.options.d/disabledlog4j.options

(root@geekone)~[/home/geekone]
# chmod 2750 /etc/elasticsearch/jvm.options.d/disabledlog4j.options

(root@geekone)~[/home/geekone]
# chown root:elasticsearch /etc/elasticsearch/jvm.options.d/disabledlog4j.options
```

Figure 3.49: 2.Mitigate Log4j2 vulnerability:

```
(root@geekone)~[/home/geekone]
# systemctl daemon-reload

(root@geekone)~[/home/geekone]
# systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /lib/systemd/system/elasticsearch.service.

(root@geekone)~[/home/geekone]
# systemctl start elasticsearch.service

(root@geekone)~[/home/geekone]
# systemctl status elasticsearch.service
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-03-11 13:09:37 CET; 20s ago
     Docs: https://www.elastic.co
    Main PID: 64324 (java)
      Tasks: 76 (limit: 18738)
     Memory: 1.2G
        CPU: 21.187s
    CGroup: /system.slice/elasticsearch.service
            └─64324 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.netw>
```

Figure 3.50: 3.Enable and start the Elasticsearch service

```
(root@geekone)~[/home/geekone]
# export JAVA_HOME=/usr/share/elasticsearch/jdk/ 66 /usr/share/elasticsearch/plugins/opendistro_security/tools/se
curityadmin.sh -cd /usr/share/elasticsearch/plugins/opendistro_security/securityconfig/ -nhnv -cacert /etc/elastics
earch/certs/root-ca.pem -cert /etc/elasticsearch/certs/admin.pem -key /etc/elasticsearch/certs/admin-key.pem
Open Distro Security Admin v7
Will connect to localhost:9300 ... done
Connected as CN=admin,OU=Docu,O=Wazuh,L=California,C=US
Elasticsearch Version: 7.10.2
Open Distro Security Version: 1.13.1.0
Contacting elasticsearch cluster 'elasticsearch' and wait for YELLOW clusterstate ...
Clustername: elasticsearch
Clusterstate: GREEN
Number of nodes: 1
Number of data nodes: 1
```

Figure 3.51: 4. Run the Elasticsearch securityadmin script to load the new certificates information and start the cluster

Test if the installation was successful with this command:

```
(root@geekone)~/home/geekone
# curl -XGET https://localhost:9200 -u admin:admin -k
{
  "name" : "node-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "WKcwCgL8QvKBQG0JnjPbfg",
  "version" : {
    "number" : "7.10.2",
    "build_flavor" : "oss",
    "build_type" : "deb",
    "build_hash" : "747e1cc71def077253878a59143c1f785afa92b9",
    "build_date" : "2021-01-13T00:42:12.435326Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 3.52: Test the installation

5. As recommended, we delete The Open Distro for Elasticsearch performance analyzer plugin. We don't need it in our context and it has a bad impact on the performance of our system resources.

```
(root@geekone)~/home/geekone
# /usr/share/elasticsearch/bin/elasticsearch-plugin remove opendistro-performance-analyzer
→ removing [opendistro-performance-analyzer] ...

(root@geekone)~/home/geekone
# systemctl restart elasticsearch.service
```

Figure 3.53: 5.delete The Open Distro for Elasticsearch performance analyzer plugin

3.3.5 Installing Filebeat

```
(root@geekone)~/home/geekone
# apt-get install filebeat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libvpx6 libxatracker2 python3-twisted-bin
  ruby-atomic ruby-thread-safe
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
```

Figure 3.54: 1. Install the Filebeat package

2. Download the preconfigured Filebeat configuration file used to forward the Wazuh alerts to Elasticsearch, alert template for Elasticsearch and then the Wazuh module for Filebeat


```

(root@geekone)~[/home/geekone]
# curl -sO /etc/filebeat/filebeat.yml https://packages.wazuh.com/resources/4.2/open-distro/filebeat/7.x/filebeat_all_in_one.yml

(root@geekone)~[/home/geekone]
# curl -sO /etc/filebeat/wazuh-template.json https://raw.githubusercontent.com/wazuh/wazuh/4.2/extensions/elastic-search/7.x/wazuh-template.json

(root@geekone)~[/home/geekone]
# chmod go+r /etc/filebeat/wazuh-template.json

(root@geekone)~[/home/geekone]
# curl -s https://packages.wazuh.com/4.x/filebeat/wazuh-filebeat-0.1.tar.gz | tar -xvz -C /usr/share/filebeat/module
wazuh/
wazuh/module.yml
wazuh/archives/
wazuh/archives/config/
wazuh/archives/config/archives.yml
wazuh/archives/ingest/
wazuh/archives/ingest/pipeline.json
wazuh/archives/manifest.yml
wazuh/alerts/
wazuh/alerts/config/
wazuh/alerts/config/alerts.yml
wazuh/alerts/ingest/
wazuh/alerts/ingest/pipeline.json
wazuh/alerts/manifest.yml
wazuh/_meta/
wazuh/_meta/config.yml
wazuh/_meta/fields.yml
wazuh/_meta/docs.asciidoc

```

Figure 3.55: 2. Download the preconfigured Filebeat configuration file

```

(root@geekone)~[/home/geekone]
# mkdir /etc/filebeat/certs

(root@geekone)~[/home/geekone]
# cp ~/certs/root-ca.pem /etc/filebeat/certs/

(root@geekone)~[/home/geekone]
#

(root@geekone)~[/home/geekone]
# mv ~/certs/filebeat* /etc/filebeat/certs/

```

Figure 3.56: Copy the Elasticsearch certificates into /etc/filebeat/certs

```

(root@geekone)~[/home/geekone]
# systemctl daemon-reload

(root@geekone)~[/home/geekone]
# systemctl enable filebeat
Synchronizing state of filebeat.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable filebeat
Created symlink /etc/systemd/system/multi-user.target.wants/filebeat.service → /lib/systemd/system/filebeat.service
.

(root@geekone)~[/home/geekone]
# systemctl start filebeat

(root@geekone)~[/home/geekone]
# systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
   Loaded: loaded (/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-03-11 13:22:36 CET; 8s ago
     Docs: https://www.elastic.co/products/beats/filebeat
    Main PID: 69351 (filebeat)
      Tasks: 13 (limit: 18738)
     Memory: 17.8M
        CPU: 61ms
    CGroup: /system.slice/filebeat.service
            └─69351 /usr/share/filebeat/bin/filebeat --environment systemd -c /etc/filebeat/filebeat.yml --path.h>

```

Figure 3.57: 3. Enable and start the Filebeat service


```
(root@geekone)~/home/geekone
# filebeat test output
elasticsearch: https://127.0.0.1:9200...
  parse url ... OK
  connection ...
    parse host ... OK
    dns lookup ... OK
    addresses: 127.0.0.1
    dial up ... OK
  TLS ...
    security: server's certificate chain verification is enabled
    handshake ... OK
    TLS version: TLSv1.3
    dial up ... OK
  talk to server ... OK
  version: 7.10.2
```

Figure 3.58: 4. Test if the installation of Filebeat was successful

3.3.6 Installing Kibana

```
(root@geekone)~/home/geekone
# apt-get install opendistroforelasticsearch-kibana
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libgdk-pixbuf-xlib-2.0-0 libgdk-pixbuf2.0-0 libvpx6 libxatracker2 python3-twisted-bin
  ruby-atomic ruby-thread-safe
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  opendistroforelasticsearch-kibana
0 upgraded, 1 newly installed, 0 to remove and 199 not upgraded.
Need to get 234 MB of archives.
After this operation, 692 MB of additional disk space will be used.
Get:1 https://packages.wazuh.com/4.x/apt/stable/main amd64 opendistroforelasticsearch-kibana amd64 1.13.2 [234 MB]
Fetched 234 MB in 6min 13s (629 kB/s)
Selecting previously unselected package opendistroforelasticsearch-kibana.
(Reading database ... 324568 files and directories currently installed.)
Preparing to unpack .../opendistroforelasticsearch-kibana_1.13.2_amd64.deb ...
Unpacking opendistroforelasticsearch-kibana (1.13.2) ...
Setting up opendistroforelasticsearch-kibana (1.13.2) ...
useradd warning: kibana's uid 138 outside of the UID_MIN 1000 and UID_MAX 60000 range.
chown: cannot access '/usr/share/kibana/optimize': No such file or directory
no optimize folder
```

Figure 3.59: 1. Install the Kibana package:

```
(root@geekone)~/home/geekone
# curl -sO /etc/kibana/kibana.yml https://packages.wazuh.com/resources/4.2/open-distro/kibana/7.x/kibana_all_in_one.yml

(root@geekone)~/home/geekone
# mkdir /usr/share/kibana/data

(root@geekone)~/home/geekone
# chown -R kibana:kibana /usr/share/kibana/data
```

Figure 3.60: 2. Download the Kibana configuration file and Create the /usr/share/kibana/data directory

```
(root@geekone)~[/home/geekone]
# cd /usr/share/kibana

(root@geekone)~[/usr/share/kibana]
# sudo -u kibana /usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/4.x/ui/kibana/wazuh_kibana-4.2.5_7.10.2-1.zip
Attempting to transfer from https://packages.wazuh.com/4.x/ui/kibana/wazuh_kibana-4.2.5_7.10.2-1.zip
Transferring 33111704 bytes.....
```

Figure 3.61: 3. Install the Wazuh Kibana plugin. The installation of the plugin must be done from the Kibana home directory

```
(root@geekone)~[/home/geekone]
# mkdir /etc/kibana/certs

(root@geekone)~[/home/geekone]
# cp ~/certs/root-ca.pem /etc/kibana/certs/

(root@geekone)~[/home/geekone]
# mv ~/certs/kibana* /etc/kibana/certs/

(root@geekone)~[/home/geekone]
# chown kibana:kibana /etc/kibana/certs/*
```

Figure 3.62: 4. Create the /etc/kibana/certs directory and ownership and Copy the Elasticsearch certificates into /etc/kibana/certs

```
(root@geekone)~[/home/geekone]
# setcap 'cap_net_bind_service=+ep' /usr/share/kibana/node/bin/node
```

Figure 3.63: 5. Link Kibana socket to privileged port 443

```
(root@geekone)~[/home/geekone]
# systemctl daemon-reload

(root@geekone)~[/home/geekone]
# systemctl enable kibana
Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable kibana
Created symlink /etc/systemd/system/multi-user.target.wants/kibana.service → /etc/systemd/system/kibana.service.

(root@geekone)~[/home/geekone]
# systemctl start kibana

(root@geekone)~[/home/geekone]
# systemctl status kibana.service
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-03-11 13:46:28 CET; 17s ago
     Main PID: 76657 (node)
       Tasks: 11 (limit: 18738)
      Memory: 174.2M
         CPU: 3.655s
    CGroup: /system.slice/kibana.service
            └─76657 /usr/share/kibana/bin/.. /node/bin/node /usr/share/kibana/bin/.. /src/cli/dist -c /etc/kibana/ki>

Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["info","savedobjec>
Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["info","savedobjec>
Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["info","plugins-sy>
Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["error","elasticsearch>
Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["error","elasticsearch>
Mar 11 13:46:31 geekone kibana[76657]: {"type":"log","@timestamp":"2022-03-11T12:46:31Z","tags":["error","elasticsearch>
```

Figure 3.64: 6. Enable and start the Kibana service

3.3.7 Secure Wazuh Access on the Browser

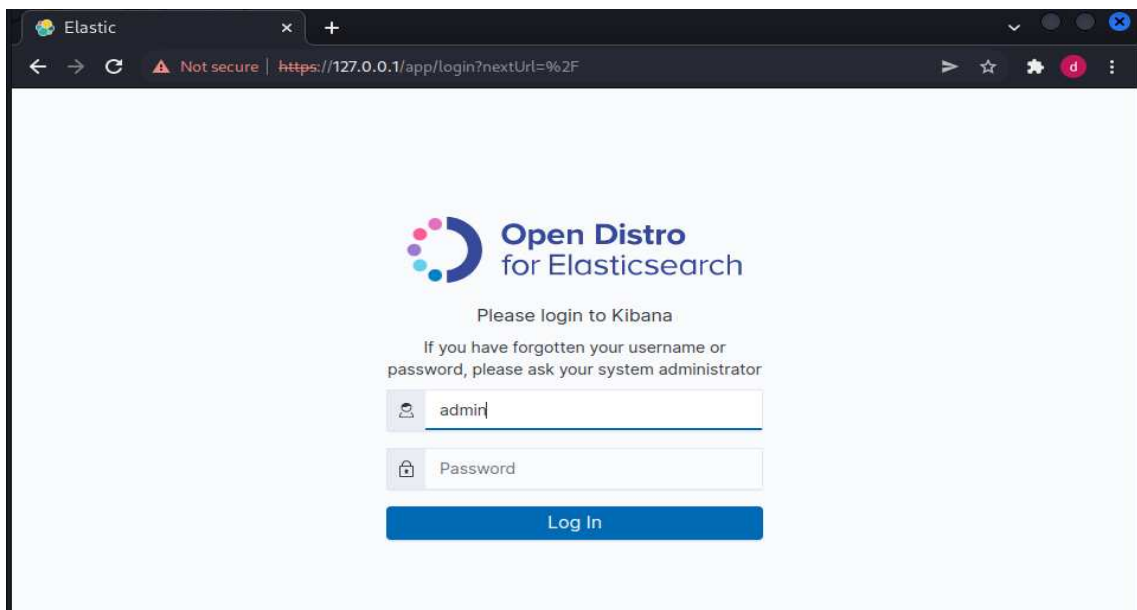


Figure 3.65: 1. Access the Wazuh login interface on the browser localhost

You can clearly see that the connection is still HTTP (Not HTTPS), thus not secure.

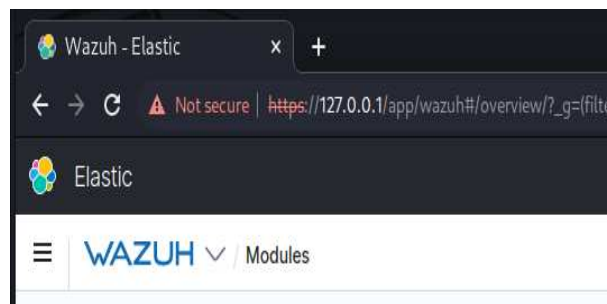


Figure 3.66: 2. Connection is not secure

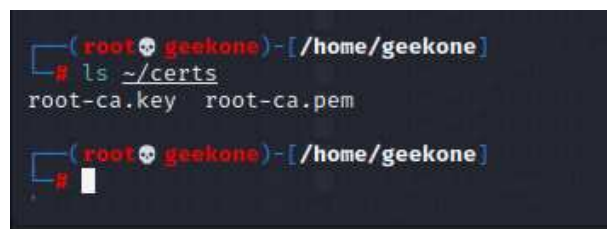


Figure 3.67: 3. Notice the Root Certification Authority certificate that we have previously downloaded

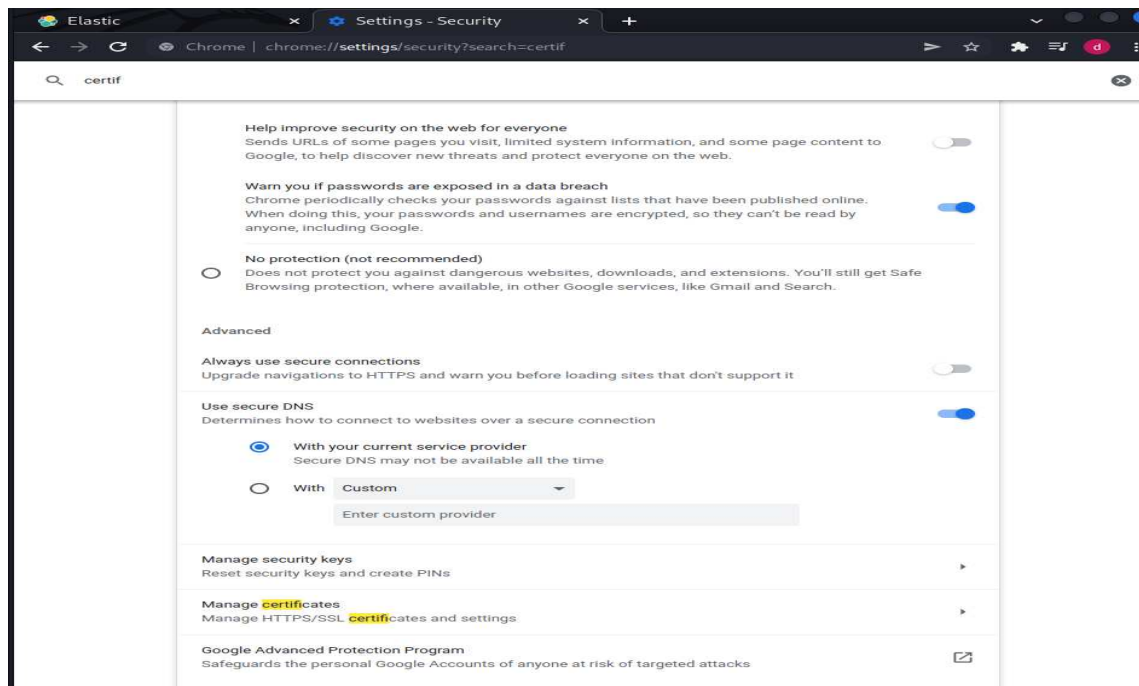


Figure 3.68: 4. Go to the Web Browser Settings (We are using Chrome for this project). Select Manage certificates:

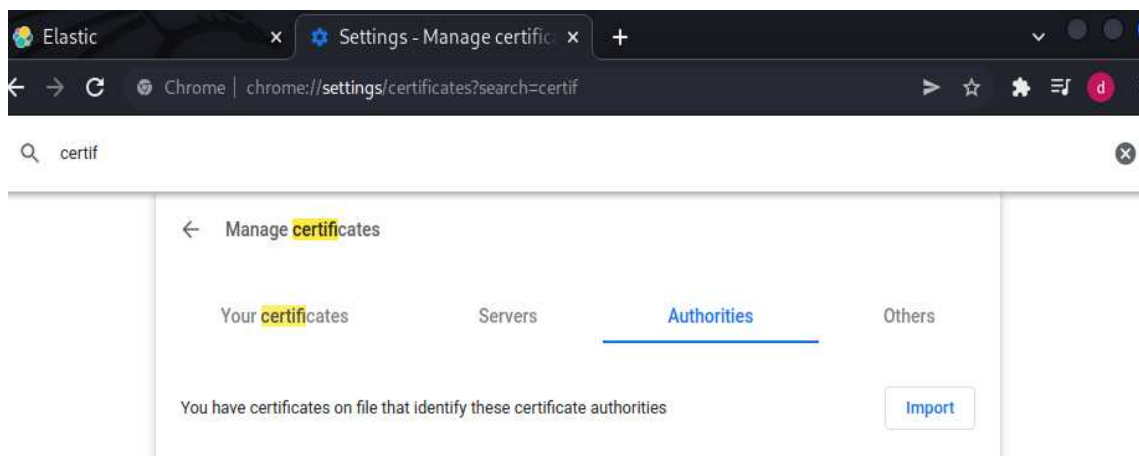


Figure 3.69: 5. Select Authorities section and then Import

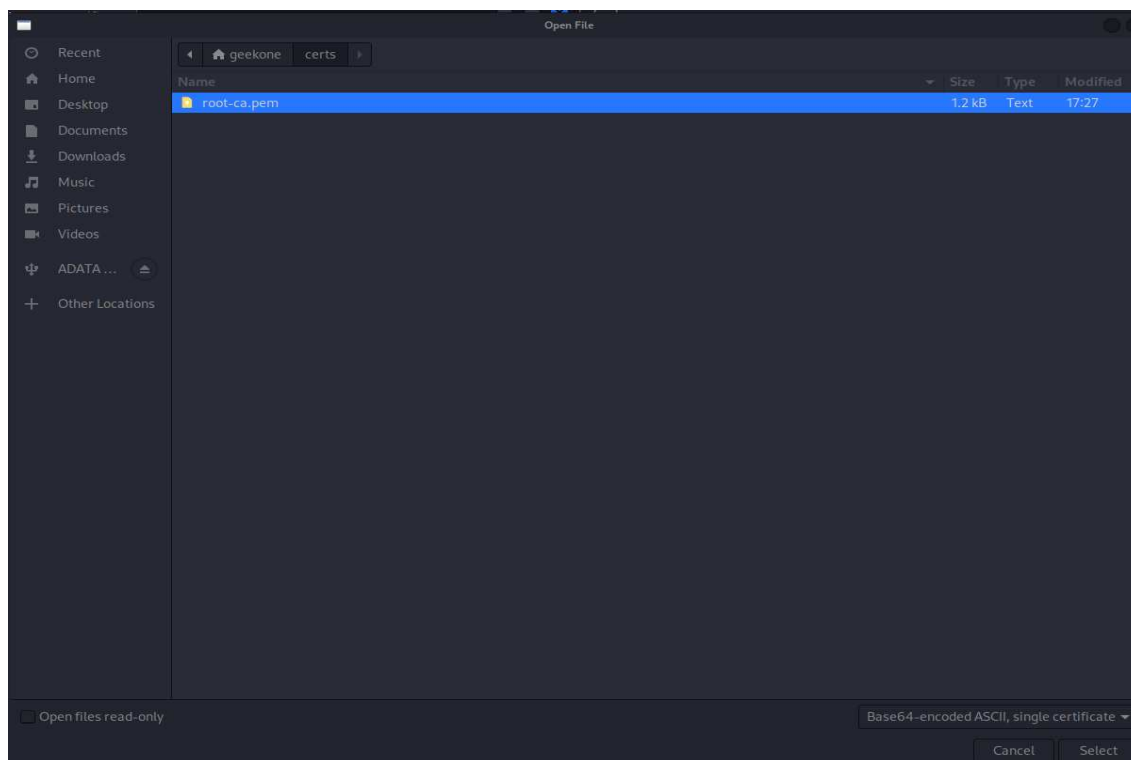


Figure 3.70: 6.Select the root-ca.pem certificate that we have already downloaded

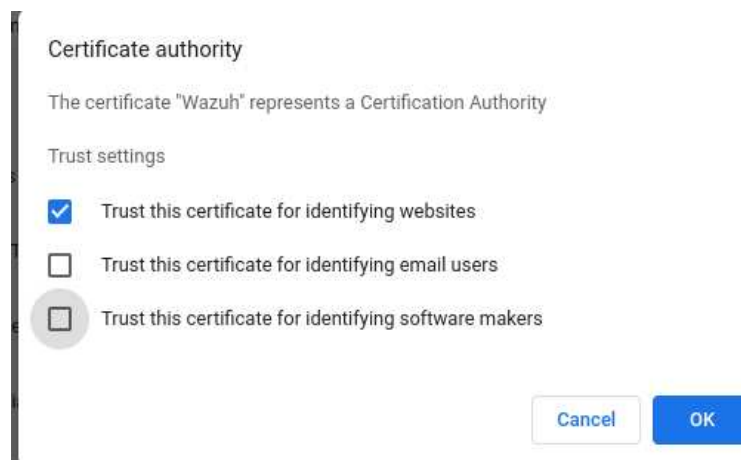


Figure 3.71: 7.Check 'Trust this certificate for identifying websites'

You can just to check 'Trust this certificate for identifying websites' because we don't need Wazuh to identify email users or software makers.

3.3.8 Connect Wazuh server To the agent

Since our aim is to supersize and monitor the Front-tier of our Multi-tier application, we need to establish a connection between the on-premises Wazuh server and the Wazuh agent. Our Wazuh server runs on the localhost `https://127.0.0.1` . And our Wazuh agent should be applied on the worker node that handles the frontend ,which is an EC2 instance that belongs to Amazon EKS Cluster on the cloud .

To establish this connection, we need to :

- First expose our local server online so that Amazon EKS instance can connect to the internet.
- Second expose the Frontend tier of our application so it can be reachable by Wazuh server

3.3.8.1 Expose our local Wazuh server to the internet

To make our Wazuh server accessible bu internet users ,we can use several methods .One of which is ngrok.

Ngrok is a cross-platform program that uses the Internet to expose local server ports. It provides a random public subdomain reachable on the internet.

So , to expose Wazuh server to the internet, first we have to install ngrok and use it to make the server public.

1.a Install Ngrok On Kali Linux: 1.a.1: Go to the official website of Ngrok <https://ngrok.com/doc> ,and make a FREE sub- scription to the website or you can use Google or GitHub account to sign-in. (Personally, I used my GitHub account to sign-in)

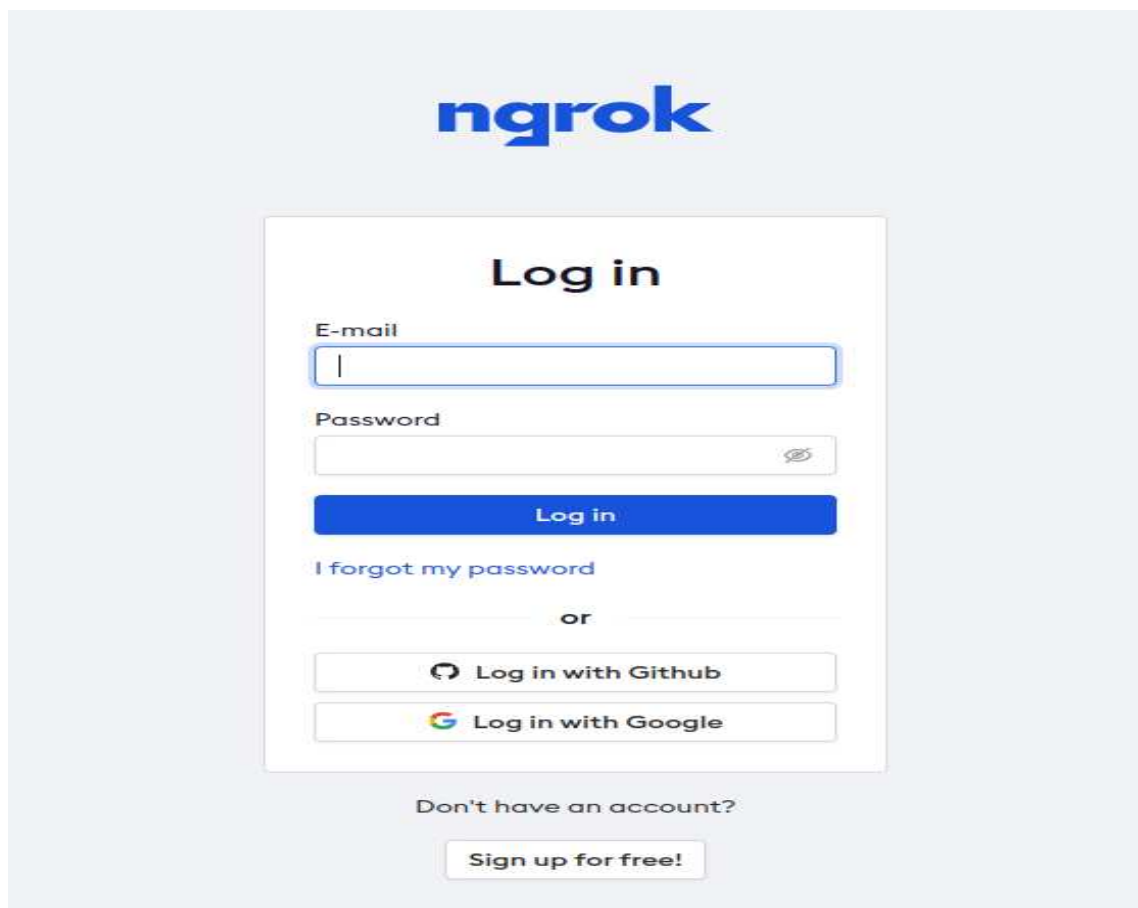


Figure 3.72: Sign-in to Ngrok

1.a.2 : Download Ngrok Installation compressed file for Kali Linux with this command


```
(geekone@geekone)~$ sudo wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
[sudo] password for geekone:
--2022-03-26 02:02:28-- https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
Resolving bin.equinox.io (bin.equinox.io)... 54.237.133.81, 18.205.222.128, 54.161.241.46, ...
Connecting to bin.equinox.io (bin.equinox.io)|54.237.133.81|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13770165 (13M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.tgz'

ngrok-stable-linux-amd64.tgz 100%[=====] 13.13M 1.26MB/s in 18s

2022-03-26 02:02:47 (751 KB/s) - 'ngrok-stable-linux-amd64.tgz' saved [13770165/13770165]
```

Figure 3.73: Download Ngrok Installation file for Kali linux

1.a.3 : Verify the installation of the Ngrok stable version compressed file on the current directory

```
(geekone@geekone)~$ ls
certs      Music      virt1.gpg
CTF        ngrok-stable-linux-amd64.tgz  virt1.key
Desktop    Pictures   virt1-keyring.gpg
Documents  Public     virt1-keyring.gpg-
Downloads  root-ca.pem virt.gpg
google-chrome-stable_current_amd64.deb teamviewer_keyring.gpg virt.key
kubeProject teamviewer_keyring.gpg- virt-keyring.gpg
mongodb-define.yaml Templates  virt-keyring.gpg-
mongodb-secret.yaml Videos    'VirtualBox VMs'
```

Figure 3.74: Verify Ngrok file download

1.a.4: Extract Ngrok installation file with tar command.

```
(geekone@geekone)~$ tar zvfz ngrok-stable-linux-amd64.tgz
ngrok
(geekone@geekone)~$ ls
certs      ngrok      virt1.key
CTF        ngrok-stable-linux-amd64.tgz  virt1-keyring.gpg
Desktop    Pictures   virt1-keyring.gpg-
Documents  Public     virt.gpg
Downloads  root-ca.pem virt.key
google-chrome-stable_current_amd64.deb teamviewer_keyring.gpg virt-keyring.gpg
kubeProject teamviewer_keyring.gpg- virt-keyring.gpg-
mongodb-define.yaml Templates  'VirtualBox VMs'
mongodb-secret.yaml Videos    virt1.gpg
Music
```

Figure 3.75: Extract Ngrok installation file

1.a.5: Authenticate our Ngrok agent. This operation is done once and for all. The Authentication token used for this step is already saved in the default configuration file that we have previously extracted

```
(geekone@geekone)-[~]
$ ./ngrok authtoken 26tud0kpIQTjh80zgw2ylryPCHO_4nKWj39aFbv4bVsYo5LYT
Authtoken saved to configuration file: /home/geekone/.ngrok2/ngrok.yml
```

Figure 3.76: Authenticate our Ngrok agent

1.a.6: Inspect Ngrok file (verify if it's executable, if not you have to give it execute permission with chmod command)

```
(geekone@geekone)-[~]
$ file ngrok
ngrok: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, Go BuildID=bj-6b5zp0rd8kz4LsbjJ/L6lu
c_GG82vtt5Yflkf4/azzFJ5D-JUhY91hM9Ecj/M_j-cvtbbX85RCrJG7sf, not stripped
```

Figure 3.77: Inspect Ngrok file

1.a.7: Verify if the required ports by wazuh server are all open (View Documentation to learn more about the ports). If they are not open , use 'sudo ufw allow port/protocol'

```
(geekone@geekone)-[~]
$ sudo ufw status
Status: active

To Action From
--
1514/udp ALLOW Anywhere
1516/tcp ALLOW Anywhere
1514/tcp ALLOW Anywhere
1515/tcp ALLOW Anywhere
514/udp ALLOW Anywhere
514/tcp ALLOW Anywhere
55000/tcp ALLOW Anywhere
9200/tcp ALLOW Anywhere
9300/tcp ALLOW Anywhere
9400/tcp ALLOW Anywhere
443/tcp ALLOW Anywhere
1514/udp (v6) ALLOW Anywhere (v6)
1516/tcp (v6) ALLOW Anywhere (v6)
1514/tcp (v6) ALLOW Anywhere (v6)
1515/tcp (v6) ALLOW Anywhere (v6)
514/udp (v6) ALLOW Anywhere (v6)
514/tcp (v6) ALLOW Anywhere (v6)
55000/tcp (v6) ALLOW Anywhere (v6)
9200/tcp (v6) ALLOW Anywhere (v6)
9300/tcp (v6) ALLOW Anywhere (v6)
9400/tcp (v6) ALLOW Anywhere (v6)
443/tcp (v6) ALLOW Anywhere (v6)
```

Figure 3.78: Verify if the required ports

1.a.8 : Look For the command that will help us to expose our https local Wazuh server


```

NAME:
  ngrok - tunnel local ports to public URLs and inspect traffic

DESCRIPTION:
  ngrok exposes local networked services behinds NATs and firewalls to the
  public internet over a secure tunnel. Share local websites, build/test
  webhook consumers and self-host personal services.
  Detailed help for each command is available with 'ngrok help <command>'.
  Open http://localhost:4040 for ngrok's web interface to inspect traffic.

EXAMPLES:
  ngrok http 80 # secure public URL for port 80 web server
  ngrok http -subdomain=baz 8080 # port 8080 available at baz.ngrok.io
  ngrok http foo.dev:80 # tunnel to host:port instead of localhost
  ngrok http https://localhost # expose a local https server
  ngrok tcp 22 # tunnel arbitrary TCP traffic to port 22
  ngrok tls -hostname=foo.com 443 # TLS traffic for foo.com to port 443
  ngrok start foo bar baz # start tunnels from the configuration file

VERSION:
  2.3.40

AUTHOR:
  inconshreveable - <alan@ngrok.com>

COMMANDS:
  authtoken  save authtoken to configuration file
  credits    prints author and licensing information
  http       start an HTTP tunnel
  start      start tunnels by name from the configuration file
  tcp        start a TCP tunnel
  tls        start a TLS tunnel
  update     update ngrok to the latest version
  version    print the version string
  help       Shows a list of commands or help for one command

```

Figure 3.79: Discover possible commands by Ngrok with `-help` attribute

Expose our https local Wazuh server 1.b.1 : Now let's expose our https local Wazuh server

```

(geekone@geekone)-[~]
$ ./ngrok http https://localhost

```

Figure 3.80: expose our https local Wazuh server

1.b.2: Here you can view the execution of Ngrok command, our local https server is exposed and it can be accessed on the internet with the highlighted (sub-)domain

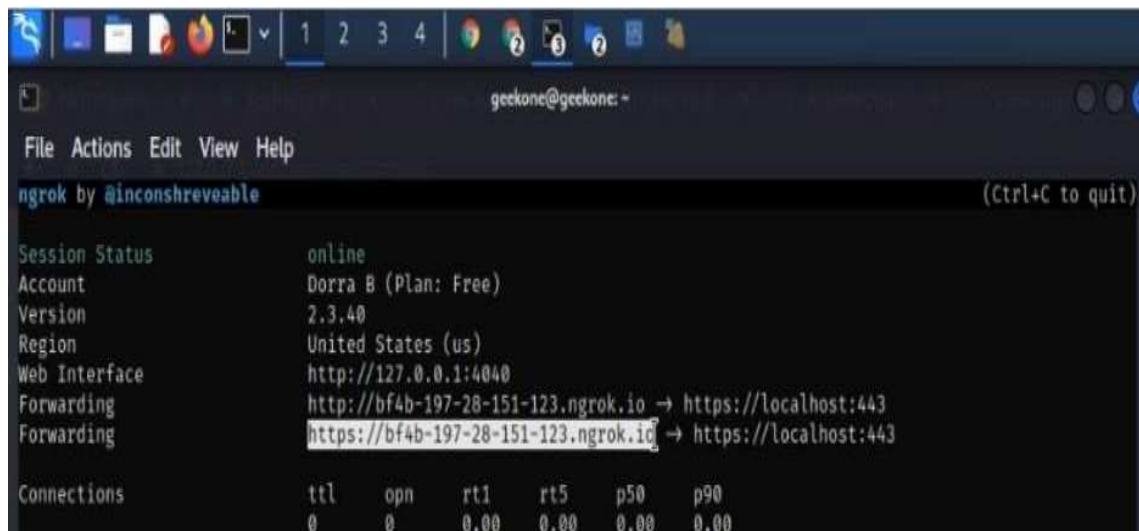


Figure 3.81: Display of Public server Information

1.b.3: You can copy and paste the https Url and try to login to the browser. continue with "This unsafe website"

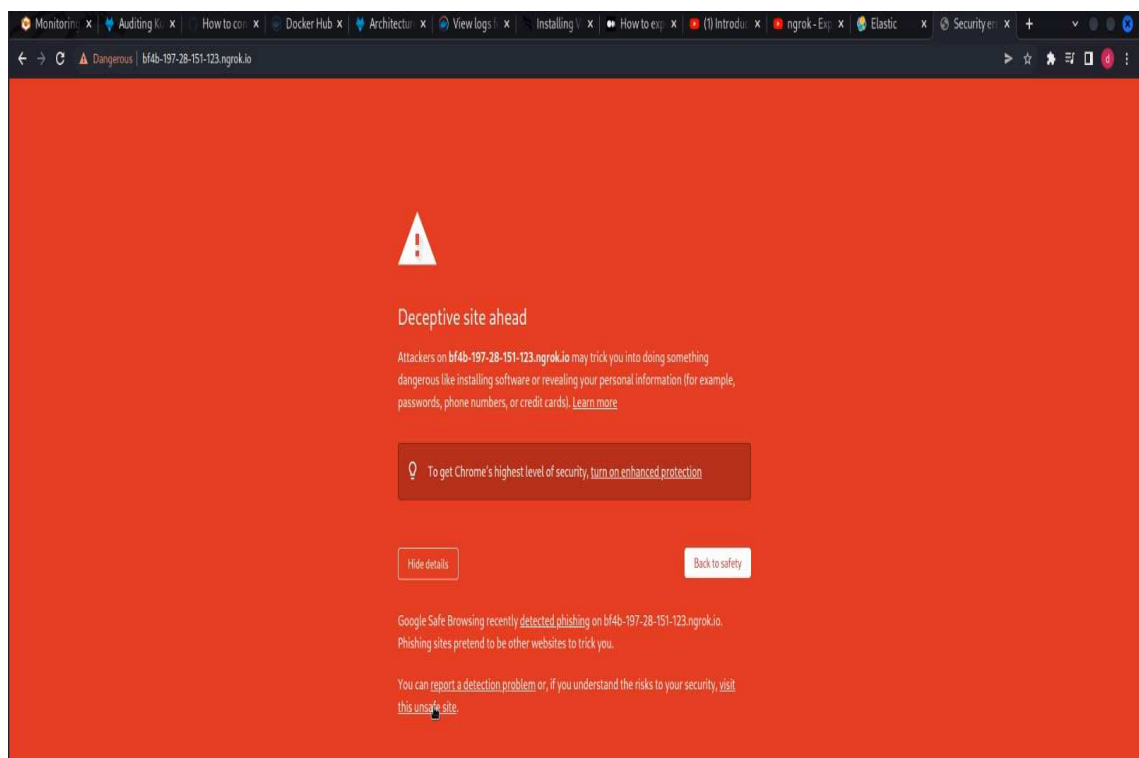


Figure 3.82: Login to our Internet exposed Wazuh local server

1.b.4: Log-in is successful , Our local Wazuh server is finally exposed to the internet

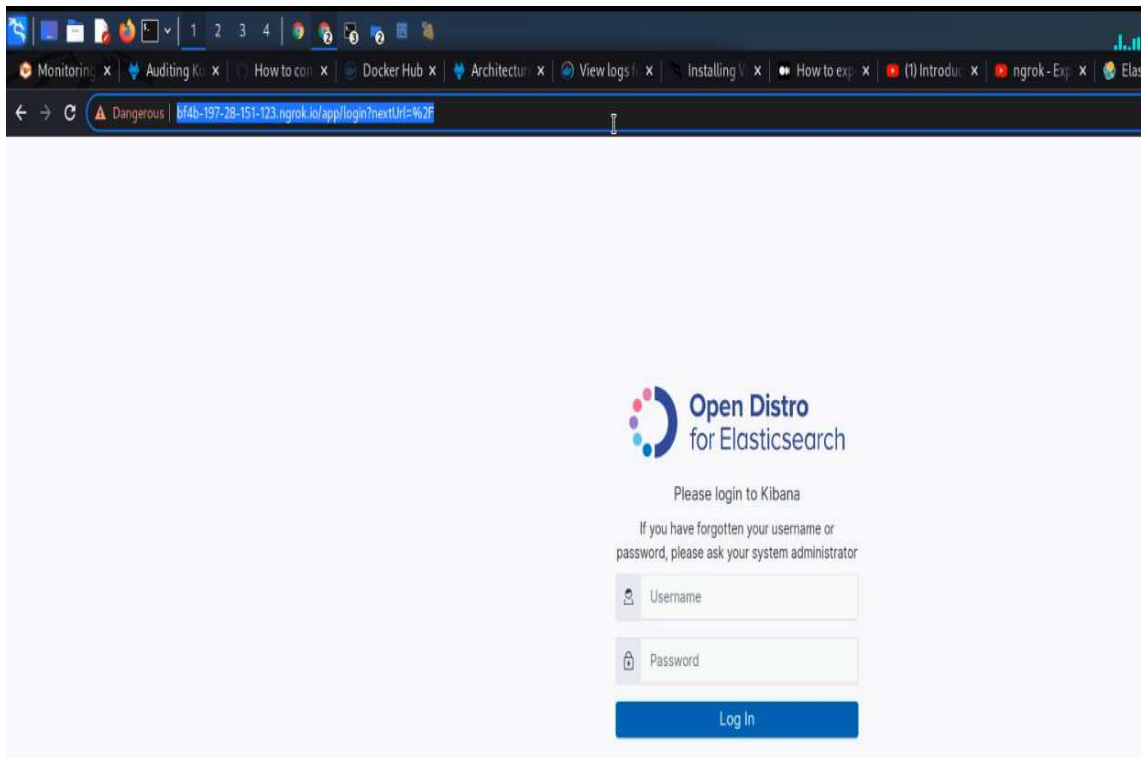


Figure 3.83: Local server is exposed successfully to the internet

3.3.8.2 Expose Frontend tier of our Application to the Wazuh server through the internet

To be able to reach this EC2 instance, we have to give our server access permissions to the EC2 instance while modifying the configuration of the Security group attached to the EC2 instance.

A security group controls incoming and outgoing traffic for your EC2 instances by acting as a virtual firewall. Inbound rules regulate the traffic that comes into your instance, while outbound rules control the traffic that leaves it. You can specify one or more security groups when launching an instance. To permit http and https connection between the Wazuh server and EC2 instance.

2.a : Notice the Frontend Tier Worker Node on Amazon EKS Cluster Dashboard and in the EC2 Dashboard

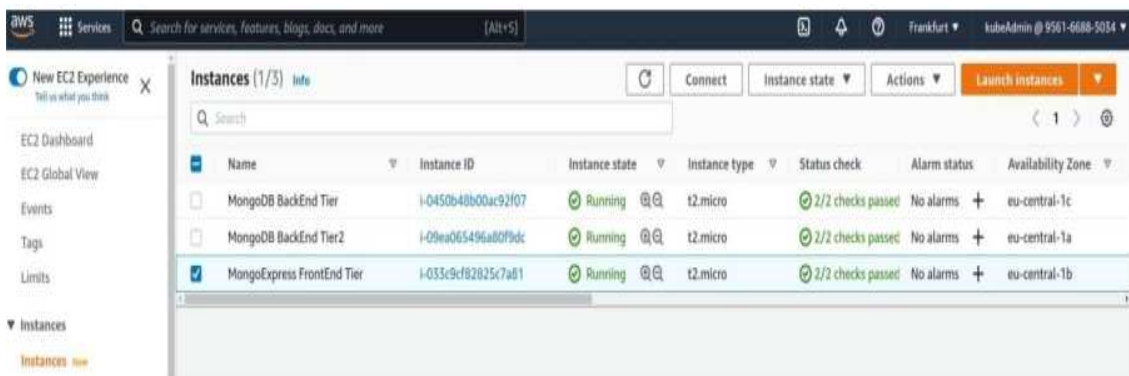
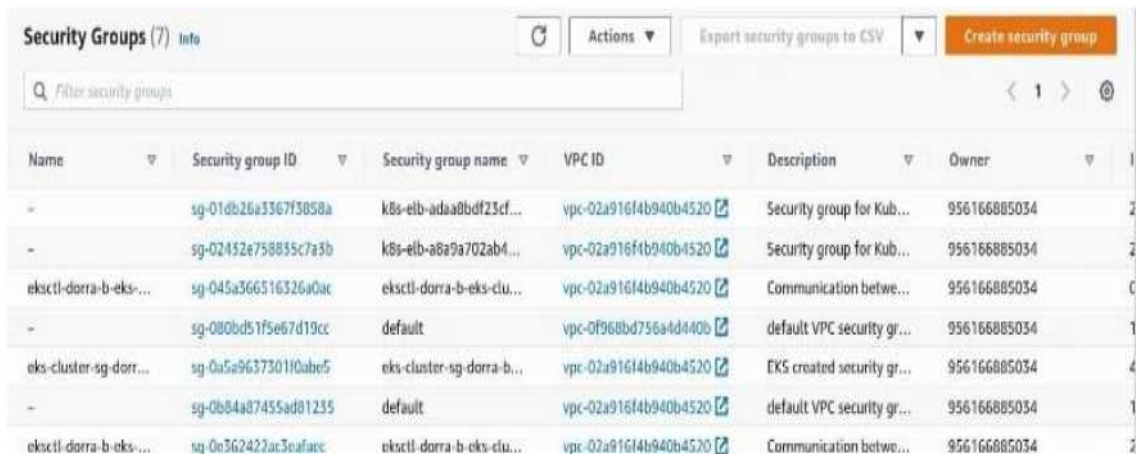


Figure 3.84: Application Frontend Tier is an EC2 instance

2.b : View the Security group section and determine which one is of our concern

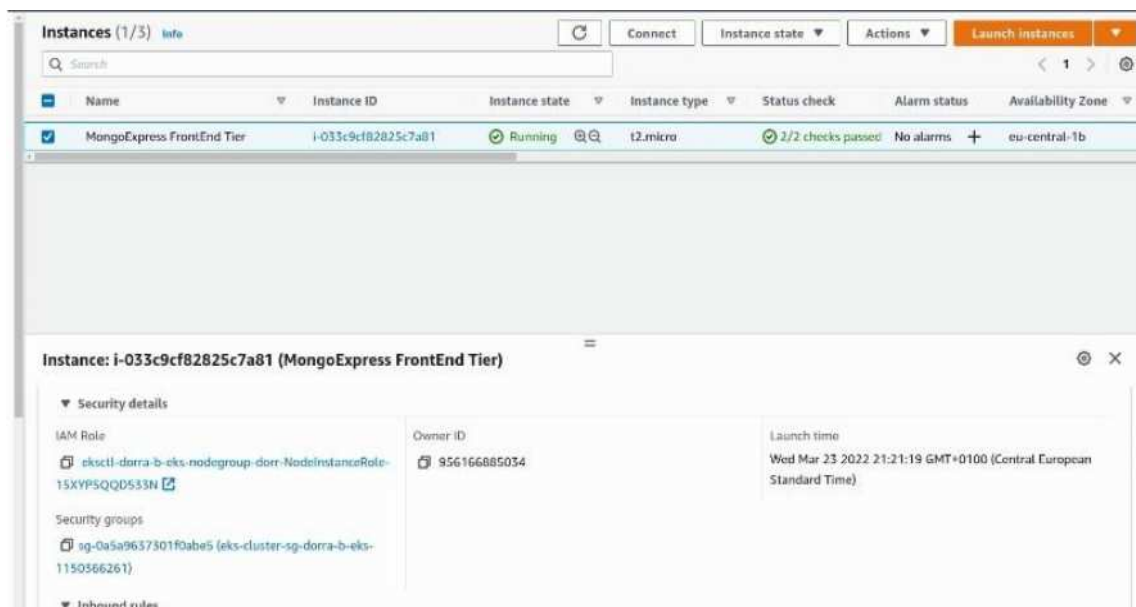


The screenshot shows the AWS Management Console 'Security Groups' page. It displays a table with columns: Name, Security group ID, Security group name, VPC ID, Description, and Owner. The table lists several security groups, including 'k8s-elb-adad0bdf23cf...', 'k8s-elb-a8a9a702ab4...', 'eksctl-dorra-b-eks-...', 'eks-cluster-sg-dorra-...', and 'eksctl-dorra-b-eks-...'.

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-01db26a3367f3858a	k8s-elb-adad0bdf23cf...	vpc-02a916f4b940b4520	Security group for Kub...	956166885034
-	sg-02452e758855c7a3b	k8s-elb-a8a9a702ab4...	vpc-02a916f4b940b4520	Security group for Kub...	956166885034
eksctl-dorra-b-eks-...	sg-045a366516326a0ac	eksctl-dorra-b-eks-clu...	vpc-02a916f4b940b4520	Communication betwe...	956166885034
-	sg-080bd51f5e67d19cc	default	vpc-0f968bd756a4d440b	default VPC security gr...	956166885034
eks-cluster-sg-dorra-...	sg-0a5a9637301f0abe5	eks-cluster-sg-dorra-b...	vpc-02a916f4b940b4520	EKS created security gr...	956166885034
-	sg-0b84a87455ad81235	default	vpc-02a916f4b940b4520	default VPC security gr...	956166885034
eksctl-dorra-b-eks-...	sg-0e362422ac3eafacc	eksctl-dorra-b-eks-clu...	vpc-02a916f4b940b4520	Communication betwe...	956166885034

Figure 3.85: Security Group

2.c : View the Security Group attached to the Frontend tier EC2 instance



The screenshot shows the AWS Management Console 'Instances' page. It displays a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. The instance 'MongoExpress FrontEnd Tier' is highlighted. Below the table, the details for this instance are shown, including IAM Role, Security groups, and Inbound rules.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
MongoExpress FrontEnd Tier	i-033c9cf82825c7a81	Running	t2.micro	2/2 checks passed	No alarms	eu-central-1b

Instance: i-033c9cf82825c7a81 (MongoExpress FrontEnd Tier)

- IAM Role:** eksctl-dorra-b-eks-nodegroup-dorra-NodeInstanceRole-15XVPSQQD533N
- Owner ID:** 956166885034
- Launch time:** Wed Mar 23 2022 21:21:19 GMT+0100 (Central European Standard Time)
- Security groups:** sg-0a5a9637301f0abe5 (eks-cluster-sg-dorra-b-eks-1150366261)
- Inbound rules:**

Figure 3.86: the Security Group attached to the Frontend tier EC2 instance

2.d : View the inbound rules of the frontend tier which are related to the allowed Protocols, Ports and IP addresses to get IN the EC2 instance. Inbound rules can be also a predefined security group

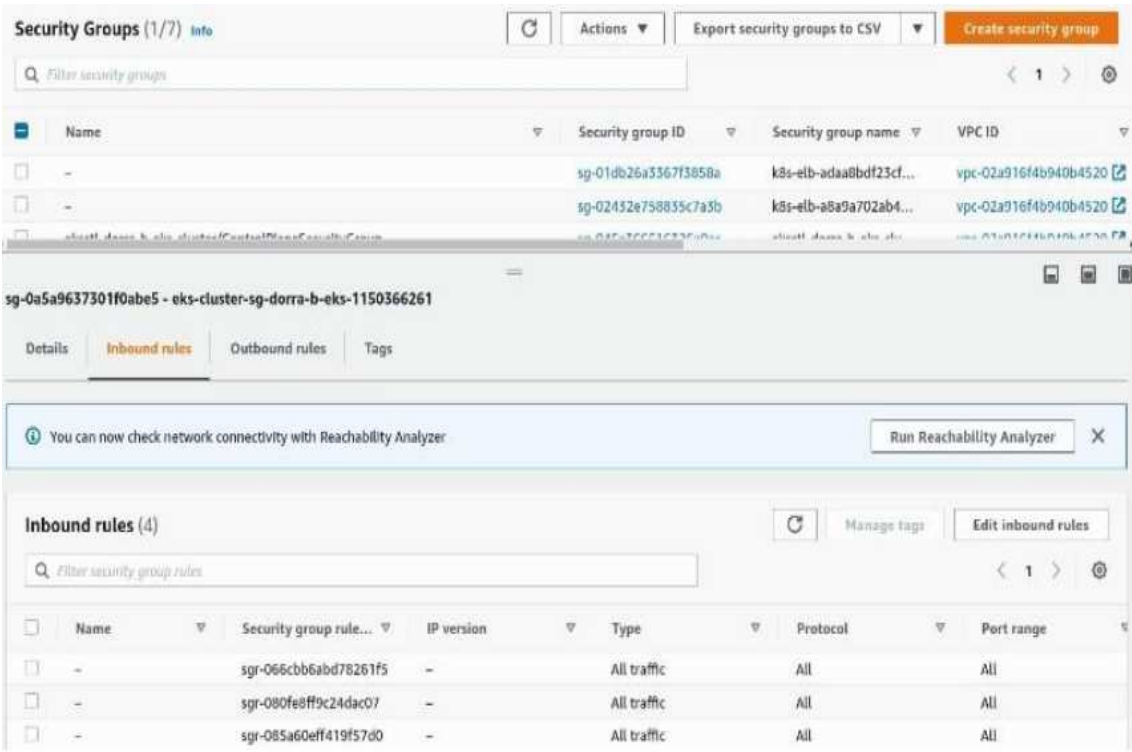


Figure 3.87: Inbound Rules

2.e : Go into the Security group configuration and Click on 'Add rule'

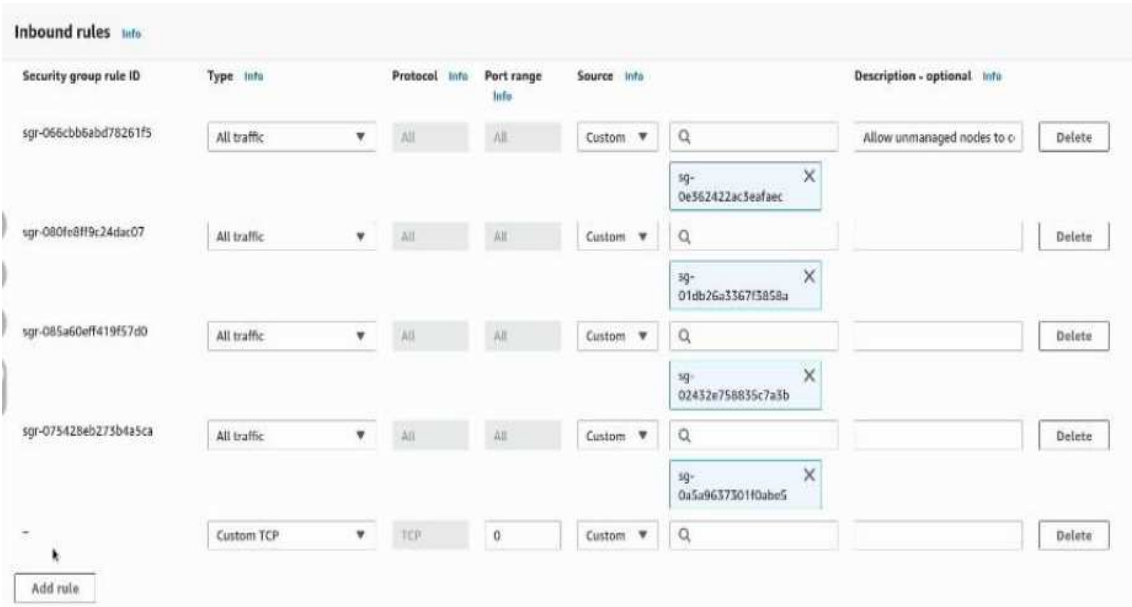


Figure 3.88: Add the rules

2.f : Define the rules while using the public IP address of Wazuh server (View Wazuh server documentation to see required ports :1515/TCP,1514/TCP, 1514/UDP ...)



Figure 3.89: Start by adding rules

What is NOT Recommended: This is to define the IP address as 0.0.0.0/0 because this gives the permission for everyone to access your EC2 instance.

If you want to ssh to the EC2 instance, you have to add a permission on SSH protocol for the wazuh server public IP address.

2.g : Define the outbound rules

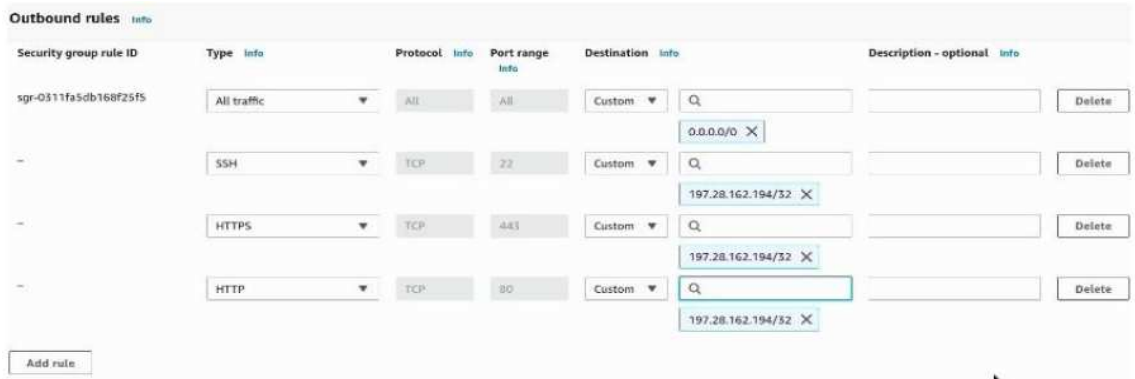


Figure 3.90: Outbound Rules

2.h: Security Group is updated successfully

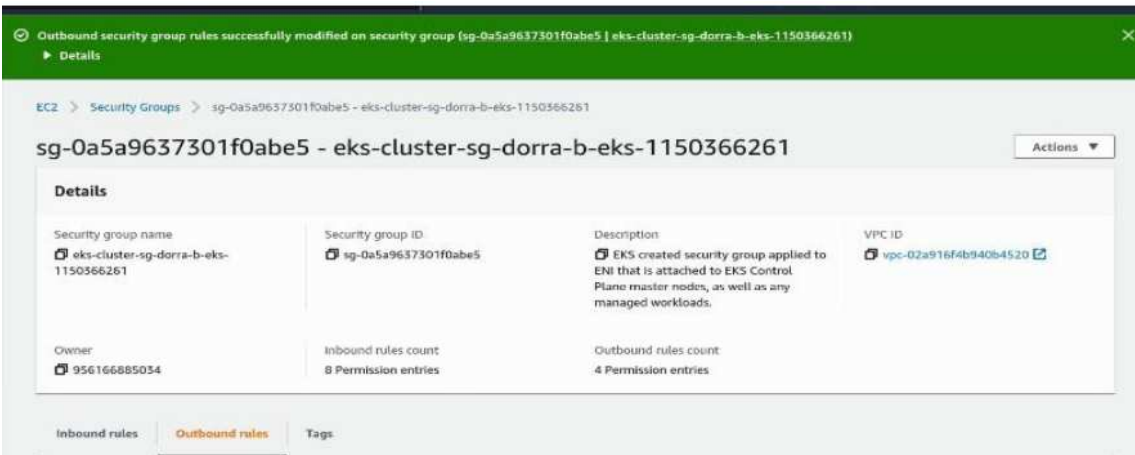


Figure 3.91: Successful Update of Security group corresponding to the Frontend Tier EC2 instance

3.3.8.3 Establish the connection between Wazuh server and Frontend Tier Worker Node

3.a: Connect to the Frontend Tier Worker Node which will be the wazuh agent. The Frontend Tier Worker Node is an EC2 instance to which we can connect with ssh or simply with AWS Session Manager

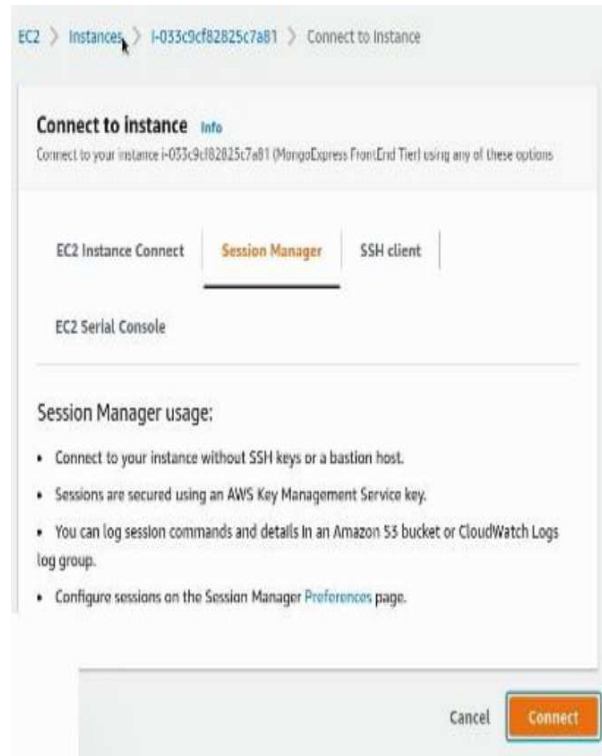


Figure 3.92: Go to the Session Manager of the Frontend Tier Worker Node (E2)

3.b : In Wazuh server Dashboard , Add an agent and copy the provided command that should be run on the agent.

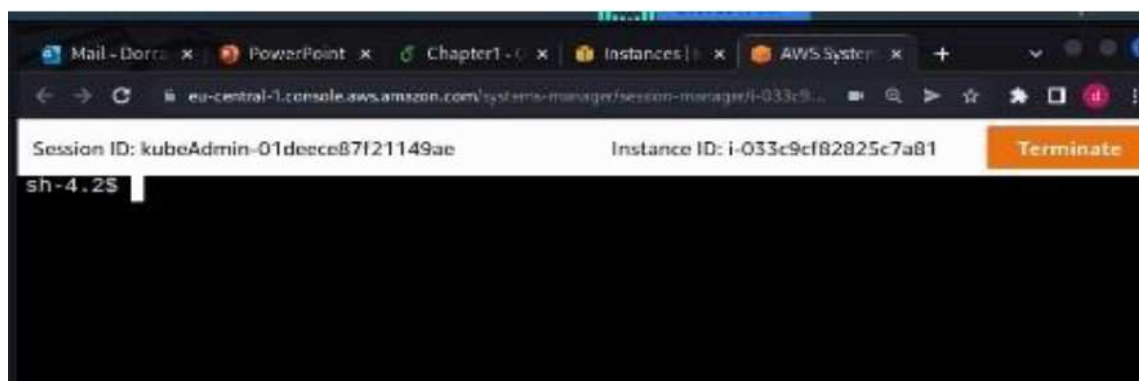


Figure 3.93: Establish connection to the Frontend Tier Worker Node (E2)

3.c : First go to 'active agents'



Figure 3.94: Go to the Session Manager of the Frontend Tier Worker Node (E2)

3.d: Give the details related to the agent while giving the server name as the domain name provided by ngrok

- To know what family on linux does the Amazon linux belong we can run the command:
' cat /etc/os- release '

```
Session ID: kubeAdmin-0c6b7fcad60f17e3d Instance ID: i-033c9cf82825c7a8

[root@ip-192-168-50-103 ~]# cat /etc/os-release
NAME="Amazon Linux"
VERSION="2"
ID="amzn"
ID_LIKE="centos rhel fedora"
VERSION_ID="2"
PRETTY_NAME="Amazon Linux 2"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2"
HOME_URL="https://amazonlinux.com/"
```

Figure 3.95: Linux Wazuh agent Os characteristics

- Copy the (sub-)domain provided by ngrok

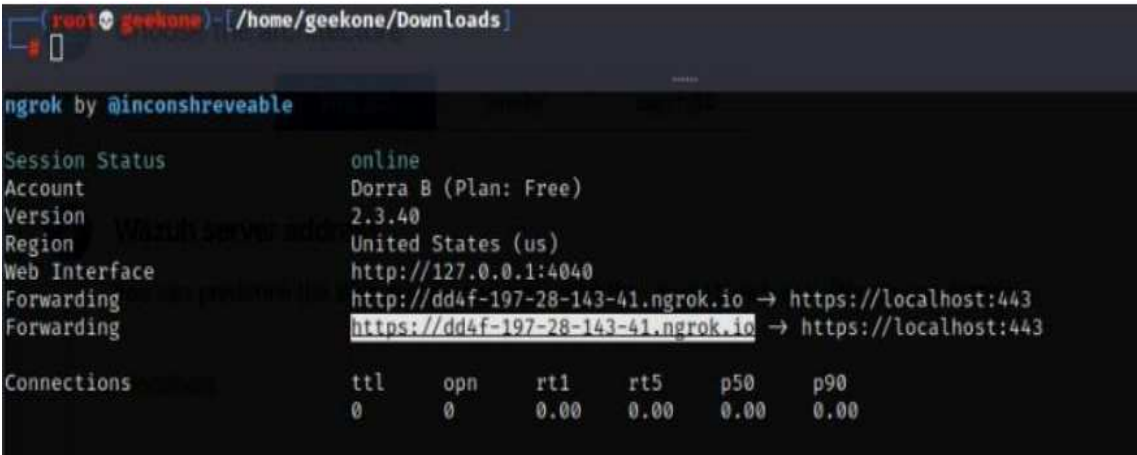


Figure 3.96: ngrok Session Status

- Fill in the agent parameters



Deploy a new agent

1 Choose the Operating system

Red Hat / CentOS Debian / Ubuntu Windows MacOS

2 Choose the version

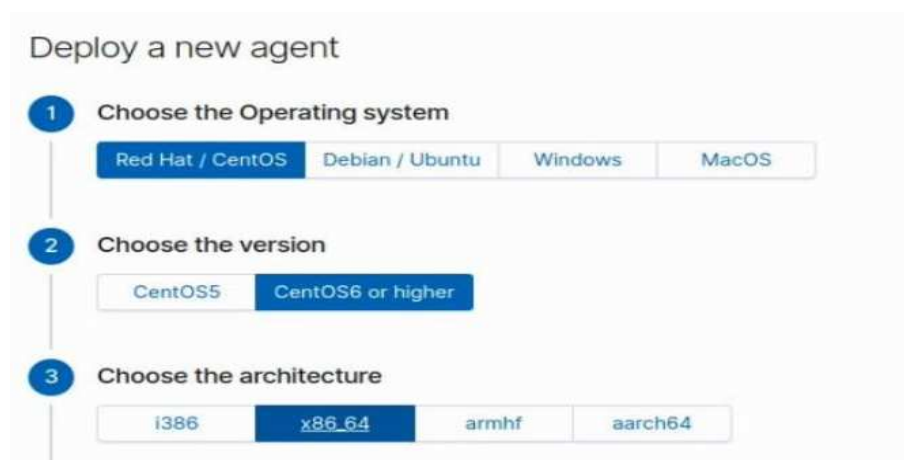
CentOS5 CentOS6 or higher

3 Choose the architecture

i386 x86_64 armhf aarch64

Figure 3.97: Fill in the agent parameters 1.0

- Copy the command and run it on the targeted EC2 instance



Deploy a new agent

1 Choose the Operating system

Red Hat / CentOS Debian / Ubuntu Windows MacOS

2 Choose the version

CentOS5 CentOS6 or higher

3 Choose the architecture

i386 x86_64 armhf aarch64

Figure 3.98: Fill in the agent parameters 1.0

4 Wazuh server address
You can predefine the Wazuh server address with the `enrollment.dns` Wazuh app setting.

`dd4f-197-28-143-41.ngrok.io`

5 Assign the agent to a group
Select one or more existing groups

`default` x

6 Install and enroll the agent
You can use this command to install and enroll the Wazuh agent in one or more hosts.

③ Running this command on a host with an agent already installed upgrades the agent package without enrolling the agent. To enroll it, see the [Wazuh documentation](#).

```
sudo WAZUH_MANAGER='dd4f-197-28-143-41.ngrok.io' WAZUH_AGENT_GROUP='default' yum install
https://packages.wazuh.com/4.x/yum/wazuh-agent-4.2.5-1.x86_64.rpm
```

Copy command

Figure 3.99: Fill in the agent parameters 1.1

```
[root@ip-192-168-50-103 admin]# sudo WAZUH_MANAGER='dd4f-197-28-143-41.ngrok.io' WAZUH_AGENT_GROUP='default' yum install https://packages.wazuh.com/4.x/yum/wazuh-agent-4.2.5-1.x86_64.rpm
Loaded plugins: priorities, update-motd, versionlock
wazuh-agent-4.2.5-1.x86_64.rpm | 6.4 MB 00:00:00
Examining /var/tmp/yum-root-dZGM0Q/wazuh-agent-4.2.5-1.x86_64.rpm: wazuh-agent-4.2.5-1.x86_64
Marking /var/tmp/yum-root-dZGM0Q/wazuh-agent-4.2.5-1.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
```

Figure 3.100: Run the command on FrontEnd Tier worker node

Now, we have to restart the daemonset as well as the wazuh-agent service .

```
Session ID: kubeAdmin-Of416ed37a61731b3 Instance ID: I-033c9cf82825c7a81

[root@ip-192-168-50-103 admin]# sudo systemctl daemon-reload
[root@ip-192-168-50-103 admin]# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@ip-192-168-50-103 admin]# systemctl enable wazuh-agent
Created symlink from /etc/systemd/system/multi-user.target.wants/wazuh-agent.service to /usr/lib/systemd/system/wazuh-agent.service.
[root@ip-192-168-50-103 admin]# systemctl start wazuh-agent
[root@ip-192-168-50-103 admin]# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2022-04-25 10:10:10 CEST; 1min 1s ago
```

Figure 3.101: Restart Daemon and Wazuh-agent service

At this moment, we can see clearly the Active connected agent on Wazuh server Dashboard

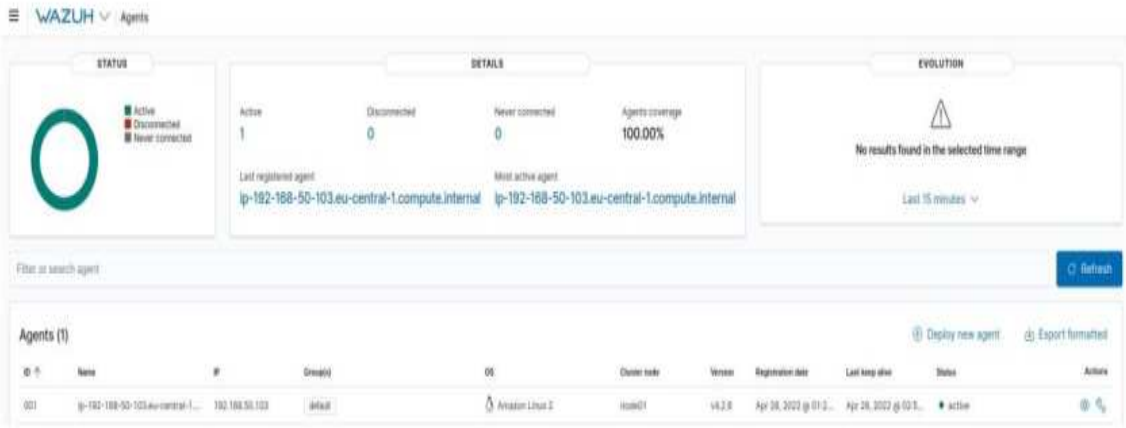


Figure 3.102: Connected Agent

Conclusion

Many approaches can be used to reach this same model of deployment, but our main purpose in this work is to reach the best results while minimizing the expenses. This is the main reason why we hosted the wazuh server on our personal computer. But ,we have also tried to create the wazuh server on an EC2 instance while using Amazon AMI image. The cost is remarkably high for this case.

Chapter 4

Platform Vulnerability, Attack Scenarios, and recommendations

Introduction

Now , once we have our infrastructure and applications ready, it's time to apply some attack scenarios and see what result will Wazuh provide. This cases will inform us on how far can we inspect , monitor and protect an Amazon EKS Cluster with Wazuh Open Source SIEM

4.1 Vulnerability detection

4.1.1 How It Works?

To detect vulnerabilities, agents may now collect a list of installed apps natively and transmit it to the manager on a regular basis (where it is stored in local sqlite databases, one per agent). The manager also creates a global vulnerability database from publicly available CVE repositories, which he then uses to cross-correlate with the agent's application inventory data.

The documentation of Wazuh provides a user manual that lists the repositories from which data is pulled to create the global vulnerability database.

- <https://canonical.com>: Used to pull CVEs for Ubuntu Linux distributions.
- <https://www.redhat.com>: Used to pull CVEs for Red Hat and CentOS Linux distributions.
- <https://www.debian.org>: Used to pull CVEs for Debian Linux distributions.
- <https://nvd.nist.gov/>: Used to pull CVEs from the National Vulnerability Database.
- <https://feed.wazuh.com/>: Used to pull the MSU feed with CVEs and patches for Microsoft products.

Figure 4.1: Repositories[5]

This database may be set to be updated on a regular basis, ensuring that the solution is always up to date.

Since the Amazon linux version is like CentOS , we will mainly need <https://www.redhat.com> to pull our CVEs of concern.

4.1.1.1 What is a CVE?

The acronym CVE stands for Common Vulnerabilities and Exposures, and it is a collection of publicly documented computer security issues. When someone mentions a CVE, they're referring to a security problem with a CVE ID number. At least one CVE ID is nearly usually mentioned in vendor and researcher security warnings.

The detection mechanism looks for susceptible packages in the inventory databases when the global vulnerability database (with CVEs) is built (unique per agent). When a CVE (Common Vulnerabilities and Exposures) affects a package known to be installed on one of the monitored servers, an alert is produced. When a package's version falls inside a CVE's affected range, it is considered vulnerable. The findings are shown in the form of alerts and are also saved in a database. So you may look at the most recent scan alerts or query each agent's susceptible software database.

4.1.2 Scan Types:

There are two types of scans:

- **Full scan:** The first time, Vulnerability Detector scans every single package installed. After this, all the available packages are scanned again only when the configured `ignore_time` expires.
- **Partial scans:** Only new packages are scanned while `ignore_time` is still valid.

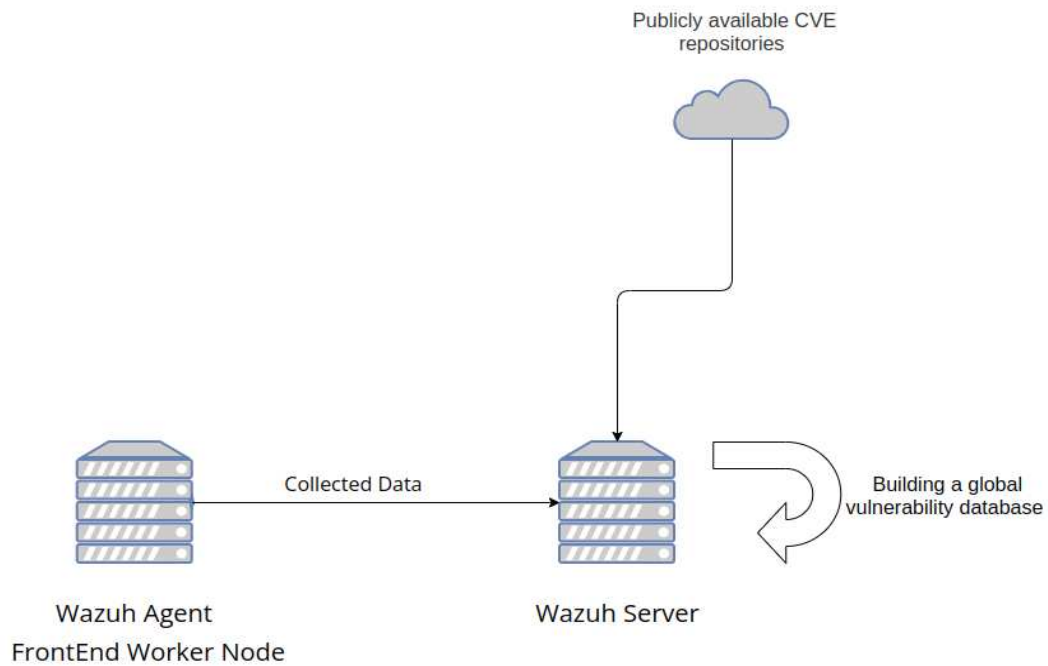


Figure 4.2: Vulnerability detection with Feeds from CVE

Through the next example of DoS attack ,we will see clearly a vulnerability detection witness .

4.2 DoS attack scenario

4.2.1 What is a DoS attack

A Denial-of-Service (DoS) attack is one that attempts to bring a system or network to a halt, rendering it unreachable to its intended users. DoS attacks work by inundating the target with traffic or delivering it information that causes it to crash.

4.2.2 DoS Attack scenario

We intend to create a SYNflood DDOS attack on the frontend Tier. An attacker who often uses a false IP address sends repeated SYN packets to every port on the targeted server in a SYN flood attack. Unaware of the assault, the server receives many, seemingly valid requests for communication and responds to each attempt with a SYN-ACK packet from each open port. This makes the victim overwhelmed and the service becomes denied.

Here , the SYNflood attack will be done on the frontend tier with Metasploit

```
(geekone@geekone)-[~]
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	40d
mongodb-service	ClusterIP	10.100.78.225	<none>	27017/TCP	40d
mongoexpress-service	LoadBalancer	10.100.81.186	adaa8bdf23cf247e9882395db25516ae-1557553037.eu-central-1.elb.amazonaws.com	8081:30000/TCP	40d
wazuh-agent-service	LoadBalancer	10.100.62.255	a8a9a702ab4694982966893d231b3c43-744030852.eu-central-1.elb.amazonaws.com	55000:30001/TCP	39d

Figure 4.3: The Exposed service of frontend tier which we will attack

```
(geekone@geekone)-[~]
$ ping 18.156.195.174
PING 18.156.195.174 (18.156.195.174) 56(84) bytes of data:
^C
— 18.156.195.174 ping statistics —
5 packets transmitted, 0 received, 100% packet loss, time 4088ms

(geekone@geekone)-[~]
$ wget 18.156.195.174:8081
--2022-05-03 01:51:48-- http://18.156.195.174:8081/
Connecting to 18.156.195.174:8081... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7942 (7.8K) [text/html]
Saving to: 'index.html'

index.html          100%[=====>] 7.76K --.-KB/s  in 0.01s
2022-05-03 01:51:49 (801 KB/s) - 'index.html' saved [7942/7942]
```

Figure 4.4: Service DO NOT accept ICMP protocol as protection measure

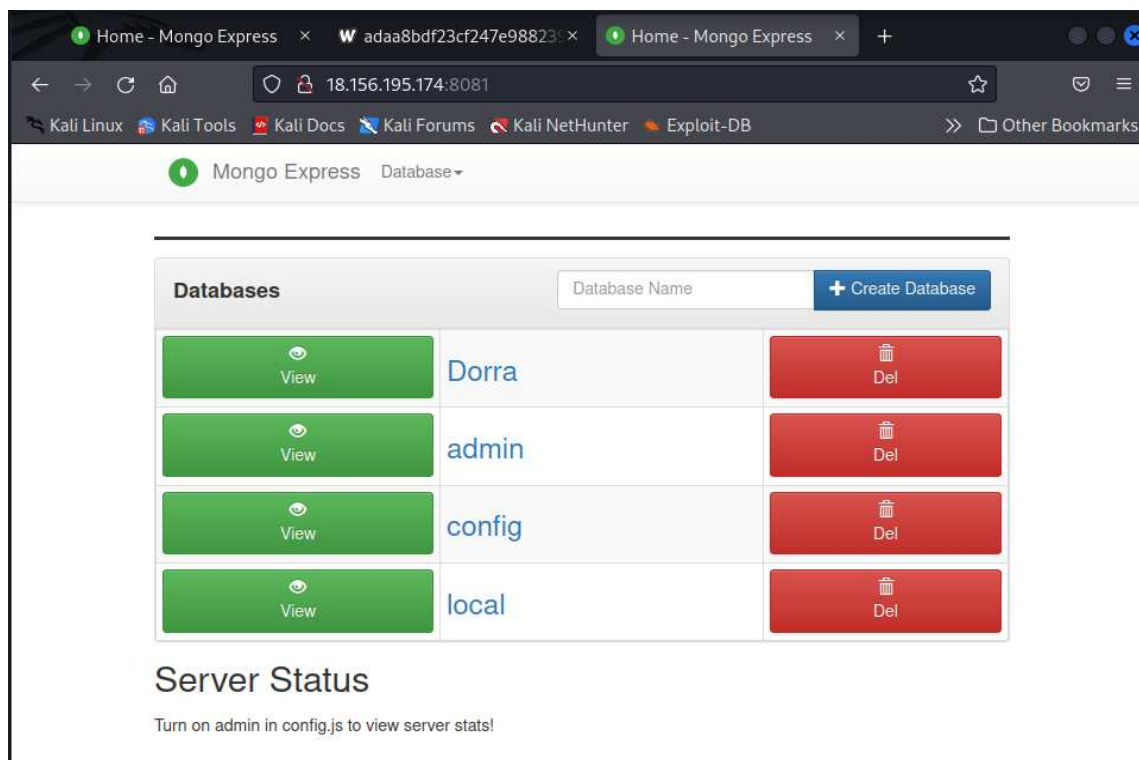


Figure 4.5: Display of our application on the browser

Now, to find out which port we will attack on that server, we will run at first nmap to scan open ports.

```
(geekone@geekone)-[~]
$ sudo nmap 18.156.195.174
[sudo] password for geekone:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-03 02:13 CET
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.12 seconds
```

Figure 4.6: nmap scan did not provide any information about open Ports

It's crystal clear that the Frontend tier service created by kubernetes to expose the MongoExpress deployment is immune. It blocks our ping probes. This is why we will do a deeper scan with -Pn option:

```
(geekone@geekone)-[~]
$ sudo nmap -Pn 18.156.195.174
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-03 02:14 CET
Nmap scan report for ec2-18-156-195-174.eu-central-1.compute.amazonaws.com (18.156.195.174)
Host is up (0.073s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
8081/tcp  open  blackice-icecap

Nmap done: 1 IP address (1 host up) scanned in 173.63 seconds
```

Figure 4.7: nmap deep scan shows an open port

We can see that , finally , we have an exposed port 8081/tcp that we can attack and it's open.

Now, let's start this attack with metasploit

```
(geekone@geekone)-[~]
$ sudo msfconsole
[sudo] password for geekone:
Sorry, try again.
[sudo] password for geekone:

METASPLOIT CYBER MISSILE COMMAND V5

#####
# WAVE 5 ##### SCORE 31337 ##### HIGH FFFFFFFF #
#####
https://metasploit.com

Metasploit tip: You can upgrade a shell to a Meterpreter
session on many platforms using sessions -u <session_id>
```

Figure 4.8: Start Metasploit on kali Linux

```
msf6 > search synflood

Matching Modules

# Name Disclosure Date Rank Check Description
- - - - -
0 auxiliary/dos/tcp/synflood normal No TCP SYN Flooder

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood
```

Figure 4.9: Search for the location of synflood auxiliary

```
msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) >
```

Figure 4.10: use the SYNflood auxiliary

```
msf6 auxiliary(dos/tcp/synflood) > show options
Module options (auxiliary/dos/tcp/synflood):


| Name      | Current Setting | Required | Description                                                                                  |
|-----------|-----------------|----------|----------------------------------------------------------------------------------------------|
| INTERFACE |                 | no       | The name of the interface                                                                    |
| NUM       |                 | no       | Number of SYNs to send (else unlimited)                                                      |
| RHOSTS    |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit |
| RPORT     | 80              | yes      | The target port                                                                              |
| SHOST     |                 | no       | The spoofable source address (else randomizes)                                               |
| SNAPLEN   | 65535           | yes      | The number of bytes to capture                                                               |
| SPORT     |                 | no       | The source port (else randomizes)                                                            |
| TIMEOUT   | 500             | yes      | The number of seconds to wait for new data                                                   |


```

Figure 4.11: Show and select the SYNflood option to use

```
msf6 auxiliary(dos/tcp/synflood) > set SHOST 1.2.3.4
SHOST => 1.2.3.4
msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 18.156.195.174
RHOSTS => 18.156.195.174
msf6 auxiliary(dos/tcp/synflood) > set RPORT 8081
RPORT => 8081
```

Figure 4.12: Set A spoofable source address, set the targeted IP and Port as well

```
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 18.156.195.174
[*] SYN flooding 18.156.195.174:8081 ...
```

Figure 4.13: Start flooding the application with the command exploit

While flooding , we try to access the web interface of our application. We can remark clearly that it becomes more and more impossible to access and it keeps just loading...We cannot view, add, or delete a database.
The access is denied.

4.2.3 Wazuh Outcome:

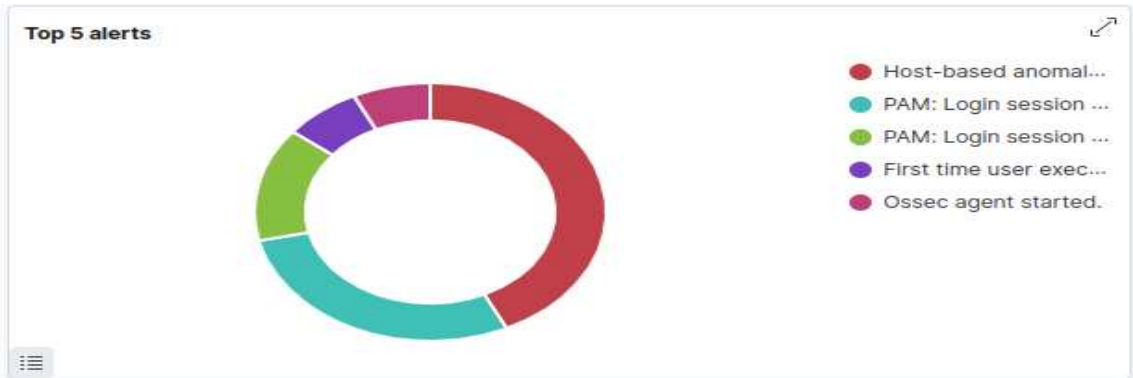


Figure 4.14: The DOS attack Wasn't identified

Unfortunately, Wazuh didn't provide the expected output to receive after the DOS attack scenario is performed. And through our research, it looks like this SIEM is not the best choice to detect DOS attacks

I have been trying to DoS one of my wazuh agent, and it's not produce any log, nor in syslog, or access.log.
Is it possible for wazuh ossec to detect DDoS attack ?

Figure 4.15: The Common asked Question

There is nothing configured for this in Wazuh. This does not mean that we cannot configure certain options to interpret this type of attack and act if possible, however, due to performance, we recommend the use of third-party tools designed for this purpose.

For example, an Apache web server obtains logs of each request. We could create a rule that detects if an event recorded by the Apache web server (a request) happens repeatedly, and then triggers the active response and blocks the network.

This way of action would be viable, but there are tools to detect DDoS attacks in a faster and more precise way, such as ModSecurity.

Kind regards,

Figure 4.16: Professional Response on the Expected outcome

4.3 Network Scanning

To establish the DOS attack , we had to scan the ports of our target with nmap tool while using the -Pn attribute because the target blocks ping probes (ICMP protocol is not allowed)

```
(geekone@geekone)-[~]
$ sudo nmap -Pn 18.156.195.174
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-03 02:14 CET
Nmap scan report for ec2-18-156-195-174.eu-central-1.compute.amazonaws.com (18.156.195.174)
Host is up (0.073s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
8081/tcp   open  blackice-icecap
Nmap done: 1 IP address (1 host up) scanned in 173.63 seconds
```

Figure 4.17: nmap deep scan shows an open port

Here is the output of Wazuh. The abnormal scan of ports was immediately detected:

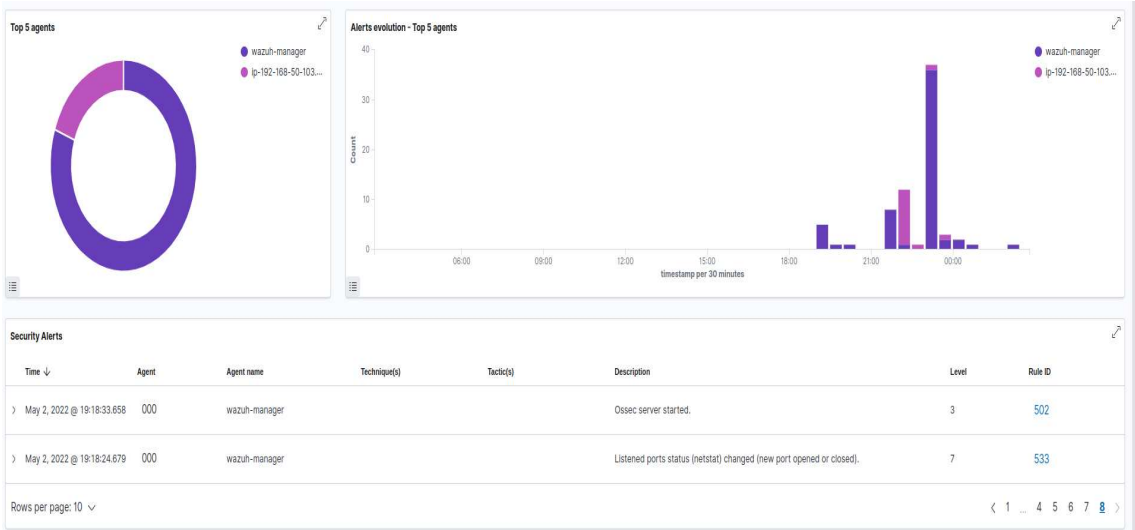


Figure 4.18: Port scan is detected by Wazuh

Conclusion

We can see clearly that Wazuh is a powerful tool with a long set of capabilities such as Log data collection, File integrity monitoring , vulnerability detection, etc. But , for some other set of attacks such as DOS , it's recommended to use a third tier dedicated for the detection of DOS and DDOS .

Chapter 5

Conclusion et perspectives

Introduction

Although this implementation provided a considerable privilege while detecting threats and susceptible attacks on frontend tier, there are still some limitations that we need to consider.

5.1 Limitations

5.1.1 Ngrok is good, but....

Although Ngrok is acknowledged as a very reliable solution to expose local server to Internet users, it has a pertinent disadvantage. In fact, each time you need to expose your server, you will be provided with randomly chosen public available (sub-)domain that may be accessed over the internet.

This randomness makes it difficult to whitelist Wazuh server on the security group of Frontend tier EC2. Furthermore, you can remark while browsing our website, we had to click to "This unsafe website" which is NOT recommended. We have to get a certificate from ngrok for this web interface and add it to our browser in order to establish an ssl/tls secure connection with the client

5.1.2 Whitelisting is nearly impossible

While being entities without fixed IP, the Wazuh Server will have all the agents disconnected ones it reboots or the agents reboot. AWS EC2 instances who have Public IP addresses change public IP addresses each time they restart. The optimal solution for this issue is the allocation of **Elastic IP addresses** for the instances instead of **Public IP addresses**.

5.1.3 Wazuh is an excellent HIDS, but

Wazuh is an excellent HIDS (Host-based Intrusion Detection System) among other things. In addition to its rule-based analysis of log events from agents and other devices, it also performs file integrity monitoring and anomaly detection. This provides a great deal of

insight into the security of your digital assets. However, some security issues are most successfully detected by inspecting a server's actual network traffic, which generally is not accounted for in logs. This is where a NIDS (Network Intrusion Detection System) can provide additional insight into your security in a way that is highly complementary to the HIDS functionality in Wazuh.

Conclusion

Through the first version of this work, we can see clearly a considerable number of limitations that we have to treat carefully as well as incomplete security provided to the Amazon EKS cluster as well. To make an immune cluster , we need a complete set of tools , along with Wazuh , to reach a desirable state of protection.

Annex

This annex contains the yaml files that we used to build the deployments as well as the configMap , the secret, and services

```
(geekone@geekone)-[~/kubeProject]
$ cat mongoExpress-definition.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-express
  labels:
    app: mongo-express
spec:
  replicas: 2
  selector:
    matchLabels:
      app: mongo-express
  template:
    metadata:
      labels:
        app: mongo-express
    spec:
      containers:
        - name: mongo-express
          image: mongo-express
          ports:
            - containerPort: 8081
          env:
            - name: ME_CONFIG_MONGODB_ADMINUSERNAME
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo_admin_USER_NAME
            - name: ME_CONFIG_MONGODB_ADMINPASSWORD
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo_admin_PASSWORD
            - name: ME_CONFIG_MONGODB_SERVER
              valueFrom:
                configMapKeyRef:
                  name: mongodb-configmap
                  key: mongodb_url
```

Figure 5.1: The Frontend MongoExpress Deployment Yaml Definition file


```
(geekone@geekone)-[~/kubeProject]
$ cat mongodb-definition.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 2
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo
          ports:
            - containerPort: 27017
          env:
            - name: MONGO_INITDB_ROOT_USERNAME
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo_admin_USER_NAME
            - name: MONGO_INITDB_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo_admin_PASSWORD
```

Figure 5.2: The Backend MongoDB Deployment Yaml Definition file

```
(geekone@geekone)-[~/kubeProject]
$ cat clusterIP-definition.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

Figure 5.3: Internal service Yaml Definition File


```
(geekone@geekone)-[~/kubeProject]
$ cat mongoNodeIP-definition.yaml
apiVersion: v1
kind: Service
metadata:
  name: mongoexpress-service
spec:
  selector:
    app: mongo-express
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
      nodePort: 30000
  type: LoadBalancer
```

Figure 5.4: External Service Node IP Service Definition File

```
(geekone@geekone)-[~/kubeProject]
$ cat secret-definition.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  mongo_admin_USER_NAME: cm9vdA==
  mongo_admin_PASSWORD: cm9vdA==
```

Figure 5.5: Secret Yaml Definition File

A terminal window with a dark background and light blue text. The prompt is '(geekone@geekone) - [~/kubeProject]'. The command 'cat configMap-definition.yaml' has been executed. The output is a YAML configuration for a ConfigMap.

```
(geekone@geekone) - [~/kubeProject]
$ cat configMap-definition.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-configmap
data:
  mongodb_url: mongodb-service
```

Figure 5.6: ConfigMap Yaml Definition File

Conclusion

This project opened for us a huge windows to the possible vulnerabilities that can be detected with Wazuh. But this implementation can have more improvements to ameliorate the detection experience especially in production environments. For instance, ngrok is not recommended at all. It's an easy solution for experimentation but it doesn't allow users (agents) to whitelist the Wazuh server.

General Conclusion

To crown all, in this report we worked on the problem of “Securing Kubernetes Cluster On AWS”. We tackled the project mainly from worker node perspective. We focused on securing the frontend tier on the kubernetes Cluster . Through our journey , we tried to find a good visualization of possible threat scenarios.

In order to guarantee the understanding of the required theoretical knowledge, we highlighted a chapter that digs deep in the concepts related to securing containerized Infrastructure On the cloud. At this point, we illustrated the different possible infrastructures that are commonly used. Then we explained Containerization concept as well as the possible solutions that we can use on the selected infrastructure. This phase allowed us to be familiar with . After that, we presented the SIEM that we are going to implement for monitoring and threat detection. As we shed the light on a specific type of SIEM , which is Wazuh.

Secondly, we went through a detailed explanation of the platform that we want to deploy , starting with Kubernetes cluster infrastructure, the multi-tier deployed application , and Wazuh implementation .

Thirdly, We went through a step-by-step installation guide where we built our kubernetes cluster, we deployed a multi-tier application on AWS EKS , we installed Wazuh on-premises and then we connected it to the frontend tier worker node that we want to supervise.

Least but not last, We tried to apply some attack scenarios such as Vulnerability detection, DOS attack and Shellshock attack as well.

While observing the results of the attack scenarios and while proceeding the installation , we have remarked a set of recommendations that we collected in the last chapter entitled Conclusions and perspectives.

However, our solution is considered the first version. Various improvements can be added. On the first hand, We need to get a fixed public IP address that should be allocated to Wazuh Server as well as another fixed IP address for the agents that need to be supervised. On the other hand, Wazuh is still not that efficient in the detection of DOS attacks and it's always recommended to find an external tool dedicated for this kind of attack. Furthermore, we will have the opportunity to strengthen the inspection of Kubernetes cluster while making the Wazuh agent is mainly the pod .

Bibliography

- [1] Ashish Patel, Kubernetes-Architecture Overview
<https://medium.com/devops-mojo/kubernetes-architecture-overview-introduction-to-k8s-architecture-and-understanding-k8s-cluster-components-90e11eb34ccd>, Aug 12, 2021
- [2] Gartner Report About 2021 Magic Quadrant for Cloud Infrastructure & Platform and Services, viewed on 3th April, 2022 : <https://aws.amazon.com/resources/analyst-reports/gartner-mq-cips-2021/>
- [3] Wazuh Official Documentation, Installation Section, viewed on 10th April, 2022 : <https://documentation.wazuh.com/current/installation-guide/>
- [4] AWS EKS documentation , viewed on 10th April , 2022 : <https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html>
- [5] AWS EKS documentation, User Manual section, Wazuh Capabilities in Vulnerability Detection , viewed on 11th April , 2022 : <https://documentation.wazuh.com/current/user-manual/capabilities/vulnerability-detection/how-it-works.html>
- [7] National Security Agency, Cybersecurity and Infrastructure Security Agency, Cybersecurity Technical Report , Kubernetes Hardening Guidance, August 2021