Department of Information Engineering and Computer Science

Master in Computer Science
2020-2021

Final thesis

Analysis and comparison of Log Shipment solutions at AWS S3 for Windows 10.

Francisco Manuel Colmenar Lamas

SUMMARY

A fundamental aspect that every company must address to start building its security infrastructure is visibility. Increasing a company's visibility raises the quality and effectiveness of all other existing security solutions. The objective was to implement an endpoint log forwarding solution for the Windows 10 devices of the company About You. To accomplish the objective, several concepts and knowledge in the scope of log management solutions were studied, as well as the use of AmazonWeb Services (AWS) dedicated to these activities.

After analyzing the different solutions, Kinesis Windows Agent was chosen to implement the endpoint log shipment solution. Because it provides a serverless architecture, where the agent sends logs from the endpoints to Kinesis Firehose. In addition, it does not require any heavy-weight dependencies and its configuration is straightforward.

Also, since Kinesis Firehose is an AWS managed service, there is no need to handle the scaling or fault tolerance issues common in a client-server architecture and it integrates seamlessly with S3.

Regarding the implementation, the code for the installation and maintenance of the Kinesis Windows Agent was mainly developed in Powershell scripts triggered remotely using Ninjarmm. And the AWS infrastructure code required for this project was developed using Terraform. In addition, through Gitlab's CI/CD pipeline, AWS resources are automatically updated if the code is modified.

As a conclusion, the deployment of the Kinesis agent for Windows on all employee Windows devices was a success.

Keywords

- Logs
- Cloud provider
- CI/CD
- Terraform
- Endpoint

CONTENTS

1. INTRODUCTION 2
1.1. Motivation of Work 2
1.2. Goals 3
1.3. Document's structure 4
2. STATE OF THE ART 5
2.1. Log Shipment 5
2.1.1. ELK Stack 5
2.1.2. Fluentd 5
2.1.3. Amazon Kinesis Agent for Microsoft Windows 6
2.2. Cloud providers 6
2.3. Amazon Web Services 7
2.4. Microsoft Azure 7
3. TECHNOLOGIES AND TOOLS 8
3.1. Amazon Web Services 8
3.1.1. Amazon Machine Image (AMI) 8
3.1.2. CloudWatch 8
3.1.3. DynamoDB 9
3.1.4. Elastic Compute Cloud (EC2) 9
3.1.5. Amazon Elasticsearch Service (ES) 9
3.1.6. Identity and Access Management (IAM) 10
3.1.7. Kibana 10
3.1.8. Kinesis Data Firehose 10
3.1.9. Kinesis Data Streams (KDS) 10
3.1.10. Simple Storage Service (S3) 11
3.2. CI/CD 11
3.3. GitLab 11
3.4. GitLab CI/CD 12
3.5. PowerShell 12
3.6. Ninjarmm 12

3.7. Ruby . . . 13
3.8. Security Information And Event Management (SIEM) . . . 13
3.9. Terraform . . . 13
3.10. Visual Studio Code . . . 13
4. LOG SHIPMENT ANALYSIS . . . 14
4.1. Options considered . . . 14
4.1.1. Fluentd . . . 14
4.1.2. Fluent bit . . . 16
4.1.3. Amazon Kinesis Agent for Microsoft Windows . . . 19
4.1.4. Logstash . . . 22
4.1.5. NxLog . . . 24
4.1.6. Windows Event Forwarder . . . 26
4.2. Summary of the analysis . . . 28
4.3. Final Choice . . . 29
5. PROJECT ARCHITECTURE . . . 31
5.1. Log shipment . . . 31
5.2. Remote management of the devices . . . 31
5.3. Update of AWS infrastructure . . . 32
5.4. Update of Amazon Kinesis Agent configuration. . . . 33
5.5. Whole architecture . . . 33
6. PROJECT IMPLEMENTATION . . . 34
6.1. Project Organization . . . 34
6.1.1. Project Structure . . . 34
6.2. GitLab . . . 35
6.2.1. Project Structure . . . 35
6.2.2. gitlab . . . 35
6.2.3. GitLab Pipeline . . . 36
6.3. Setup . . . 37
6.3.1. Project Structure . . . 37
6.3.2. config . . . 37
6.3.3. scripts . . . 38
6.4. Terraform . . . 40
6.4.1. Project Structure . . . 40

6.4.2. environments 41
6.4.3. kinesis 41
6.4.4. tf_backend 41
7. FUTURE DEVELOPMENTS 42
7.1. SIEM analysis and implementation 42
7.2. Lambda for forwarding the logs to the SIEM 42
7.3. Setting alarms at the SIEM depending on the logs 42
7.4. Future developments architecture 42
8. CONCLUSIONS 44
8.1. Fulfilment of the initial objectives 44
8.2. Conclusion of the analysis 44
8.3. Conclusion of the implementation 46
BIBLIOGRAPHY 48

LIST OF FIGURES

2.1 ELK stack architecture. 5
2.2 Fluentd problem reduction. 6
2.3 Amazon Kinesis Agent for Microsoft Windows summary. 6

3.1 Kinesis Data Firehose architecture. 10
3.2 CI/CD architecture. 11
3.3 Gitlab CI/CD architecture. 12

4.1 Fluentd architecture. 14
4.2 Fluent bit architecture. 17
4.3 Kinesis Windows Agent architecture. 20
4.4 Logstash and beats architecture. 23
4.5 Nxlog architecture. 25
4.6 Windows Event Forwarder architecture. 27

5.1 Log shipment architecture. 31
5.2 Remote management of endpoint devices. 32
5.3 Update of AWS infrastructure. 32
5.4 Update of Amazon Kinesis Agent configuration. 33
5.5 Whole architecture for Kinesis Windows Agent. 33

6.1 GitLab pipeline overview (for staging). 36

7.1 Architecture of the future developments. 43

LIST OF TABLES

4.1 Fluentd analysis 16

4.2 Fluent bit analysis 18

4.3 Amazon Kinesis Agent for Microsoft Windows analysis 21

4.4 Logstash analysis 24

4.5 Nxlog analysis 26

4.6 Windows Event Forwarder analysis 28

4.7 Summary of the Analysis 29

8.1 Summary of the Analysis 46

1. INTRODUCTION

This master thesis discusses the experience and knowledge gained through an internship at About You, a company based in Hamburg, Germany. About You is a fashion and technology company focused on digitalizing classic shopping by providing inspiring and personalized shopping experience to each user [1].

The goal of the internship is to implement an endpoint log shipment solution for Windows 10 devices. Therefore, to accomplish the objective of the internship, several concepts and knowledge in the scope of log management solutions have been studied as well as the use of Amazon Web Services (AWS) dedicated to these activities.

The activities performed are therefore part of the broader framework of Information Technology (IT) security, specially from the log aggregation, storage, analysis and Security Information and Event Management (SIEM) capabilities. These activities are the cornerstones of for most IT security designs and infrastructures, making clear their importance is remarkable [2].

In addition to an introduction to the project developed, this chapter will explain the reasons for choosing this project. Besides, in the last section of this chapter, the general structure of this Master Thesis will be outlined.

1.1. Motivation of Work

A fundamental aspect that every company must address to start building its security infrastructure is visibility. No company can correctly design and implement a security posture if it does not know what is happening on their devices and networks. Thus, increasing the visibility of a company's infrastructure is a crucial step upon which the rest of the security measures will be built [3].

Therefore, in the absence of clear visibility into infrastructure devices and networks, it might be known that something is wrong and that there is an incident, but it would be remarkably complex to identify what exactly is wrong. Consequently, the reaction time to stop the spread of a compromise throughout the infrastructure would be significantly longer, thus increasing the probability of a considerable impact.

Furthermore, to identify the damage caused by an attack, it is necessary to analyze the different logs produced. Each log file stores information about its use that, if compared with the rest of the log files, can describe the situation of the system [4].

Log management is crucial to achieving the goal of improving the visibility capabilities of an infrastructure. Log management is a security product that focuses on the collection, storage and analysis of log files. However, in this master's thesis, we will only discuss the collection and storage of logs. And for the log analysis and correlation stages, we will suggest using a SIEM solution described in the future development chapter.

Once the visibility of events has been increased, it is possible to create a more accurate design and implementation of security measures that addresses potential threats. Therefore, increasing visibility of an enterprise elevates the quality and effectiveness of all other existing security solutions [5].

SIEM is a security management tool that combines the functionalities of Security Information Management (SIM) and Security Event Management (SEM) [6]. The main features of a SIEM are the ability to analyze the

received data, such as endpoint and network logs, and to be able to raise alarms detecting security incidents [7].

As a consequence of the Covid-19 pandemic, there has been a considerable shift from office-based work to telecommuting, and this will continue to occur in the near future. In advanced economies, between 20% and 25% of the workforce could now work from home three to five days a week. Increasing telecommuting by a factor of four to five compared to before the pandemic. Moreover, some companies are considering a permanent shift to flexible workspaces because of the positive aspects of teleworking discovered during the pandemic [8].

The main security consequence of this situation is the shift from a more controlled environment, as is the office with its monitored networks, to an uncontrolled environment, as is remote working. Therefore, a company's infrastructure has changed dramatically from having most devices under the same network to have a set of endpoint devices, such as employee laptops, located on very different networks that cannot be treated as secure.

Consequently, the need for enterprises to have a strong stance on endpoint log management is crucial in the current and future scenarios. The fact that remote working will be the norm in the future creates the need for considerable visibility into endpoint devices, specially when they are not on the enterprise network.

1.2. Goals

This Master's Thesis aims to analyze the log management options available on the market and choose the best solution to install on the company's Windows 10 endpoint devices.

The following requirements must be met to achieve the aforementioned objective:

- *Analysis of the different log management solutions for Windows 10 devices:* An analysis to the most prevalent log management solutions will be performed considering the use case of a Windows 10 employee laptop to ideally send the logs to S3 for storage.

- *Election of the solution to be implemented:* Once the analysis of the different solutions has been accomplished, it is necessary to choose the solution to implement. Furthermore, the solution chosen must take into consideration the use case explained above and the company's infrastructure.

- *Proof of Concept implementation of the chosen solution:* After choosing the solution to be used, a PoC should be accomplished to have a working version of the project, in which we can develop further improvements.

- *Solution's infrastructure management automation:* To facilitate the management of the solution, the cloud infrastructure required for this purpose must be capable of providing automation functions for its update.

- *Remote management capabilities integration:* The developed solution has to support remote management approaches to enable remote installation and maintenance of the solution on employee laptops.

- *Implementation of the solution on employee laptops:* The chosen solution needs to be implemented on all Windows 10 employee devices to achieve maximum usability.

1.3. Document's structure

The aim of this section is to provide a guideline of the structure this document will follow. This document is divided into the following chapters.

- *Introduction and goals*: This chapter will describe the current situation of log management solutions, providing an overview of the importance that this field has achieved in recent years and will continue to have in the future.

  Additionally, the objective of this Bachelor Thesis will be discussed to provide an overview of the document.

- *State of the Art*: This chapter aims to give a technical description of the background of this project.

- *Technologies and tools*: This section describes the different technologies and tools used in this project for a better understanding of the following sections.

- *Log Shipment analysis*: This chapter explains the analysis of the different log shipment solutions.

- *Project Architecture*: The architecture of the chosen solution for sending logs from Windows 10 devices is developed, thus facilitating the understanding of the overall solution.

- *Project Implementation*: This chapter describes the different scripts developed for the implementation of the Kinesis Windows Agent for Windows.

- *Future developments*: The purpose of this chapter is to describe three distinct future developments for the project that will continue the security approach of using endpoint logs to detect malicious activity.

- *Conclusions*: This chapter presents and explains all the conclusions and ideas achieved from the development of this project. Furthermore, the goals proposed for this Master Thesis will be analyzed to determine the fulfilment of the perspectives of this project.

2. STATE OF THE ART

This chapter explains the current state of the art of log shipment and cloud providers.

2.1. Log Shipment

The following is a brief description of the most advanced and widely used log shipment solutions. A more extensive explanation of their characteristics will be provided in the following chapters.

2.1.1. ELK Stack

The ELK stack is composed by Elasticsearch, Logstash and Kibana, and it is provided by Elastic.io as open-source products. ELK is among the most popular log shipment stacks focused on endpoint devices.

In addition to the main components from ELK, beats is fundamental element of the stack. Beats are the lightweight log shippers which can be installed on endpoint devices of various operating systems. They retrieve different types of information and forward it to Elasticsearch or Logstash.

The following diagram provides an overview of the ELK architecture.

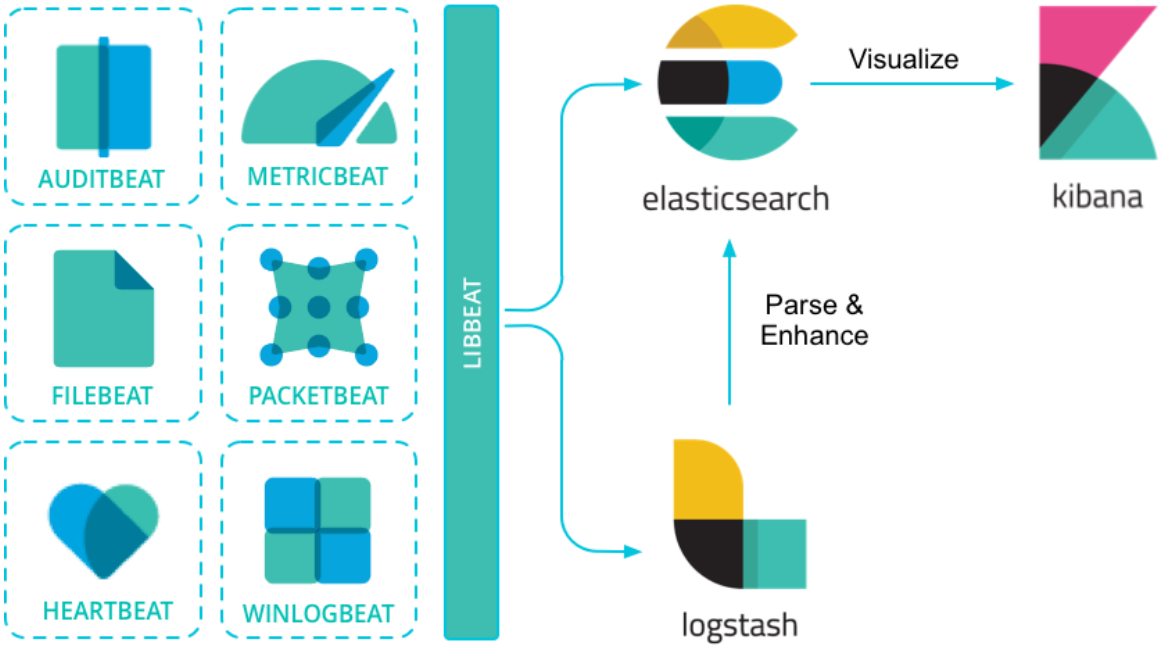

Fig. 2.1. ELK stack architecture.

2.1.2. Fluentd

Fluentd is an open-source data collector. The core idea behind Fluentd is to unify the logging layer. Consequently, an initial problem of the order of M x N, M systems shipping data to N locations with N different paths,

could be reduced to an N + M problem.

Fluentd is an open-source project which includes over 500 different plugins, such as S3 or Elasticsearch plugins. Moreover, it can be installed on Windows and Linux devices, thus not being constrained to a specific operating system.

The following diagram depicts the problem that Fluentd solves.

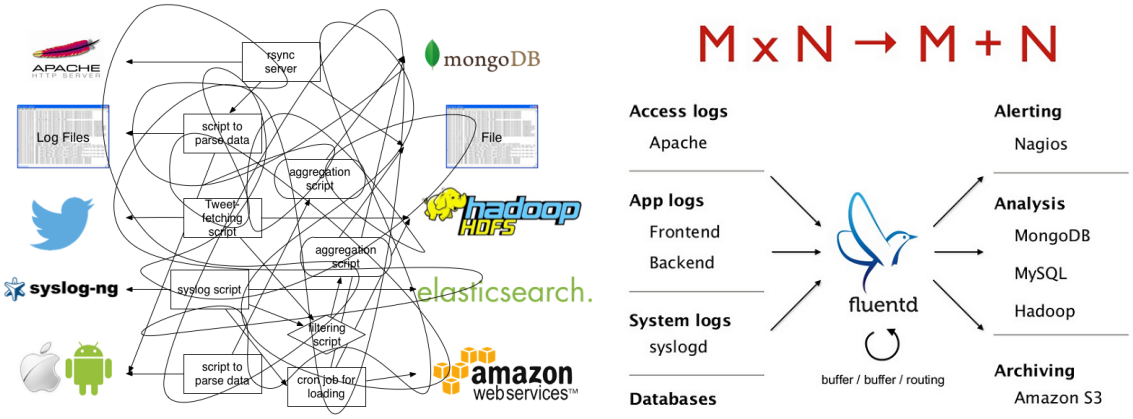

Fig. 2.2. Fluentd problem reduction.

2.1.3. Amazon Kinesis Agent for Microsoft Windows

Amazon Kinesis Agent for Microsoft Windows is the log shipment solution from Amazon Web Services (AWS). As the name suggests, it solely focuses on Microsoft Windows operating system.

It provides native interoperability with other AWS services, such as Kinesis Firehose or Cloudwatch. Similarly with the previous solutions, Amazon Kinesis Agent for Microsoft Windows is open-source. However, it does not have a wide variety of plugins available.

The following diagram displays and overview of the Amazon Kinesis Agent for Microsoft Windows architecture.

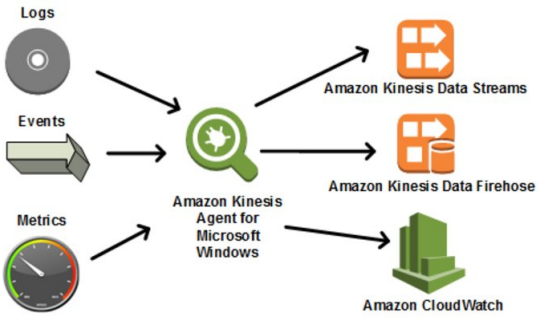

Fig. 2.3. Amazon Kinesis Agent for Microsoft Windows summary.

2.2. Cloud providers

The two fundamental and forefront cloud providers are going to be described below.

2.3. Amazon Web Services

Amazon Web Services, or AWS, is the dominant cloud service provider worldwide. It has a 47.8% market share compared to Microsoft's 15.5% in 2018 [9]. Consequently, the log shipment solutions have AWS plugins to integrate with them.

AWS provides a wide selection of services, such as data storage or log monitoring [10]. Additionally, Amazon Web Services security capabilities and services greatly in meeting high-security standards for security teams of different sizes [11].

AWS is the cloud provider used for this project. Therefore, a more thorough description of it and its services is provided in the following chapters.

2.4. Microsoft Azure

Microsoft Azure is the second-largest cloud provider by market share.

The main differentiating feature of Microsoft Azure is its full integration with Office 365 and Active Directory. Furthermore, if an existing Windows Server or SQL Server license is migrated to Azure, the fee payable for Microsoft Azure is reduced. Thus, it facilitates the shift to the cloud for companies heavily relaying on Microsoft products [12].

Therefore, Azure dominates in Hybrid Cloud Organizations that integrates onsite servers with Cloud instances. However, Microsoft has limited Linux options [13].

3. TECHNOLOGIES AND TOOLS

This chapter will describe the different technologies and tools used throughout the development of the analysis and implementation to help understand the subsequent sections.

3.1. Amazon Web Services

Amazon Web Services, or AWS, is the leading cloud service provider worldwide with a market share in 2018 of 47.8% compared with the 15.5% of Microsoft, the second largest company in the market [9].

AWS offers a wide variety of services, from blockchain application, support for quantum technologies to simply data storage [10].

Amazon Web Services provides several advantages to its customers compared to on-premise hardware and software infrastructure, such as the availability to adjust its resources to business needs, enhancing scalability and cost savings by reducing the number of employees required to deploy and maintain the IT infrastructure. In addition, its security capabilities and services greatly facilitate to ensure high-security standards without the need of a large security team [11].

All the definitions of the services or tools used in this Master Thesis related to Amazon Web Services are explained below for easy reference and understanding.

3.1.1. Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) contains all the information necessary to launch an AWS instance, which is an EC2 (see 3.1.4). Therefore, it is strictly necessary that when an EC2 is launched, an AMI is attached to it, determining the specifications of the system to be created [14].

Furthermore, an Amazon Machine Image is not bounded only to one instance. There can be as many instances as desired launched with a specific AMI. Thus, allowing to reuse of the same configuration between different devices.

In addition, there are freely available AMIs in AWS that can be used to directly launch EC2s with those specifications, avoiding device setup time.

On the other hand, it is also possible to create an Amazon Machine Image specific to the system you want to use which will be explained in the implementation section.

3.1.2. CloudWatch

Amazon CloudWatch is a service focused on implementing monitor and observation capabilities providing the user with data from different applications. As a result, it increases the possibility to respond to system performance changes, optimize resource utilization, and detect specific security incidents by having a unified place where these metrics and data are displayed [15].

CloudWatch receives data in the form of logs, metrics and events and allows the user to trigger certain actions, such as stopping an EC2 instance or getting automatic email notifications when a specific log is received.

However, even though there is the ability to ingest data into CloudWatch from on-premise devices via the CloudWatch Agent or the API, it is mainly focused on AWS services thanks to the easy integration of CloudWatch with AWS services [16] [17].

3.1.3. DynamoDB

Amazon DynamoDB is a key-value and document database, or NoSQL database, fully managed, multi-region, with built-in security and backup and restore capabilities [18].

Furthermore, DynamoDB provides flexible pricing, together with a stateless connection model with consistent response time independently of the database size [19].

One of DynamoDB's key aspects is its high and constant throughput levels due to its tables are replicated across the different AWS regions. Thus, providing its users with local access to global applications.

Furthermore, DynamoDB supports ACID transactions providing the possibility to integrate the service in business-critical applications. ACID stands for atomicity, consistency, isolation, and durability. They are the four main properties which a critical transaction must comply with [20].

3.1.4. Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud, or EC2, is an AWS service that provides secure and resizable compute capacity in Amazon's cloud. In addition, EC2 can easily integrate with other AWS services facilitating its adoption on cloud infrastructures [21].

It is designed to facilitate web-scale cloud computing easier by providing developers with the possibility to create easily configurable virtual machines instances with escalating capabilities [22].

Furthermore, EC2 allows to choose the processor, storage, networking, operating system, and purchase model of the computing service. Hence, delivering considerable flexibility to the developers and architects when designing the infrastructure to be used.

3.1.5. Amazon Elasticsearch Service (ES)

Amazon Elasticsearch Service, or ES, allows to deploy, secure and run Elasticsearch effortless and scalably. It integrates with different AWS services, including AWS Kibana Service, see 3.1.7, supporting the visualization of the data from ES while avoiding the operational overhead of installing and managing the whole infrastructure [23].

Elasticsearch is a distributed search and analytics engine which accepts any data, wheter textual, numerical, structured, and unstructured. Data comes to Elasticsearch as raw data flows from different sources, including logs, metrics, and web applications requests and responses. Then, this data is indexed in Elasticsearch allowing the user to run queries against these data which could then be visualized with Kibana [24].

However, Elastic.io, the company behind ES and Kibana, decided to change the license of these products from the Apache License, Version 2.0 (ALv2) to Elastic License or the Server Side Public License. As a consequence, these two products are not going to be open-source anymore. Therefore, Elastic.io version 7.11 is going to provide a different set of features than Amazon Elasticsearch Service, which decided to fork the original projects to continue them as open-source [25] [26].

3.1.6. Identity and Access Management (IAM)

AWS Identity and Access Management, as known as IAM, provides the ability to define access management policy for different AWS services securely and flexibly. IAM allows to create users, groups and roles, which can be assigned different permissions. Therefore, it allows achieving the specific desired permissions that each user needs to access AWS services [27].

3.1.7. Kibana

Kibana is a data visualization and exploration tool on top of the indexed content belonging to an Elasticsearch cluster. Consequently, Kibana is often the choice for visualizing data stored in Elasticsearch [28].

Kibana's use cases are diverse, ranging from log analytics to web application monitoring and operational intelligence. It offers the possibility to create heat maps, identify geospatial data, histograms and line graphs, among others [29].

Furthermore, Kibana allows establishing alarms depending on the result of a clearly defined query against the data stored at Elasticsearch. Thus, the capabilities which Elasticsearch and Kibana provide from a security point of view could be similar to a SIEM, see 3.8, providing log aggregation, visibility of the logs stored and alerting if a certain threshold is exceeded. However, it lacks incident management capabilities [30].

3.1.8. Kinesis Data Firehose

Amazon Kinesis Data Firehose is an AWS managed service that enables to capture, transform, and deliver streaming data to Amazon S3, Amazon Elasticsearch or HTTP endpoints, among others.

Because it is a fully managed service from AWS, it automatically handles its scaling caused by spikes of throughput, thus removing the complexity of end-user infrastructure. In addition, it provides the possibility of encrypting and compressing the data as well as providing near real-time data delivery, with one minute being the lowest buffer time [31].

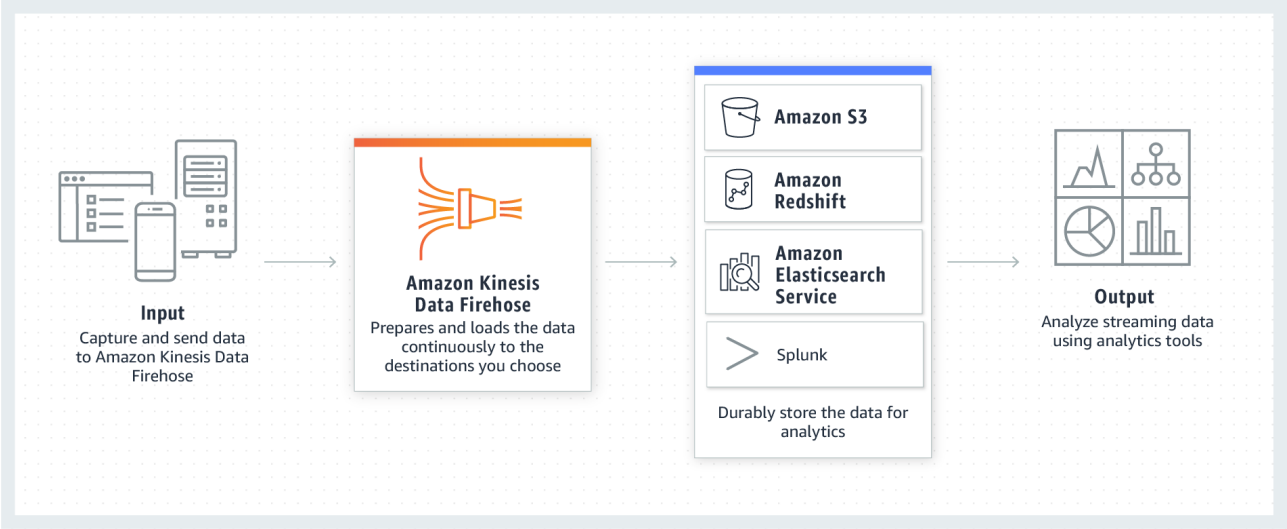

Fig. 3.1. Kinesis Data Firehose architecture.

3.1.9. Kinesis Data Streams (KDS)

Amazon Kinesis Data Streams, also known as KDS, is a service similar to *Kinesis Data Firehose* but with some significant differences.

Kinesis Data Firehose and *Kinesis Data Streams* capture, transform and deliver streaming data to different services. However, in this case, the delivery is in real-time and not near real-time. Therefore, *Kinesis Data Streams* are more suitable for time critical services and applications.

Additionally, unlike *Kinesis Data Firehose*, *Kinesis Data Streams* does not automatically handle throughput scaling [32].

3.1.10. Simple Storage Service (S3)

Amazon Simple Storage Service, or S3, is an AWS service that provides object storage with managed scalability, availability, security and performance.

Amazon Simple Storage Service can be used for a wide range of purposes, such as hosting a website, storing logs, saving backups, and more.

Furthermore, it provides the ability to manage access control to the data stored in it and fulfill compliance requirements [33].

3.2. CI/CD

Continuous Integration, or CI, focuses on pushing small pieces of code to the shared codebase hosted in a Git repository. Every push activates some defined scripts to build, test and validate the newly created code, previously to merge it to the shared codebase.

On the other hand, Continuous Delivery, or CD, aims at including all the new pushed changes to the production version of the repository.

Due to the use of CI/CD methodologies, the capability to find bugs and errors early in the development process, plus ensuring code compliance with the code standards established for the application, considerably increases the efficiency of the development process [34].

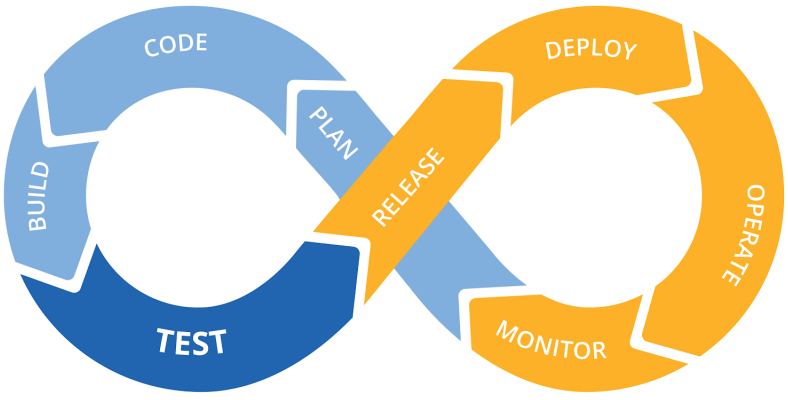

Fig. 3.2. CI/CD architecture.

3.3. GitLab

GitLab is a DevOps platform offered as a web-based application that provides Continuous Delivery, Continuous Integration, Auto DevOps, SAST, DAST and Source Code Management, among other services[35].

3.4. GitLab CI/CD

GitLab CI service, or Continuous Integration, allows the developer to build and test the software when pushed to the repository.

GitLab CD, or Continuous Deployment, on the other hand, provides the possibility to add the new code changes directly to production. Consequently, it allows deploying to production every day if desired [36].

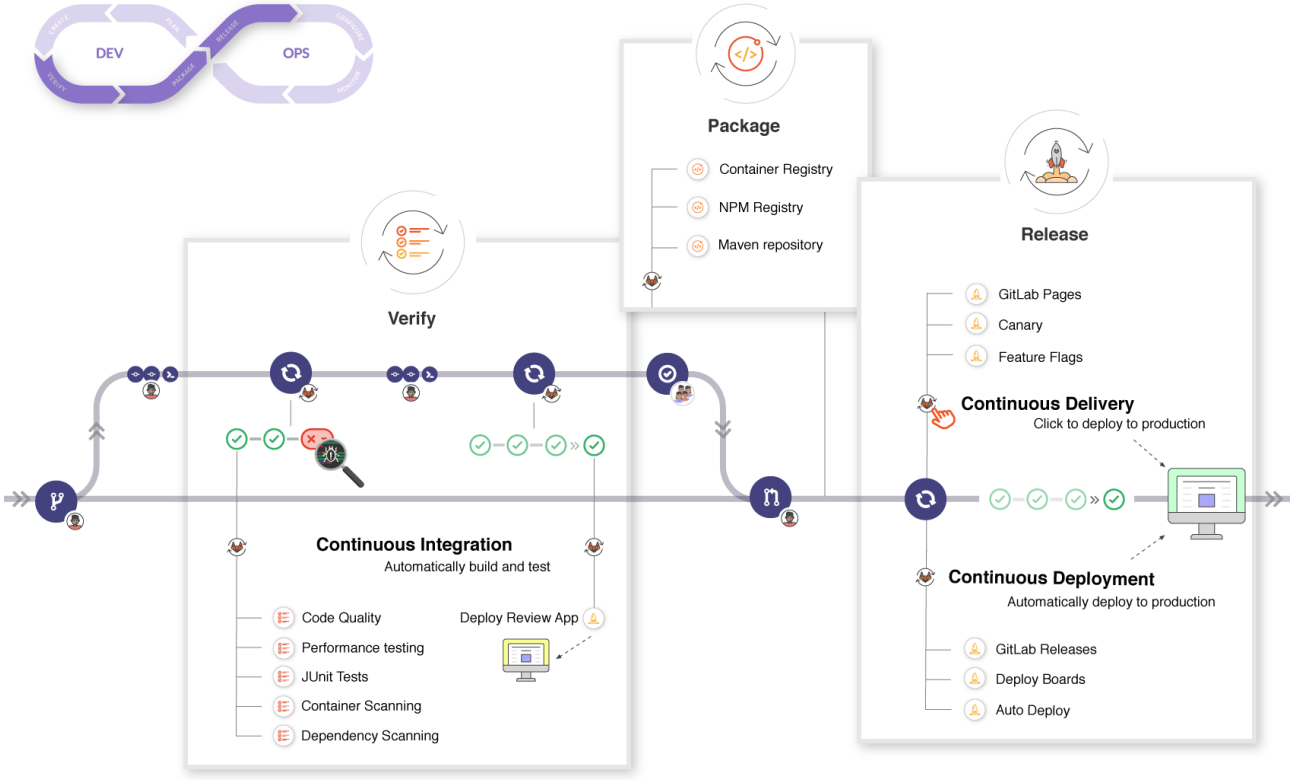

Fig. 3.3. Gitlab CI/CD architecture.

3.5. PowerShell

PowerShell is a cross-platform task automation tool comprising a command-line shell, scripting language, and configuration management framework.

PowerShell runs on Windows, Linux, and macOS. Furthermore, it returns not only text but also .NET objects [37].

3.6. Ninjarmm

NinjaRMM is a *Remote Monitoring and Management* tool that provides companies with the opportunity to dispatch centralized IT services across their infrastructures with only one tool [38].

Additionally, NinjaRMM allows automatizing routine tasks, such as on-schedule script running or as a consequence of performance thresholds.

3.7. Ruby

Ruby is an open-source programming language focused on providing simplicity and productivity to software developers [39].

Designed by Yukihiro Matsumoto in Japan in the mid-1990s is a high-level, general-purpose supporting multiple programming paradigms such as object-oriented language [40].

3.8. Security Information And Event Management (SIEM)

Security Information And Event Management, or SIEM, is a security management tool combining SIM (security information management) and SEM (security event management) functionalities [6].

The core features of a SIEM are log event collection and management from different types of sources, the capability to analyze the received data, such as event logs or general-purpose logs, and be able to raise alarms and to detect when a certain detection rule is satisfied or a threshold is reached [7].

3.9. Terraform

The objective of Terraform is to build, change, and version infrastructure safely and efficiently, specially cloud infrastructure. Thus, Terraform grants support to the major cloud providers, such as AWS, Azure, Google Cloud Platform and Alibaba Cloud [41].

Terraform provides a high-level configuration syntax allowing the reuse and versioning of Terraform files similarly to other programming languages code development. As a consequence, the sharing and cooperation on infrastructure management are considerably increased [42].

In addition to this, Terraform is one of the leading DevOps tools for managing cloud resources, and thus it is used by leading companies on the technology market such as Uber, Slack or Udemy, among others [43].

3.10. Visual Studio Code

Visual Studio Code is a free coding editor that allows to code in any programming language, such as Python, Java, C++ or JavaScript, without switching editors.

It provides debugging support, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git, among other features. On top of this, users can install or create extensions to further extend their capabilities [44].

Visual Studio Code is developed by Microsoft for Windows, Linux and macOS and is available for free. Its code can be found at GitHub at the repository *Microsoft/vscode* [45].

Furthermore, Visual Studio Code was ranked in the Stack Overflow 2019 Developer Survey as the most popular developer environment tool, with 50.7% of all participants reporting using it [46].

4. LOG SHIPMENT ANALYSIS

4.1. Options considered

This section will describe the different options considered for log shipment from Windows 10 devices to a storage destination, preferably S3.

In addition, we will provide the advantages and disadvantages of each of the assessed solutions along with a summary table that complies them.

4.1.1. Fluentd

Fluentd is an open-source data collector that aims to be the solution that unifies the logging layer. Consequently, it is possible to decouple data sources from backend systems by placing Fluentd between them as a pipeline that forwards the data [47].

The Fluentd architecture is an agent architecture. Fluentd can forward logs from an endpoint device, such as a laptop or a server, directly to the final destination, like S3, without the need for an intermediary server. Furthermore, Fluentd is written in C and Ruby, intending to reduce the number of system resources required to run it effectively [48].

Fluentd is an open-source project under the Apache 2.0 License. It includes over 500 different plugins available, as well as the ability to create custom plugins. Consequently, Fluentd has a wide community around it with industry adoption , with industry leaders such as Microsoft and Atlassian using Fluentd [49].

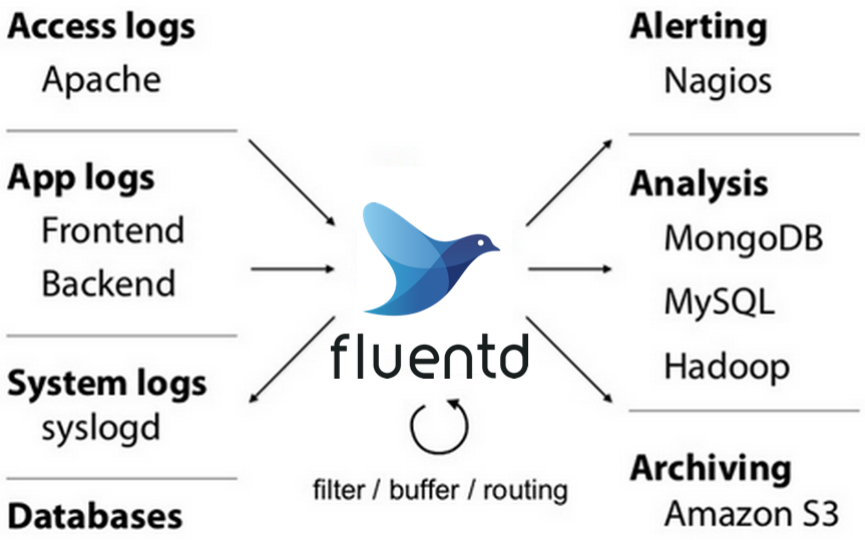

Fig. 4.1. Fluentd architecture.

4.1.1.1 Advantages

The advantages of using *Fluentd* as the log shipment option are the following:

- *Linux installation:* It is possible to install Fluentd on Debian/Ubuntu, Red Hat Linux and MacOS. Al-

though the project is focused on Windows devices, the possibility to use the same tool for Windows and Linux Operating Systems is an important point to consider, making the migration of the log shipping solution across the entire infrastructure simpler and smother [50].

- *Serverless architecture:* Fluentd can send logs directly from the endpoint devices to the final destination, like S3, without the need to use a server to do so. Consequently, the infrastructure is simpler and easier to manage.

- *High-quality documentation:* The available documentation for Fluentd is well structured and straightforward to refer easing the adoption of the solution [51].

- *Engaged and active community:* As a consequence of being an open-source tool and its considerable popularity, it has a large community around it that develops additional plugins and writes blogs and tutorials about Fluentd.

- *Monitoring specific files:* There is the possibility to determine the particular files that Fluentd agent will monitor using the *tail* option. In case the monitored files are modified, they will be forwarded to the defined destination. Therefore, this option increases the solution's flexibility to adapt it to different possible use cases, such as monitoring log files of various tools installed on the endpoint devices using only Fluentd for this purpose [52].

- *S3 integration:* A plugin is available a plugin to forward data to S3 directly from the endpoint device [53].

- *Elasticsearch integration:* Additionally, a plugin for sending data from Fluentd's agent to Elasticsearch is also available, see 3.1.5 [54].

- *Windows Event Logs plugin:* There is a plugin to read Windows Event Logs from the device to send them to the defined destination, called *windows_eventlog*, this plugin could be considered as an out of the box solution for Windows Event Logs shipment [55].

- *Log filtering capabilities:* Fluentd provides eight different types of filters to apply to the received logs allowing high flexibility in the choice of the collected logs. Furthermore, it provides data enrichment capabilities.

4.1.1.2 Disadvantages

The disadvantages of using *Fluentd* for shipping logs from Windows 10 devices are the following:

- *Ruby is necessary:* To run Fluentd at a Windows 10 device Ruby is required to be installed together with S3 or Elasticsearch plugins, see 3.7. This requirement would affect performance and highly increase the complexity of installing and maintaining the solution, specially on user-managed devices such as on employee laptops [56].

- *Impossibility to monitor evtx files:* The only way to monitor evtx files, which are files that store the Windows Events Logs, is through the *windows_eventlog* plugin. However, it is not possible to monitor evtx files with the *tail* option, which is the option used to monitor specific files, such as .config or .txt files. Therefore, monitoring specific files and Windows Event Logs at the same time could be complex.

4.1.1.3 Summary

A summary of the advantages and disadvantages of *Fluentd* is displayed in the table below:

Fluentd Analysis	
Advantages	Linux installation
	Serverless architecture
	High-quality documentation
	Engaged and active community
	Monitor specific files
	S3 integration
	Elasticsearch integration
	Windows Event Logs Plugin
	Log filtering capabilities
Disadvantages	Ruby is necessary
	Impossibility to monitor evtx files

Table 4.1: Fluentd analysis

4.1.2. Fluent bit

Fluent bit is an even lighter version of Fluentd. Fluent bit is a subproject from Fluentd, thus being also open-source. Therefore, Fluent bit is less mature than Fluentd.

The architecture of Fluent bit is similar to Fluend's architecture. Fluent bit can operate as an agent which ships the logs directly to the final destination, like Elasticsearch.

Nonetheless, Fluent bit can use Fluentd as a server in a client-server architecture. Following this approach, Fluent bit's agent installed on the endpoint device will forward the logs to Fluentd's system, which behaves as a server, aggregating and modifying the data and then shipping it to the final destination.

Fluent bit's design aims to increase its performance, achieving a high throughput with low CPU and memory usage. Consequently, Fluent bit is focused on resource-poor containerized environments.

As opposed to Fluentd, Fluent bit is just written in C language with a similar plugable architecture. Fluent bit counts with more than 70 extensions, in contrast with the more than 500 belonging to Fluentd [57].

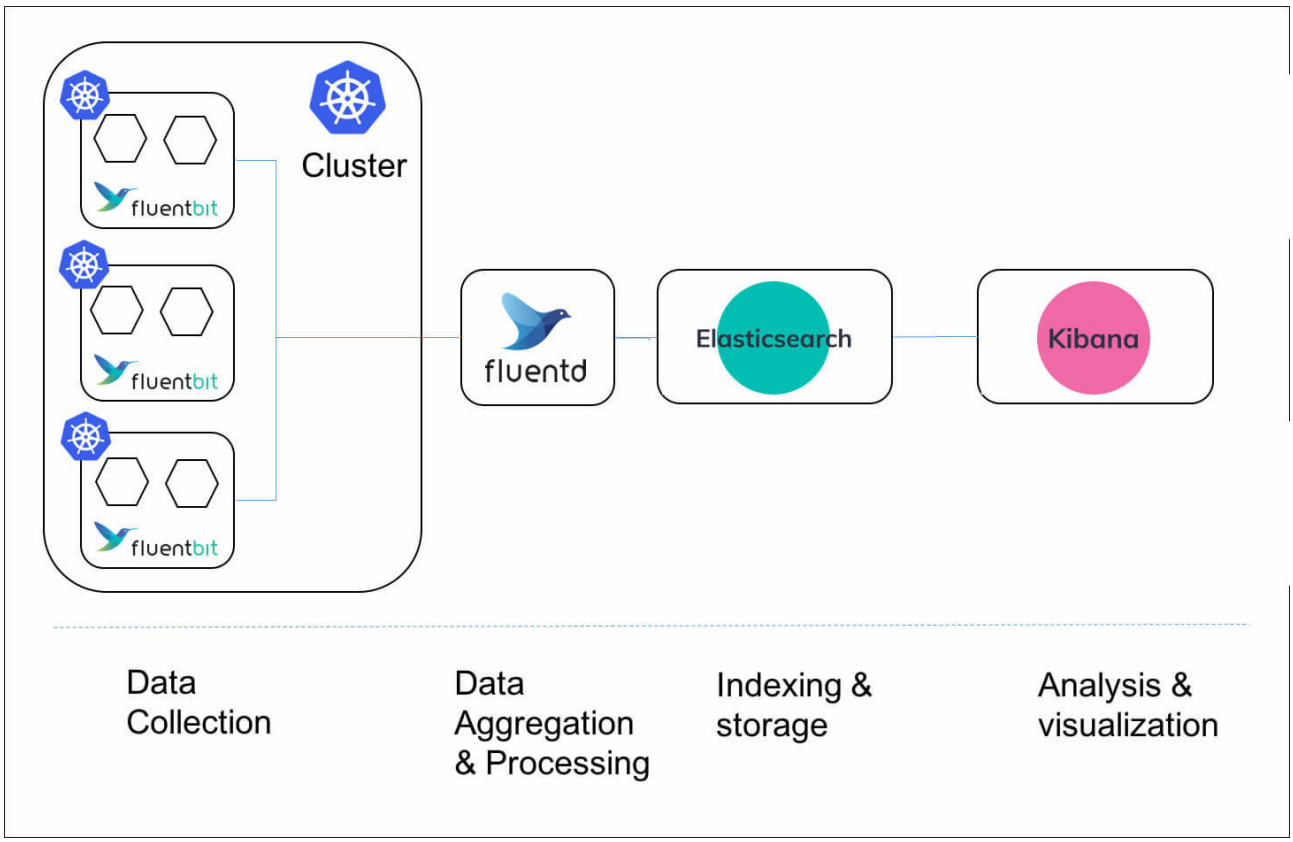

Fig. 4.2. Fluent bit architecture.

4.1.2.1 Advantages

The advantages of using *Fluent bit* as the log shipment option are the following:

- *Linux and container installation:* In addition to the Operating Systems in which Fluentd is available, Fluent bit can be installed at Kubernetess, Docker containers, AWS containers, Raspbian, Embedded Linux and FreeBSD.

- *Serverless architecture:* Fluent bit can be arranged to have a serverless architecture. Therefore, the agent installed on the endpoint device can send logs directly to the destination without needing a server system. However, not all destinations available in Fluentd have this option in Fluent bit. If this is the case, a client-server architecture will be necessary.

- *High-quality documentation:* As a consequence of being a subproject of Fluentd, Fluent bit's documentation follows its documentation's structure easing its reference.

- *Monitor specific files:* As in Fluentd, it is possible to monitor specific files using the *tail* option.

- *Elasticsearch integration:* There is an output plugin available for forwarding data into Elasticsearch directly.

- *Windows Event Logs Plugin:* Fluent bit includes an input plugin called *winlog* that reads Windows Event Logs from the defined channels [58]

- *Log filtering capabilities:* Fluent bit provides the option to filter the received logs using different filter options.

- *Improved performance compared to Fluentd:* The aim of Fluent bit is performance and lightness, thus reducing the downtime perceived by the device user.

4.1.2.2 Disadvantages

The disadvantages of using *Fluent bit* for shipping logs from Windows 10 devices are the following:

- *S3 plugin for Windows not fully supported:* At the time of the Fluent bit analysis the latest version of Fluent bit was 1.7. In this version, the S3 plugin for Windows 10 devices was released. However, there were some bugs and issues, see [59] and [60], which were solved at version 1.7.3. Nevertheless, as can be observed, the maturity of the plugin is not sufficient to include it in a production environment due to the uncertainty of its reliability.

- *Less mature project:* As demonstrated with the S3 plugin, even though Fluent bit is a project with excellent development, it may not be as mature as it should be to be placed in a company's production environment.

- *Impossibility to monitor evtx files:* Similarly to Fluentd, Fluent bit cannot monitor defined evtx files using the *tail* option.

4.1.2.3 Summary

The summary of the advantages and disadvantages of *Fluent bit* is displayed in the following table:

Fluent bit Analysis	
Advantages	Linux and container installation
	Serverless architecture
	High-quality documentation
	Monitor specific files
	Elasticsearch integration
	Windows Event Logs Plugin
	Improved performance compared to Fluentd
	Log filtering capabilities
Disadvantages	S3 plugin for Windows not fully supported
	Less mature project
	Impossibility to monitor evtx files

Table 4.2: Fluent bit analysis

4.1.3. Amazon Kinesis Agent for Microsoft Windows

Amazon Kinesis Agent for Microsoft Windows, or Kinesis Windows Agent, is an agent focused on the shipment of logs from Windows devices, such as desktops or server, to AWS services, such as S3 or, Cloudwatch, see 3.1.2.

Amazon Kinesis Agent for Windows gathers, parses, modify, and ship logs and events from the Windows device to its destination. However, if it is desired to keep the agent as light as possible, the modification could be avoided focusing only on the shipment of the collected logs.

The infrastructure of Kinesis Windows Agent is a serverless agent-based approach. Therefore, there is no need for a server to operate to forward the logs to their final destination.

However, Kinesis Firehose service from AWS is used as a server forwarder of logs. Nonetheless, as it is a service managed by AWS, there is no need to supervise its load balancing or fault tolerance capabilities, thus removing most of the complexity of a log forwarder server. Consequently, in practice, the infrastructure complexity and its approach are similar to a serverless solution.

The possible destination for the shipped logs is Kinesis Data Streams, Kinesis Data Firehose, Amazon CloudWatch, and CloudWatch Logs, among others.

Furthermore, Kinesis Windows agent requires minimal set-up and maintenance, including auto-update options, hence providing an out of the box solution for Windows 10 log shipment [61].

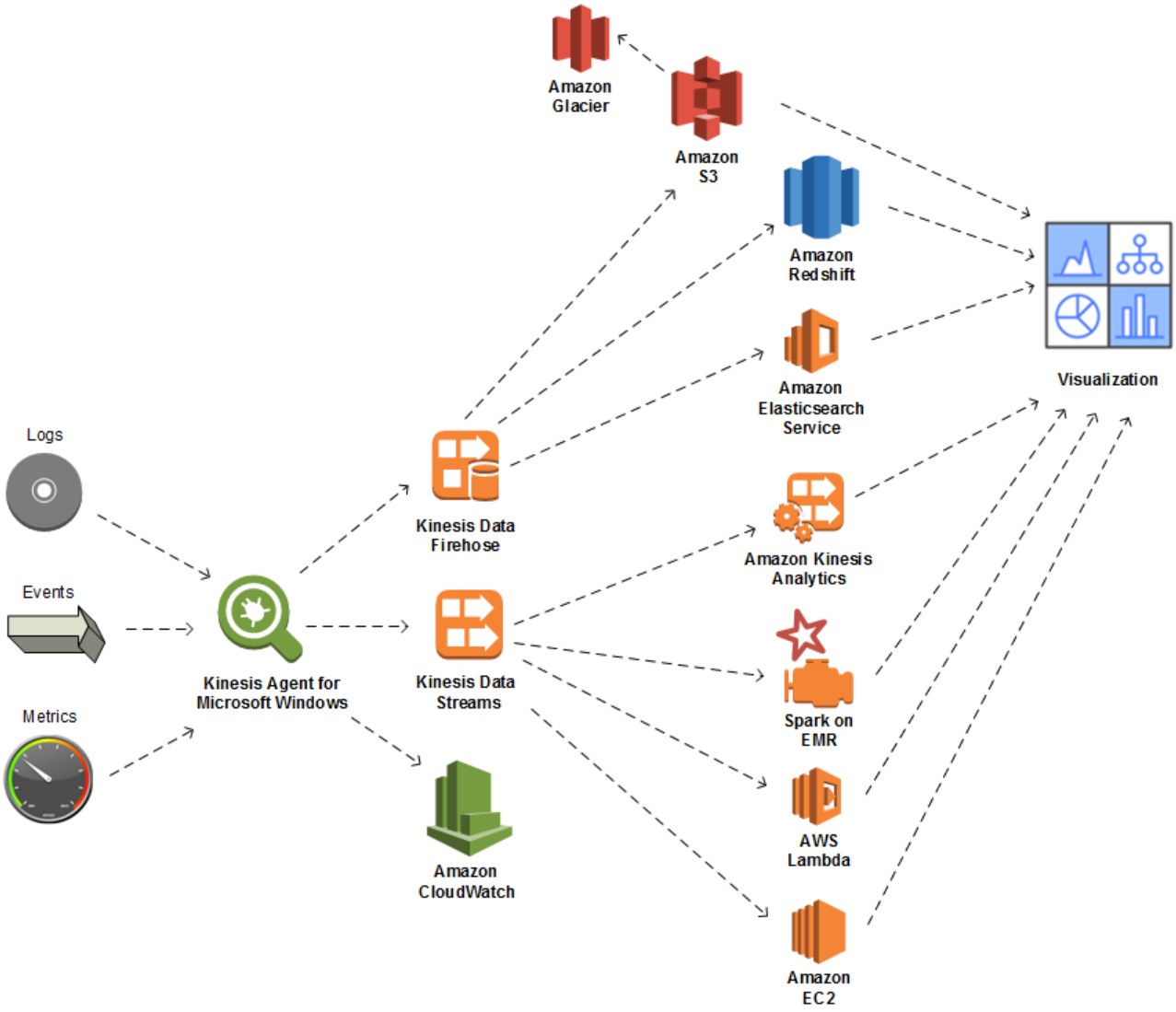

Fig. 4.3. Kinesis Windows Agent architecture.

4.1.3.1 Advantages

The advantages of using *Amazon Kinesis Agent for Windows* as the log shipment option are the following:

- *Serverless architecture:* Windows Kinesis agent follows a serverless architecture in which the logs are forward to Kinesis Firehose, which behaves as a server. However, as it is a managed service from AWS, the complexity regarding load balancing, scaling and fault tolerance is avoided.

- *High-quality documentation:* The documentation from Kinesis Windows agent follows the same structure as the rest of AWS services. Therefore, it is intuitive to navigate and its content covers all the necessary aspects to implement the solution successfully.

- *Monitor specific files:* Kinesis Windows agent provides the possibility monitoring specific files. Additionally, it allows to monitor all the files located at a defined folder, applying filtering to its files' name [62].

- *S3 integration:* It is possible to define that the data coming from the agents at Kinesis Firehose is redirected into S3.

- *Elasticsearch integration:* Additionally, at Kinesis Firehose, it is also possible to determine whether to send the data to the AWS Elasticsearch service.

- *Native support for AWS:* As a consequence of being a product developed by AWS, it has a native and direct integration with different AWS services, such as *Cloudwatch* and *Kinesis Firehose* [63].

- *Windows Event Logs native:* Kinesis Windows agent is a product tailored to read Windows Event Logs.

- *No heavyweight dependencies:* The only dependency for the Kinesis Windows agent to run is to have the *.NET Framework* installed, which is installed by default at Windows 10 [64].

4.1.3.2 Disadvantages

The disadvantages of using *Amazon Kinesis Agent for Windows* to ship logs from Windows 10 devices are the following:

- *Limited Linux OS compatibility:* The Kinesis agent can only be installed on Amazon Linux AMI and Red Hat Enterprise Linux. As a consequence, its adoption across the whole infrastructure is challenging [65].

- *Absence of log filtering capabilities:* The approach provided by the Kinesis Windows agent to filter logs is not efficient due to its inflexibility and steep learning curve of use.

4.1.3.3 Summary

The summary of the advantages and disadvantages of the *Amazon Kinesis Agent for Microsoft Windows* is displayed in the following table:

Amazon Kinesis Agent for Microsoft Windows Analysis	
Advantages	Serverless architecture
	High-quality documentation
	Monitor specific files
	S3 integration
	Elasticsearch integration
	Native support for AWS
	Windows Event Logs native
	No heavyweight dependencies
Disadvantages	Limited Linux OS compatibility
	Absence of log filtering capabilities

Table 4.3: Amazon Kinesis Agent for Microsoft Windows analysis

4.1.4. Logstash

Logstash is the native log shipment solution from Elastic.io, the company behind Elasticsearch and Kibana, see 3.1.7. Additionally, Logstash is part of the ELK stack (Elasticsearch, Logstash and Kibana), one of the most popular and widely used stacks for log aggregation and analytics [66].

Logstash is an open server-side data processing pipeline. It allows the ingestion of data from a wide range of sources by shipping them in different destinations, such as Elasticsearch, S3 or Datadog, among others [67].

The main difference between Logstash and other similar products is its ability to dynamically normalize and aggregate the collected data. As a consequence, the data can be easily enriched and cleaned, facilitating its further analysis [68].

However, Logstash is the heavyweight component of the Elastic.io log shipment solution. Logstash is recommended to be located on a different device and not on each endpoint device. Therefore, it would behave as a server receiving the data from the endpoints and routing it to the final destination.

The lightweight component of the Elastic.io solution is called beats [69].

4.1.4.1 Beats

Beats are the Lightweight data shippers for the ELK stack. They send data to Logstash or Elasticsearch from endpoint devices. They can be installed on servers, containers or the employees laptops due to their low processing load [70].

Different types of beats are available, with Filebeat and Winlogbeat being the most popular.

Filebeat allows to monitoring and sending different types of files to Logstash. Thus, Filebeat enables to easily centralize various formats logs, enhancing their subsequent analysis [71].

On the other hand, **Winlogbeat** focuses on shipping Windows event logs. Consequently, gaining visibility of Windows devices with Winlogbeat is simple and can be easily integrated with a more complex stack using Logstash [72].

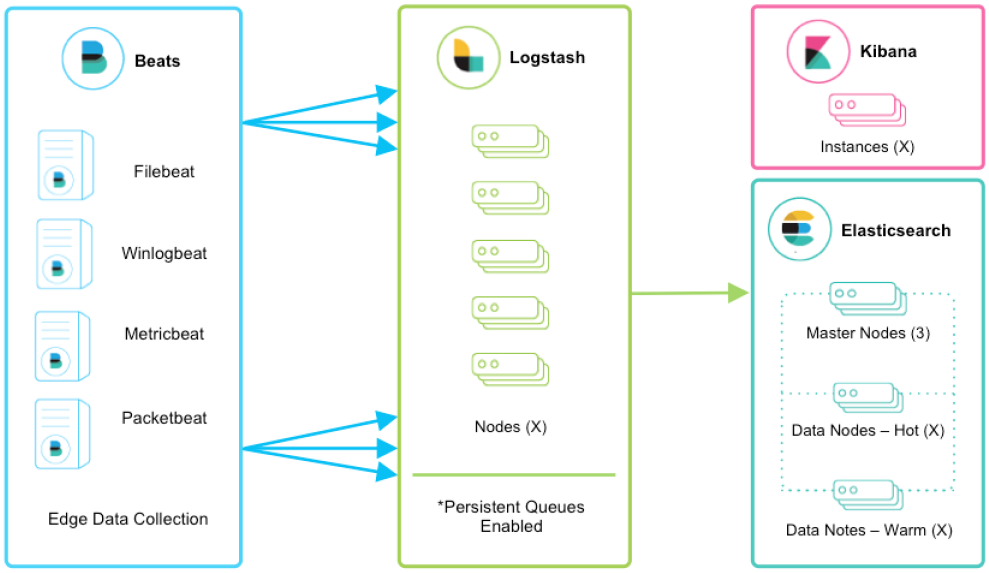

Fig. 4.4. Logstash and beats architecture.

4.1.4.2 Advantages

The advantages of using *Logstash* as the log shipment option are as follows:

- *Linux installation:* It is possible to install Logstash and Filebeat on Linux. Thus, using the same solution for Linux systems reusing the knowledge of the Windows implementation is more straightforward.
- *High-quality documentation:* There is extensive documentation for Logstash and the Beats, available including its past versions.
- *Engaged and active community:* As a consequence of their popularity plus the fact that they are open-source projects, there is an active community that uses and maintains the Logstash and Beats. Furthermore, there are over 200 plugins available, as well as the possibility to create custom plugins.
- *Monitor specific files:* Filebeat provides the possibility to specify which files are to be monitored.
- *S3 integration:* Logstash can send the received data directly to S3.
- *Elasticsearch integration:* Logstash or the Beats can send data to Elasticsearch.
- *Windows Event Logs Beat:* Winlogbeat allows to send the Windows Event Logs from the endpoints to Logstash.
- *Log filtering and aggregation capabilities:* Logstash offers the possibility of enriching and filtering data the received before shipping it to its destination, facilitating the subsequent log analysis process.

4.1.4.3 Disadvantages

The disadvantages of using *Logstash* for shipping logs from Windows 10 devices are as follows:

- *client-server architecture:* The proposed architecture for Logstash is to have the different Beats installed on the endpoint devices and, Logstash set-up as a server that forwards the data. Therefore, scalability and load balancing should be considered during the implementation and maintenance of the solution.

4.1.4.4 Summary

The summary of the advantages and disadvantages of *Logstash* is displayed in the following table:

Logstash Analysis	
Advantages	Linux installation
	High-quality documentation
	Engaged and active community
	Monitor specific files
	S3 integration
	Elasticsearch integration
	Windows Event Logs Beat
	Log filtering and aggregation capabilities
Disadvantages	client-server architecture

Table 4.4: Logstash analysis

4.1.5. NxLog

NxLog is a log collection solution that offers broad compatibility with Operating Systems, such as Windows, Mac OS or FreeBSD. Furthermore, it has different data gathering and enrichment features for each OS type [73].

Its architecture is based on agents installed on endpoint devices, such as employee laptops, which send the data to a collector. Afterwards, this data is redirected to the desired location [74].

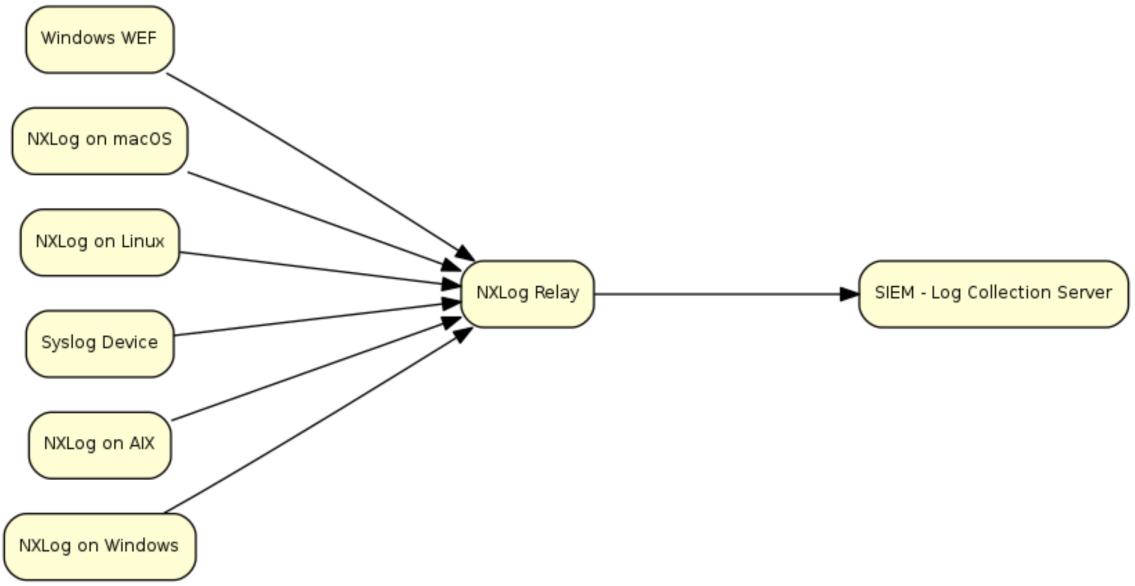

Fig. 4.5. Nxlog architecture.

4.1.5.1 Advantages

The advantages of using *Nxlog* as a log shipment option are as follows:

- *Broad OS compatibility:* NxLog is compatible with Windows, Mac OS, IBM AIX, Linux, Oracle Solaris, FreeDBS and OpenBSD. Therefore, Nxlog is a very flexible solution that can be installed throughout a company's infrastructure, reducing compatibility issue.

- *Several types of accepted logs:* The logs that Nxlog can handle range from DHCP, DNS, File integrity to event logs, among others.

- *Community version available:* Along with the full and paid version of Nxlog, there is a community version that available free of charge facilitating its application for use cases with a reduced number of devices to or requirements [75].

4.1.5.2 Disadvantages

The disadvantages of using *Nxlog* for shipping logs from Windows 10 devices are the following:

- *S3 plugin for Windows not supported:* Even though the official documentation states that the Amazon S3 plugin is available, one of its python dependencies is not supported for Windows. Therefore, the S3 plugin cannot be used at Windows [76] [77].

- *client-server architecture:* Due to Nxlog's architecture, an agent is installed at the endpoints that send the data to a forwarding system, which then ships it to the final destination. Therefore, load balancing and fault tolerance should be considered when installing this device, which increases the complexity and maintenance requirements of the solution.

4.1.5.3 Summary

The summary of the advantages and disadvantages of *Nxlog* is displayed in the following table:

Nxlog Analysis	
Advantages	Broad OS compatibility
	Several types of accepted logs
	Community version available
Disadvantages	S3 plugin for Windows not supported
	client-server architecture

Table 4.5: Nxlog analysis

4.1.6. Windows Event Forwarder

Windows Event Forwarder, or WEF, is the native Windows solution for shipping logs from endpoint devices such as Windows servers and laptops. It is based entirely on native components integrated into Windows 10 Operating Systems [78].

The Windows Event Forwarder architecture follows the client and server approach, with the endpoint devices sending logs to a Windows Event Collector server [79].

It is organized into log subscriptions, where based on event identifiers the endpoints differentiate which event logs should be forwarded to which subscription.

Furthermore, Windows Event Forwarder supports mutual authentication and encryption using Kerberos.

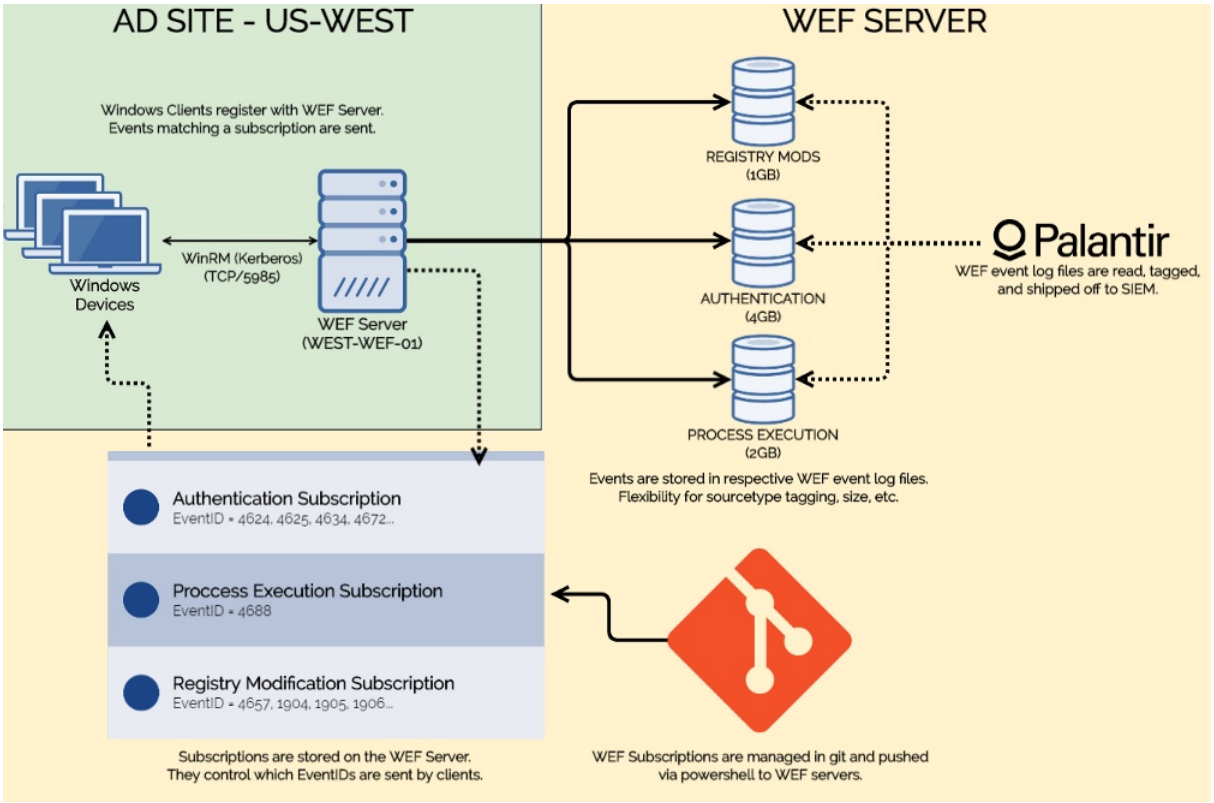

Fig. 4.6. Windows Event Forwarder architecture.

4.1.6.1 Advantages

The advantages of using *Windows Event Forwarder* as the log shipment option are the following:

- *Windows native solution:* Issues related to compatibility between the solution and the devices would be avoided since it is natively supported by Windows.
- *High-quality documentation:* The documentation available on the official Windows website is extensive with topics on its itemization, use for intrusion detection, configuration,an creating custom logs.

4.1.6.2 Disadvantages

The disadvantages of using *Windows Event Forwarder* for shipping logs from Windows 10 devices are as follows:

- *S3 destination not supported:* S3 is not supported as the final destination of the solution.
- *Elasticsearch destination not supported:* Windows Event Forwarded does not support Elasticsearch as the final destination of the solution.
- *There is no OS compatibility apart from Windows OS:* Apart from Windows Operating Systems, no other Operating System will be able to implement this solution.
- *client-server architecture:* As with analogous architecture solutions, the need to manage the systems that receive the data from the endpoints it results in increased complexity.

- *Steep learning curve.* There is no out of the box implementation provided.

4.1.6.3 Summary

The summary of the advantages and disadvantages of *Windows Event Forwarder* is displayed in the following table:

Windows Event Forwarder Analysis	
Advantages	Windows native solution
	High-quality documentation
Disadvantages	S3 destination for Windows not supported
	Elasticsearch destination for Windows not supported
	There is no OS compatibility
	client-server architecture
	Step learning curve

Table 4.6: Windows Event Forwarder analysis

4.2. Summary of the analysis

Summary of the Analysis		
Product	Advantages	Disadvantages
Fluentd	Linux installation	Ruby is necessary
	Serverless architecture	Impossibility to monitor evtx files
	High-quality documentation	
	Engaged and active community	
	Monitor specific files	
	S3 integration	
	Elasticsearch integration	
	Windows Event Logs Plugin	
	Log filtering capabilities	
Fluent bit	Linux and container installation	S3 plugin for Windows not fully supported
	Serverless architecture	Less mature project
	High-quality documentation	Impossibility to monitor evtx files
	Monitor specific files	
	Elasticsearch integration	
	Windows Event Logs Plugin	
	Improved performance compared to Fluentd	
	Log filtering capabilities	

Summary of the Analysis		
Product	Advantages	Disadvantages
Amazon Kinesis Agent for Microsoft Windows	Serverless architecture High-quality documentation Monitor specific files S3 integration Elasticsearch integration Native support for AWS Windows Event Logs native No heavyweight dependencies	Limited Linux OS compatibility Absence of log filtering capabilities
Logstash	Linux installation High-quality documentation Engaged and active community Monitor specific files S3 integration Elasticsearch integration Windows Event Logs Beat Log filtering and aggregation capabilities	Client and server architecture
Nxlog	Broad OS compatibility Several types of accepted logs Community version available	S3 plugin for Windows not supported client-server architecture
Windows Event Forwarder	Windows native solution High-quality documentation	S3 destination for Windows not supported Elasticsearch destination for Windows not supported There is no OS compatibility client-server architecture Step learning curve

Table 4.7: Summary of the Analysis

4.3. Final Choice

The two final solutions to decide which to implement were *Logstash* and *Kinesis Windows Agent*. The chosen service was Kinesis Windows Agent.

Both provide lightweight agents for sending logs from endpoints and an "out of the box" implementation without a steep learning curve for the technology to be used. Additionally, these two solutions have special

features for shipping Windows Event Logs, with Logstash having Winlogbeat and Kinesis having its own Windows Agent.

The main differentiating characteristic of Logstash is its ability to aggregate and modify data incoming before shipping it to S3. This capability could facilitate the further analysis of the logs.

Kinesis Windows Agent does not have this capability, which reduces the possible use cases for it. However, the decisive advantage of Kinesis Windows Agent is the absence of the need for a managed server to forward the logs to the final destination.

Kinesis Windows Agent uses a Kinesis Firehose as a server system which is an AWS managed service, see 3.1.8. Consequently, the fault tolerance, authentication, encryption and load balancing are managed by AWS. Therefore, the implementation of the solution, as well as its required maintenance, is reduced.

In addition, Kinesis Windows Agent integrates with the ideal use case for this analysis, which is the storage of the endpoint's Windows logs in S3. Kinesis Windows Agent sends the data to Kinesis Firehose, which natively integrates with S3 to automatically send the logs to it.

In conclusion, even though Logstash provides higher log enrichment capabilities, the advantage of Kinesis Windows Agent is to avoid having to manage a server device. Therefore, the solution of choice for implementing logs delivery from Windows endpoint devices to S3 is Kinesis Windows Agent.

5. PROJECT ARCHITECTURE

This chapter describes the architecture of Amazon Kinesis Agent for Microsoft Windows implementation.

The chapter is structured to describe the different processes of the infrastructure individually, thus facilitating its understanding of the whole infrastructure.

5.1. Log shipment

The main functionality of Amazon Kinesis Agent is shipping logs from endpoint devices to Cloudwatch and Kinesis Firehose. These logs are then forwarded from Kinesis Firehose to S3, where they are stored.

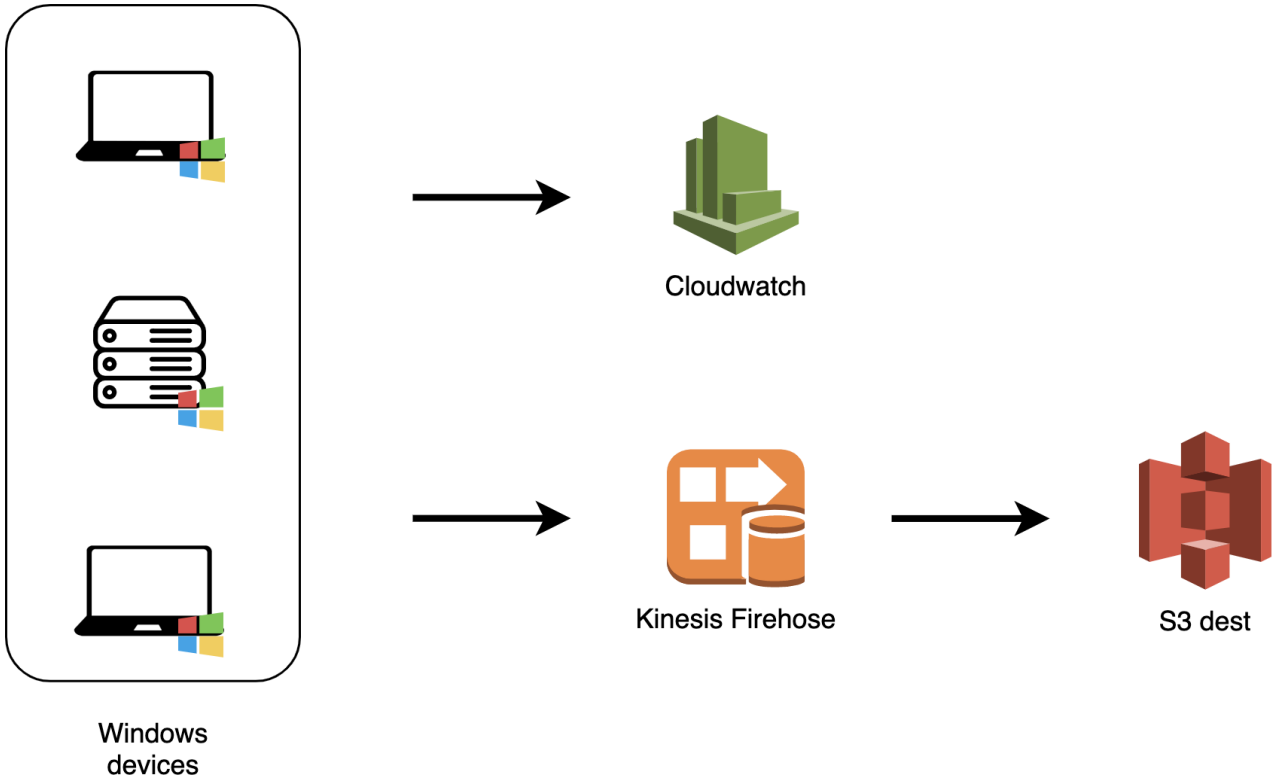

Fig. 5.1. Log shipment architecture.

5.2. Remote management of the devices

NinjaRMM is the tool used for remote management and monitoring of the endpoint devices, see 3.6. Therefore, it could be used to run the installation scripts or rotate the AWS keys used by the devices.

These functionalities are achieved by remote execution of Powershell scripts through NinjaRMM; see 3.5 for Powershell and 6.3.3 for a description of the implemented scripts.

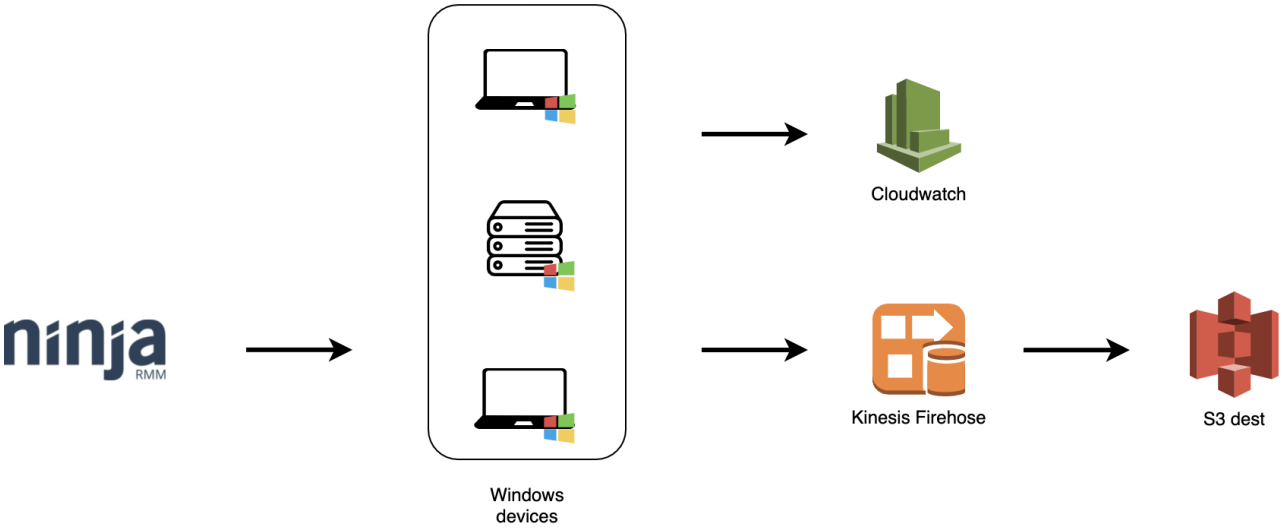

Fig. 5.2. Remote management of endpoint devices.

5.3. Update of AWS infrastructure

The entire AWS infrastructure is managed through terraform files. These terraform files are deployed thorugh a Gitlab's pipeline, see 3.9 for terraform and 3.4 Gitlab CI/CD pipeline.

Therefore, once that commit is pushed to the repository storing the terraform infrastructure code, the Gitlab pipeline updates the AWS resources with the new changes.

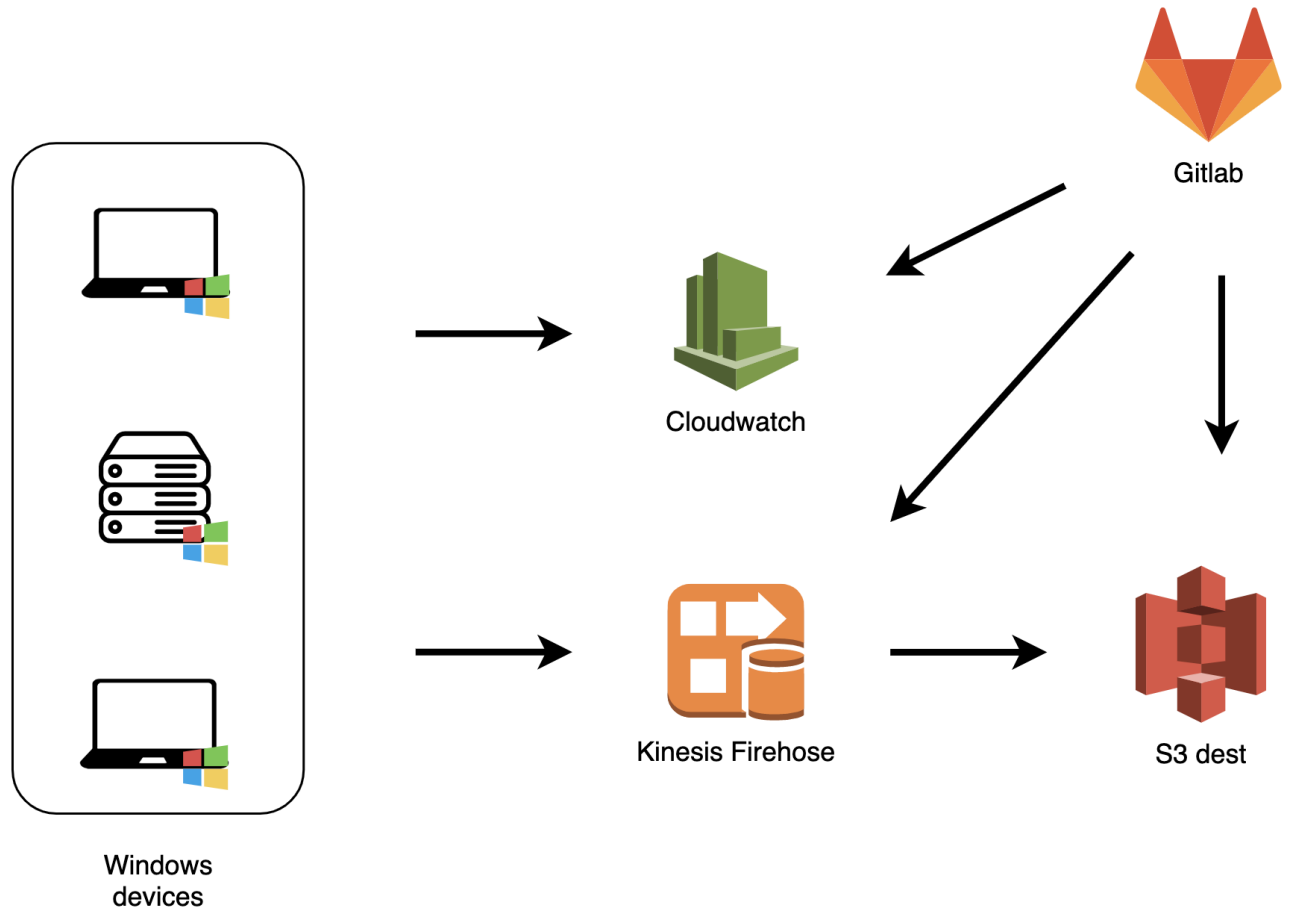

Fig. 5.3. Update of AWS infrastructure.

5.4. Update of Amazon Kinesis Agent configuration.

Similar to updating AWS resources through the Gitlab pipeline, the S3 that stores the configuration files for the Kinesis Windows Agent is automatically updated with the the new version of the configuration files.

This process is achieved through the Gitlab pipeline that checks for any changes in the configuration files code. If there is, it sends the new version stored in the Gitlab repository to the S3 bucket.

Furthermore, the Kinesis Windows Agent is configured to check for new versions of the configuration files located in the S3 bucket every day. Thus, if there is a new version for the configuration files the Kinesis Windows Agent downloads them and re-launch the agent with the new configuration.

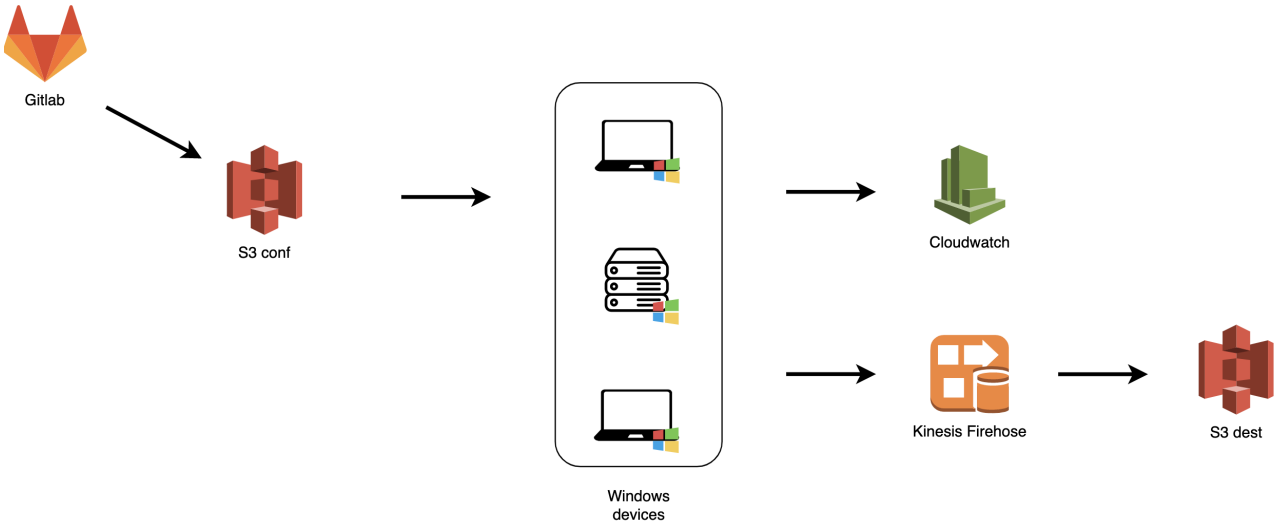

Fig. 5.4. Update of Amazon Kinesis Agent configuration.

5.5. Whole architecture

Finally, the following diagram shows the different elements of the complete infrastructure of the Kinesis Windows Agent implementation as a solution for shipping Windows logs.

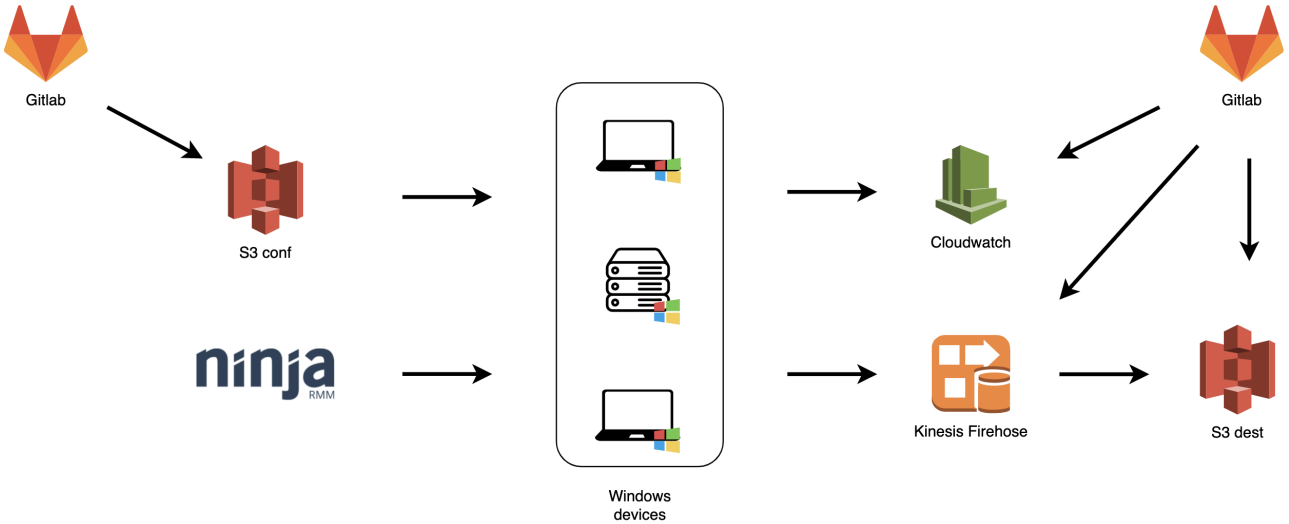

Fig. 5.5. Whole architecture for Kinesis Windows Agent.

6. PROJECT IMPLEMENTATION

This chapter explains the implementation of the Kinesis Windows Agent following the architecture described in the previous chapter.

6.1. Project Organization

The following is a brief overview of the organization of the project structure.

6.1.1. Project Structure

The following diagram of the root folders, which will be explained later, gives an overview of the project structure.

```
windows-logs-kinesis
├── gitlab
├── setup
├── terraform
└── gitlab-ci.yml
```

The following diagram shows in detail the entire structure of the project.

```
windows-logs-kinesis
├── gitlab
│   └── build_image
│       ├── scripts
│       │   └── assume_role.sh
│       ├── config
│       └── Dockerfile
├── setup
│   ├── config
│   │   ├── agent-package-version-p.json
│   │   ├── agent-package-version-s.json
│   │   ├── appsettings-p.json
│   │   ├── appsettings-s.json
│   │   └── AWSKinesisTap.exe.config
│   └── scripts
│       ├── changeAwsKeys.ps1
│       ├── changeEnv.ps1
│       ├── checkKinesis.ps1
│       ├── installKinesis.ps1
│       ├── restartKinesis.ps1
│       ├── uninstallKinesis.ps1
│       ├── updateAppsettings.ps1
│       └── updateConfig.ps1
└── terraform
    └── environments
        └── backend
            ├── <env>-p.tfvars
            └── <env>-s.tfvars
```

```
        │   │   │   │   └── <env>-p.tfvars
        │   │   │   └── kinesis
        │   │   │       ├── <env>-p.tfvars
        │   │   │       ├── <env>-s.tfvars
        │   │   │       ├── <env>-s.tfvars
        │   │   │       └── <env>-p.tfvars
        │   ├── kinesis
        │   │   └── kinesis.tf
        │   └── tf_backend
        │       ├── backend.tf
        │       └── terraform.tfstate
        └── gitlab-ci.yml
```

6.2. GitLab

This section describes the different files belonging to the *GitLab* folder and the *GitLab CI/CD pipeline*.

6.2.1. Project Structure

The following diagram represents the different files regarding the *GitLab* folder and the *pipeline*.

```
windows-logs-kinesis
├── gitlab
│   └── build_image
│       ├── scripts
│       │   └── assume_role.sh
│       ├── config
│       └── Dockerfile
└── gitlab-ci.yml
```

6.2.2. gitlab

6.2.2.1 build_image

Folder containing all files needed to create the Docker image and the script that assumes the AWS role to deploy the pipeline.

scripts/assume_role.sh is a script that assumes the AWS role to create and destroy resources through terraform in the pipeline.

Note that the AWS credentials need to be defined as secrets in the GitLab configuration for the script to access them. Additionally, the branch has to be set as protected to access the secrets.

6.2.2.2 config

File containing the *different profiles* to be used in the pipeline by terraform for the creation of AWS resources.

This file is copied to the .aws folder of the Docker container and is accessed when deploying AWS resources.

6.2.2.3 Dockerfile

Used to create the docker container in which terraform will be run. It uses the image *"hashicorp/terraform:0.14.4"*.

Furthermore, it installs *awscli*, copies the *config* file to the *.aws* folder and copies and runs *scripts/assume_role.sh*.

6.2.3. GitLab Pipeline

The file containing all the information regarding the GitLab pipeline is *.gitlab-ci.yml*. The pipeline consists of *four different stages* that are described below.

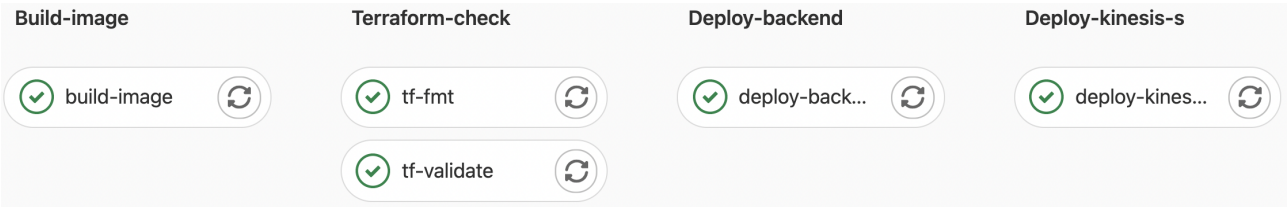

Fig. 6.1. GitLab pipeline overview (for staging).

6.2.3.1 build-image

This stage is the first stage of the pipeline that builds the docker image.

The conditions set to run this stage of the pipeline is a merge request for the production branch.

6.2.3.2 terraform-check

This stage is executed, like the *build-image*, only when there is a merge request to the production branch.

It consists of two different jobs: *tf-fmt*, which checks the correct format of the terraform files, and *tf-validate*, which validates the terraform files.

6.2.3.3 deploy-backend

Creates via terraform the *backend* to be used for the rest of the AWS resources. Note that, unlike the state of terraform files such as kinesis.tf, the state of the backend needs to be uploaded to the repository.

The file terraform/environments/backend/.tfvars is passed as an argument to both commands to perform the planning and application operations through terraforming.

Additionally, the *deploy-backend* stage is executed if there is a commit to master or production which modifies the file *.gitlab-ci.yml* or any of the files from the folder *terraform/tf_backend/*.

6.2.3.4 deploy-kinesis-s/p

These two stages are similar. However, *deploy-kinesis-s* is triggered on the master branch for staging and *deploy-kinesis-p* on the production branch and it creates all the resources belonging to the staging/production environment for the kinesis infrastructure from the *terraform/kinesis/kinesis.tf* file.

Moreover, to perform the *plan* and *apply* commands, the file *terraform/environments/kinesis/<env>.tfvars* is passed as an argument to be used by terraform.

This stage is executed if there is a commit to master that changes the *.gitlab-ci.yml* file or any file in the *terraform/kinesis/* folder or *setup/config/* folder with this name's format **-s.** or **-p.**, for staging and production, respectively.

6.3. Setup

This section explains the different files belonging to the setup folder.

6.3.1. Project Structure

The following diagram represents the different files regarding the *setup* folder.

```
windows-logs-kinesis
└── setup
    ├── config
    │   ├── agent-package-version-p.json
    │   ├── agent-package-version-s.json
    │   ├── appsettings-p.json
    │   ├── appsettings-s.json
    │   └── AWSKinesisTap.exe.config
    └── scripts
        ├── changeAwsKeys.ps1
        ├── changeEnv.ps1
        ├── checkKinesis.ps1
        ├── installKinesis.ps1
        ├── restartKinesis.ps1
        ├── uninstallKinesis.ps1
        ├── updateAppsettings.ps1
        └── updateConfig.ps1
```

6.3.2. config

This folder stores the different configuration files to be uploaded to an S3 bucket, *not publicly available*.

These configuration files are downloaded by the Windows device. The AWS keys held by the end point device have specific permissions to access this bucket.

Furthermore, the agent is set to automatically check daily if the configuration files in the S3 bucket have changed. If so, the agent will automatically download the new version of them and re-launch the agent.

On the other hand, it is also possible to force an update of the configuration files by running *updateAppsettings.ps1*, see 6.3.3.7 and *updateConfig.ps1*, see 6.3.3.8.

6.3.2.1 agent-package-version-p/s.json

The configuration file used for the auto-updating of the Kinesis Windows Agent in the production or staging environment, depending on the file suffix.

6.3.2.2 appsettings-p/s.json

Configuration file defining the *sinks*, *pipes* and *sources* for the Kinesis Windows Agent.

Note that the *sources* are the types of Windows Event logs the agent gathers, *sinks* are the destinations for these logs, such as Cloudwatch of Kinesis Firehose, and *pipes* are the relations between sources and sinks.

6.3.2.3 AWSKinesisTap.exe.config

It contains the default *AWSKinesisTap.exe.config* file with the only addition of the path to the AWS credentials and the .NET version.

Note that this configuration file is shared by both, production and staging environments, as it does not hold contain information about the AWS services with which the agent interacts.

6.3.3. scripts

6.3.3.1 installKinesis.ps1

This is the main script for the Kinesis Windows Agent setup process. It installs the Kinesis Windows agent, the setup of the AWS credentials and the download of the configuration files for the agent.

It can receive two or three *input parameters*: the first two parameters are mandatory and are the AWS access keys. The third parameter is optional and determines the environment in which the Kinesis Windows agent will run.

The environment is defined by a variable that can take the value "p" or "s" for the production and staging environments, respectively. The default value of the environment variable is "s".

The different methods belonging to *installKinesis.ps1* are the following:

- *stopService:* This method stops the Kinesis agent process and service, ensuring that Kinesis will not be running when it should be stopped, such as when we desire to perform a restart or uninstall the agent.

  It receives *one parameter* which is the name of the process to stop. The parameter is hardcoded to *AWSKinesisTap* in the call from *main* and *cleanup*.

- *cleanup:* It removes the *.aws* folder and the *Kinesis Windows Agent* files, which might belong to previous installations. Therefore, when installing the Kinesis Windows Agent it can be ensured that there is no previous installation on the device.

  Note that the *paths* are *hardcoded* as variables within the cleanup method as an array. Moreover, the variables *Env:Programfiles* and */Amazon/AWSKinesisTap* are used to create the files paths.

- *downloadFiles* The purpose of this method is to download the Kinesis installation script provided by Amazon to *C:*. The reason for choosing this path is because in that the scripts can be run as *System* from Ninjarmm.

  Similarly, the *.aws* folder created specially for the Kinesis Windows Agent is located at *C:*. Thus, since it is not located at the default AWS credentials path, it should not produce any conflicts with other AWS credentials used by the user and located in the user's folder.

  Once the installation script is downloaded, it runs it and then the script is removed from the system.

- *setAws:* This method creates the *.aws* folder with the configuration and credential files inside it. It receives two parameters, which are the AWS keys.

As the method is only called once from main, no check is made in *setAws* to verify if both parameters have content. If they did not, an error should have been raised before calling setAws.

The configuration file contains the region, which is hardcoded as a variable within the method. On the other hand, the credentials file is created using the AWS keys passed as parameters.

- *setAwsCli:* This method aims to install the *NuGet* and AWS tools for Powershell if they are not already installed on the end point device.

- *setEnvironment:* This method is called by *setAppSetings* and *setConfig*. It checks in which environment is desired to install Kinesis and sets the necessary variables for it.

  It can receive one *optional parameter* which is the key to the environment to be used. If no parameter is passed, the staging values will be used. However, if the key passed as a parameter is not recognized as one of the valid keys, an error will be displayed and, the script will terminate.

  Finally, the values for each of the environments are defined as a hash map with a key per environment and as a value from a list of all the different values required for the set up.

- *setConfig:* This method downloads the corresponding *AWSKinesisTap.exe.config* file according to the defined environment.

  It receives a single parameter which is the environment to use. Moreover, it calls the *setEnvironment* method to obtain the variables according to the environment.

- *setAppSetings:* It downloads the *appsettings.json* file belonging to the defined environment.

  Like the *setConfig* method, it receives as a parameter the environment and calls the *setEnvironment* method.

6.3.3.2 changeAwsKeys.ps1

The objective of this script is to change the AWS keys on the device to the new ones by passing them as parameters. It uses a subset of the *installKinesis.ps1* methods such as: *stopService*, *cleanup*, *setAws* and *setAwsCli*.

Concerning the method *cleanup*, it only removes the files regarding the AWS credentials and not the files belonging to the Kinesis agent. The rest of the methods are not changed concerning its implementation at *installKinesis.ps1*.

Note that the script must receive the AWS keys as parameters. Failure to pass them as parameters will result in an error.

6.3.3.3 changeEnv.ps1

It changes the environment of the configuration files and AWS keys on the end point device.

The script receives the two AWS keys of the desired environment as parameters, necessary to download the configuration files and its environment variable.

6.3.3.4 checkKinesis.ps1

This script checks if the Kinesis Windows Agent is installed and running on the device, printing the result on the screen.

It can be used from NinjaRMM as a scheduled script to detect if any device is unexpectedly not running the Kinesis Windows Agent.

6.3.3.5 restartKinesis.ps1

The objective of this script is to stop and re-launch the Kinesis Windows Agent from the device.

6.3.3.6 uninstallKinesis.ps1

It removes all files belonging to Kinesis as well as the AWS credentials. This script does not receive any parameter and contains the following methods from *installKinesis.ps1*: *uninstall*, *stopService* and *cleanup*.

6.3.3.7 updateAppsettings.ps1

The purpose of this script is to update the configuration file *appsettings.json* belonging to the Kinesis Windows Agent.

It receives one parameter, which is the environment to be used. The subset of methods used from *installKinesis.ps1* are *setEnvironment*, *setAppSetings* and *stopService*.

6.3.3.8 updateConfig.ps1

Similarly to *updateAppsettings.ps1*, this script updates the *AWSKinesisTap.exe.config* configuration file.

It receives the environment variable as a parameter. The methods used in this script belonging to *installKinesis.ps1* are*setEnvironment*, *setConfig* and *stopService*.

6.4. Terraform

The different files belonging to the *terraform* folder are be explained.

6.4.1. Project Structure

In this section, the following diagram represents the different files regarding the *terraform* folder.

```
windows-logs-kinesis
└── terraform
    ├── environments
    │   ├── backend
    │   │   ├── audit-p.tfvars
    │   │   ├── audit-s.tfvars
    │   │   └── pg-p.tfvars
    │   └── kinesis
    │       ├── audit-p.tfvars
    │       ├── audit-s.tfvars
    │       ├── pg-p-s.tfvars
    │       └── pg-p-p.tfvars
    ├── kinesis
    │   └── kinesis.tf
    └── tf_backend
```

```
├── backend.tf
└── terraform.tfstate
```

6.4.2. environments

This folder contains the variable files to be used for the different accounts and the different environments.

6.4.2.1 backend/<profile>-<env>.tfvars

These files contain the *profile* to be used in the backend for the different environments.

6.4.2.2 kinesis/<profile>-<env>.tfvars

They contain the *profile* to be used at the terraform files and the *suffix* to append to the created AWS resources, depending on the chosen environment.

6.4.3. kinesis

6.4.3.1 kinesis.tf

The file containing all the terraform code regarding the AWS resources for the Kinesis infrastructure.

6.4.4. tf_backend

6.4.4.1 backend.tf

Terraform file with the code regarding the backend to be used by the kinesis infrastructure.

6.4.4.2 terraform.tfstate

The state file belonging to the backend needs to be uploaded to GitLab to run the pipeline correctly.

7. FUTURE DEVELOPMENTS

This chapter will discuss possible future developments of the project. The aim is to use centralized logs to detect malicious activities on the endpoint devices, thus reducing the scope of a security compromise.

The objective of this project was to implement a log shipment solution by centralizing the logs at S3. Consequently, it represents the starting point to the creation of security alarms based on endpoint activity.

The future developments presented below are organized and explained in the ideal chronological order of implementation. Thus, the first step is to integrate a SIEM solution into the infrastructure, then forward the SE logs to the new SIEM, and finally set up security alarms on the received logs.

7.1. SIEM analysis and implementation

The first future development would be to perform an analysis of the different SIEM products on the market. Once the analysis is complete, the implementation of the chosen solution would be accomplished.

The objective of this improvement is to increase the observation capacity of the entire infrastructure by centralizing all the logs in a single place. It should be noted that not only the logs of the endpoint devices will be stored in the SIEM, but also those of hte entire infrastructure.

The SIEM options that would be chosen for the analysis are Splunk, ELK stack provided by Elastic.io with its security package, and Sumo Logic.

7.2. Lambda for forwarding the logs to the SIEM

Once the SIEM is implemented, the logs of the endpoint devices have to be forwarded to the SIEM. Therefore, an integration between the endpoint log shipment project and the SIEM implementation will be achieved.

One approach to accomplish this objective is to create an AWS Lambda function that sends log files to the SIEM when stored in S3.

7.3. Setting alarms at the SIEM depending on the logs

The final future development of the endpoint log shipment project is the creation of security alarms in the logs files received. These alarms will be set up in the SIEM, thus allowing the correlation of endpoint data and logs from other parts of the infrastructure.

Thank to the creation of security alarms, not only will the observability of the infrastructure increase, but also the ability to respond to and detect incidents.

7.4. Future developments architecture

The following diagram represents the proposed architecture for future developments.

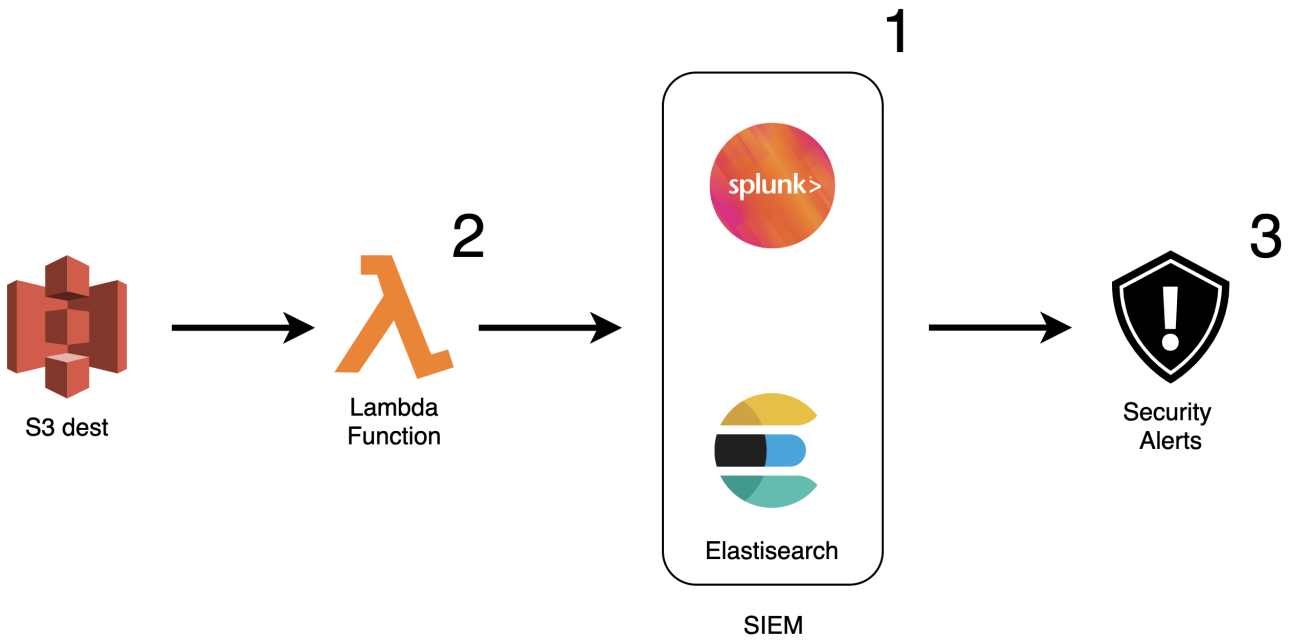

Fig. 7.1. Architecture of the future developments.

As can be observed, its integration with the endpoint log shipment project is seamless. The logs stored in S3 from the endpoint devices are sent through the Lambda function to the SIEM solution, which could be Splunk or Elasticsearch, for example.

Once the logs are stored in the SIEM, they can be processed to detect malicious behaviour and generate security alerts.

8. CONCLUSIONS

In this section, the conclusions obtained in the development of this Master Thesis will be presented. Furthermore, the fulfilment of the initial objectives will be described.

8.1. Fulfilment of the initial objectives

In the first chapter, the initial objectives for the development of this Bachelor Thesis were described. In this section the fulfilment of these objectives will be determined.

- *Analysis of the different log management solutions for Windows 10 devices:* An analysis of the most widespread log management solutions has been performed, considering the use case of an employee's Windows 10 laptop to ideally send logs to S3 for storage.

  The products assessed are Fluentd, Fluent bit, Amazon Kinesis Agent for Microsoft Windows, Logstash, NxLog and Windows Event Forwarder. The result of the analysis is described in 8.1.

- *Election of the solution to be implemented:* Once the analysis of the different solutions has been accomplished, the solution chosen for the implementation is Amazon Kinesis Agent for Microsoft Windows.

- *Proof of Concept implementation of the chosen solution:* A PoC was accomplished to have a working version of the project. The PoC was used to prove its usability and integration with the enterprise's infrastructure.

- *Solution's infrastructure management automation:* The cloud infrastructure for the solution's implementation is managed through terraform files updates in a Gitlab pipeline to facilitate the management of the solution.

- *Remote management capabilities integration:* Amazon Kinesis Agent for Microsoft Windows has been integrated with Ninjarmm, enabling remote installation and maintenance of the solution on employee laptops.

- *Implementation of the solution on employee laptops:* Amazon Kinesis Agent for Microsoft Windows was successfully implemented on all Windows 10 employee devices from the company.

8.2. Conclusion of the analysis

After the different solutions were analysed, Kinesis Windows Agent was the chosen option for implementing the endpoint log shipment solution.

Kinesis Windows Agent provides a serverless architecture, where the agent sends logs from the endpoints to Kinesis Firehose. Moreover, Kinesis Windows Agent does not require any heavy-weight dependencies and its configuration is straightforward.

Because Kinesis Firehose is an AWS managed service, there is no need to handle the scaling or fault

tolerance issues common in a client-server architecture. Furthermore, Kinesis Firehose integrates seamlessly with S3 with native data forwarding to it.

The following table with all the results is provided to summarize the analysis of the different solutions.

Summary of the Analysis		
Product	Advantages	Disadvantages
Fluentd	Linux installation Serverless architecture High-quality documentation Engaged and active community Monitor specific files S3 integration Elasticsearch integration Windows Event Logs Plugin Log filtering capabilities	Ruby is necessary Impossibility to monitor evtx files
Fluent bit	Linux and container installation Serverless architecture High-quality documentation Monitor specific files Elasticsearch integration Windows Event Logs Plugin Improved performance compared to Fluentd Log filtering capabilities	S3 plugin for Windows not fully supported Less mature project Impossibility to monitor evtx files
Amazon Kinesis Agent for Microsoft Windows	Serverless architecture High-quality documentation Monitor specific files S3 integration Elasticsearch integration Native support for AWS Windows Event Logs native No heavyweight dependencies	Limited Linux OS compatibility Absence of log filtering capabilities

Summary of the Analysis		
Product	Advantages	Disadvantages
Logstash	Linux installation High-quality documentation Engaged and active community Monitor specific files S3 integration Elasticsearch integration Windows Event Logs Beat Log filtering and aggregation capabilities	Client and server architecture
Nxlog	Broad OS compatibility Several types of accepted logs Community version available	S3 plugin for Windows not supported client-server architecture
Windows Event Forwarder	Windows native solution High-quality documentation	S3 destination for Windows not supported Elasticsearch destination for Windows not supported There is no OS compatibility client-server architecture Step learning curve

Table 8.1: Summary of the Analysis

8.3. Conclusion of the implementation

The implementation of the Kinesis Windows Agent on all employee's Windows devices was a success.

The code for the installation and maintenance of the Kinesis Windows Agent was primarily developed in Powershell scripts remotely activated using Ninjarmm, a remote management tool. The code was stored in a Gitlab repository with a CI/CD pipeline.

Additionally, the AWS infrastructure code required for this project was developed using Terraform. Through Gitlab's CI/CD pipeline, the AWS resources are automatically updated if the code is modified.

Kinesis Windows Agent configuration files are stored in a private S3 bucket. As with terraforming files, if configuration files are modified in the Gitlab repository, they are updated in the S3 bucket through the Gitlab pipeline.

Furthermore, Kinesis Windows Agent automatically checks every day if there is a new version of the configuration files at the S3 bucket. If there is, it downloads them and relaunches the agent.

Equivalently, the agent verifies every 24 hours if there is a new available version of Kinesis Windows Agent. Therefore, if there is a new released, the agent will automatically download and install it on the device.

BIBLIOGRAPHY

[1] *About us | about you*, Available at https://corporate.aboutyou.de/en/about-us.

[2] R. Cordray, *Why log management is absolutely critical for it security*, Available at https://www.itworldcanada.com/blog/why-log-management-is-absolutely-critical-for-it-security/377711, Oct. 2015.

[3] D. Bisson, *What is log management?* Available at https://www.tripwire.com/state-of-security/security-data-protection/security-controls/what-is-log-management/, Nov. 2017.

[4] W. Wei, *Importance of logs and log management for it security*, Available at https://thehackernews.com/2013/10/importance-of-logs-and-log-management.html, Oct. 2013.

[5] E. Barajas, *What is log management?* Available at https://www.tripwire.com/state-of-security/incident-detection/log-management-siem/log-management-why-important/, Feb. 2020.

[6] L. Rosencrance, *What is siem and why is it important?* Available at https://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM, Feb. 2020.

[7] *Definition of siem - it glossary*, Available at https://www.gartner.com/en/information-technology/glossary/security-information-and-event-management-siem.

[8] A. M. Susan Lund and O. Robinson, *The future of work after covid-19*, Available at https://www.mckinsey.com/featured-insights/future-of-work/the-future-of-work-after-covid-19, Feb. 2021.

[9] K. Costello and L. Goasduff, *Gartner says worldwide iaas public cloud services market grew 31.3% in 2018*, Available at https://www.gartner.com/en/newsroom/press-releases/2019-07-29-gartner-says-worldwide-iaas-public-cloud-services-market-grew-31point3-percent-in-2018, Jul. 2019.

[10] *Amazon web services (aws) - cloud computing services*, Available at https://aws.amazon.com.

[11] V. Page, *What is amazon web services and why is it so successful?* Available at https://www.investopedia.com/articles/investing/011316/what-amazon-web-services-and-why-it-so-successful.asp, May 2020.

[12] *Azure vs. aws*, Available at https://azure.microsoft.com/en-us/overview/azure-vs-aws/.

[13] *Aws vs azure-who is the big winner in the cloud war?* Available at https://www.dezyre.com/article/aws-vs-azure-who-is-the-big-winner-in-the-cloud-war/401/, Jul. 2021.

[14] *Amazon machine images (ami) - amazon elastic compute cloud*, Available at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

[15] *Amazon cloudwatch - application and infrastructure monitoring*, Available at https://aws.amazon.com/cloudwatch/.

[16] N. Patil, *How to use aws cloudwatch with on-premise application components?* Available at https://cloudnineapps.com/blogs/cloud-computing/how-to-use-aws-cloudwatch-with-on-premise-application-components/, Jan. 2020.

[17] *Installing the cloudwatch agent on on-premises servers*, Available at https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-premise.html.

[18] *Amazon dynamodb*, Available at https://aws.amazon.com/dynamodb/.

[19] A. DeBrie, *The what, why, and when of single-table design with dynamodb*, Available at https://www.alexdebrie.com/posts/dynamodb-single-table/, Feb. 2020.

[20] *Acid properties of transactions*, Available at https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions, Feb. 2021.

[21] *Amazon ec2*, Available at https://aws.amazon.com/ec2/.

[22] *What is aws ec2?* Available at https://www.sumologic.com/insight/what-is-aws-ec2/, Jun. 2019.

[23] *Amazon elasticsearch service*, https://aws.amazon.com/elasticsearch-service/.

[24] *What is elasticsearch?* https://www.elastic.co/what-is/elasticsearch.

[25] K. D. Carl Meadows Jules Graybill and M. Shah, *Stepping up for a truly open source elasticsearch*, Available at https://aws.amazon.com/blogs/opensource/stepping-up-for-a-truly-open-source-elasticsearch/, Jan. 2021.

[26] *Elastic is more than amazon elasticsearch service*, https://www.elastic.co/aws-elasticsearch-service.

[27] *Aws identity and access management (iam)*, https://aws.amazon.com/iam/.

[28] *What is kibana?* https://aws.amazon.com/elasticsearch-service/the-elk-stack/kibana/.

[29] *Kibana: Explore, visualize, discover data*, https://www.elastic.co/kibana.

[30] D. Berman, *Using the elk stack for siem*, https://logz.io/blog/elk-siem/, Jun. 2018.

[31] *Amazon kinesis data firehose*, https://aws.amazon.com/kinesis/data-firehose.

[32] *Amazon kinesis data firehose*, https://aws.amazon.com/kinesis/data-streams/.

[33] *Amazon s3*, https://aws.amazon.com/s3/.

[34] *Gitlab ci/cd*, https://docs.gitlab.com/ee/ci/.

[35] *Gitlab is the open devops platform*, https://about.gitlab.com/.

[36] *Gitlab - ci/cd*, Available at https://www.tutorialspoint.com/gitlab/gitlab_ci_cd.htm.

[37] *What is powershell?* Available at https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1.

[38] *Ninjarmm*, Available at https://www.ninjarmm.com.

[39] *Ruby*, Available at https://www.ruby-lang.org/en/.

[40] *About ruby*, Available at https://www.ruby-lang.org/en/about/.

[41] *Browse providers | terraform registry*, https://registry.terraform.io/browse/providers.

[42] *Introduction to terraform*, https://www.terraform.io/intro/index.html.

[43] *Terraform - reviews, pros & cons | companies using terraform*, https://stackshare.io/terraform.

[44] *Get started with visual studio code*, Available at https://code.visualstudio.com/learn.

[45] *Github - microsoft/vscode at 1.56.2*, Available at https://github.com/microsoft/vscode.

[46] *Developer survey results 2019 – most popular development environments*, Available at https://insights.stackoverflow.com/survey/2019#development-environments-and-tools.

[47] *Fluentd | open source data collector | unified logging layer*, Available at https://www.fluentd.org/.

[48] *What is fluentd?* Available at https://www.fluentd.org/architecture.

[49] *Testimonials | fluentd*, Available at https://www.fluentd.org/testimonials.

[50] *Installation - fluentd*, Available at https://docs.fluentd.org/installation.

[51] *Introduction - fluentd*, Available at https://docs.fluentd.org.

[52] *Tail - fluentd*, Available at https://docs.fluentd.org/input/tail.

[53] *S3 - fluentd*, Available at https://docs.fluentd.org/output/s3.

[54] *Elasticsearch - fluentd*, Available at https://docs.fluentd.org/output/elasticsearch.

[55] *Windows_eventlog - fluentd*, Available at https://docs.fluentd.org/input/windows_eventlog.

[56] *Install by .msi installer (windows) - fluentd*, Available at https://docs.fluentd.org/installation/install-by-msi.

[57] *Fluent bit*, Available at https://fluentbit.io.

[58] *Windows event log - fluent bit*, Available at https://docs.fluentbit.io/manual/pipeline/inputs/windows-event-log.

[59] *Aws plugins can not connect to aws services using docker build · issue #3154*, Available at https://github.com/fluent/fluent-bit/issues/3154.

[60] *Fluent bit .exe is complaining about missing libcrypto-1_1-x64.dll in 1.7.1 and 1.7.2 installer and zip packages · issue #3186*, Available at https://github.com/fluent/fluent-bit/issues/3186.

[61] *What is amazon kinesis agent for microsoft windows?* Available at https://docs.aws.amazon.com/kinesis-agent-windows/latest/userguide/what-is-kinesis-agent-windows.html.

[62] *Source declarations | amazon kinesis agent for microsoft windows*, Available at https://docs.aws.amazon.com/kinesis-agent-windows/latest/userguide/source-object-declarations.html#directory-source-configuration.

[63] *Sink declarations*, Available at https://docs.aws.amazon.com/kinesis-agent-windows/latest/userguide/sink-object-declarations.html.

[64] *Getting started with amazon kinesis agent for microsoft windows*, Available at https://docs.aws.amazon.com/kinesis-agent-windows/latest/userguide/getting-started.html#getting-started-prerequisites.

[65] *Writing to amazon kinesis data streams using kinesis agent*, Available at https://docs.aws.amazon.com/streams/latest/dev/writing-with-agents.html.

[66] *The elk stack*, Available at https://aws.amazon.com/elasticsearch-service/the-elk-stack/.

[67] *Logstash: Collect, parse, transform logs*, Available at https://www.elastic.co/logstash.

[68] *Logstash introduction*, Available at https://www.elastic.co/guide/en/logstash/current/introduction.html.

[69] *Deploying and scaling logstash*, Available at https://www.elastic.co/guide/en/logstash/current/deploying-and-scaling.html#_beats_and_logstash.

[70] *Beats: Data shippers for elasticsearch*, Available at https://www.elastic.co/beats/.

[71] *Filebeat: Lightweight log analysis*, Available at https://www.elastic.co/beats/filebeat.

[72] *Winlogbeat: Analyze windows event logs*, Available at https://www.elastic.co/beats/winlogbeat.

[73] *Nxlog enterprise edition*, Available at https://nxlog.co/products/nxlog-enterprise-edition.

[74] *28. centralized log collection*, Available at https://nxlog.co/documentation/nxlog-user-guide/centralized.html.

[75] *Nxlog community edition*, Available at https://nxlog.co/products/nxlog-community-edition.

[76] *153. amazon s3*, Available at https://nxlog.co/documentation/nxlog-user-guide/addon-amazon-s3.html.

[77] *Om_python module | log collection solutions*, Available at https://nxlog.co/question/6129/ompython-modulel.

[78] *Windows event forwarding for network defense*, Available at https://blog.palantir.com/windows-event-forwarding-for-network-defense-cb208d5ff86f, Sep. 2017.

[79] *Use windows event forwarding to help with intrusion detection (windows 10)*, Available at https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection, Feb. 2019.