MSc thesis

Computer Science

Improving Cloud Governance by Increasing Observability

Markus Nousiainen

June 1, 2020

Faculty of Science

University of Helsinki

Supervisor(s)

Prof. Keijo Heljanko

Examiner(s)

Prof. Keijo Heljanko and D.Sc. (Tech.) Tewodros Deneke

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Computer Science

Tekijä — Författare — Author

Markus Nousiainen

Työn nimi — Arbetets titel — Title

Improving Cloud Governance by Increasing Observability

Ohjaajat — Handledare — Supervisors

Prof. Keijo Heljanko

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
MSc thesis	June 1, 2020	43 pages, 44 appendice pages

Tiivistelmä — Referat — Abstract

Rise in popularity of Cloud computing has introduced new challenges for IT-governance. The multitude of different services and possible configurations Cloud providers offer can make it hard to get a comprehensive overview of the environment. To successfully govern an organisations Cloud environment it is important to be able to easily make accurate and reliable observations of the environments state, security, and changes to the configurations.

This thesis takes a look into the research literature to find out what kinds of risks have been identified in governing the Cloud environment and ways to mitigate them. One of the latest advancements in improving the Cloud governance is the introduction of automated formal reasoning tools for configuration analysis.

One customer case where multiple vendors are building services on multiple cloud accounts is used as an example. Architecture for application, security, and audit log collection, indexing, and monitoring is described. Special attention is given to the identity and access management requirements. The thesis concludes with the assessment of the selected approach and tools and services used to implement it. Some alternative solutions, possible improvements, and further development to the implementation are considered.

ACM Computing Classification System (CCS)

Applied computing → Enterprise computing → IT governance

Applied computing → Enterprise computing → Enterprise architectures → Enterprise architecture management

Computer systems organization → Architectures → Distributed architectures → Cloud computing

Security and privacy → Security services → Access control

Avainsanat — Nyckelord — Keywords

cloud, governance, observability, logging, monitoring, alerts

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Networking and Services specialisation line

Contents

1 Introduction

Public Cloud platforms have opened a lot of possibilities to develop IT environments, but at the same time brought new challenges for the IT-governance. Deployment and configuration of the resources can be done effortlessly and almost without the need to buy and setup new servers and hardware infrastructure. The number of the services and components used in modern computing environment on the Cloud platform can be huge. Keeping up with all the configurations and possible security risks can become overwhelming without a proper monitoring system.

In modern software projects there are often multiple vendors building applications in distributed manner on Cloud platforms. Applications and services can be distributed not only onto multiple different servers and virtual machines, but also onto multiple accounts and even platforms. This kind of architecture creates great challenges for monitoring and analysing systems operation, state and health. Traditional ways of collecting logs of individual applications and monitoring performance of individual servers is not enough. To get an holistic overview of the system, a centralised solution is required.

Cloud infrastructure and governance specialists in multi-vendor software projects have to take into account varying requirements for observability that different interest groups have. Management, security, operations and development teams all have their individual use cases for the data that can be collected. On the other hand the multi-vendor nature of projects might necessitate more focus on security details of logging and monitoring systems. Some organisations require more fine-grained access control to give just enough visibility to log and monitor data for employees and contractors based on their roles in systems development and maintenance. Compliance with legislation and regulations create their own challenges in designing an observable system, particularly EU's General Data Protection Regulations (GDPR) (European Parliament, 2016) requires extra considerations on what can be logged, who can access logs, and how to securely store the log data.

The purpose of this thesis is to present a way to increase the observability of a Cloud environment and make it easier for the IT-governance to guide the application development and to make educated decisions regarding the environment based on the observations. Chapter 2 takes a look into the research literature and earlier work related to Cloud governance and observability. Section 2.1 introduces some of the characteristics of the Public Cloud platforms. Section 2.2 lists some of the risks that have been identified to be related to Cloud governance. One key area of concern for Cloud governance is the identity and authentication

management, Section 2.3 takes a look on two papers concentrating on the subject. Some of the latest developments in the automation of monitoring the Cloud configurations are introduced in the Section 2.4. Section 2.5 takes a look into the distribution of data security related responsibilities between different interest groups in Cloud environment. In Section 2.6 basic concepts of observability are described. Chapter 3 describes one customer case that requires both improvements in systems observability and compliance to strict security guidelines and government regulations. Solution for aforementioned customers needs is presented in Chapter 4. Chapter 5 is used to evaluate the solution and selected technologies. Possible future improvements are also discussed in Chapter 5

2 Cloud Governance

In the past ten years public Cloud adaptation have grown in fast pace and often private data centres are not even considered when companies are starting to build their it-infrastructure. Figure 2.1 shows the exponential growth in enterprises spending on Cloud services compared to the rather stagnated spending on data centre hardware and software. In 2013 Borgman

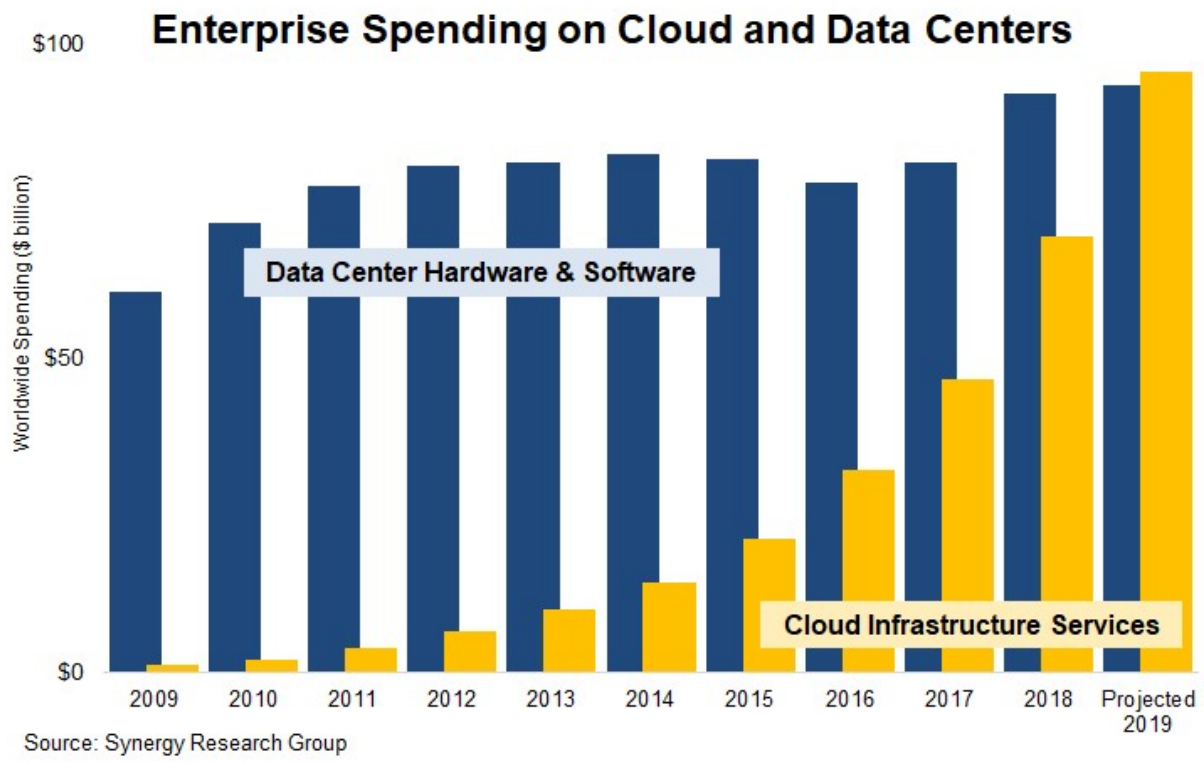

Figure 2.1: Enterprise spending on Cloud and Data Centers in the last decade worldwide. (source: https://www.srgresearch.com/articles/the-decades-megatrends-in-numbers-part-1)

et al. conducted a survey targeting IT-governance professionals and decision makers on the factors influencing the decision to adapt to the Cloud platforms (Borgman et al., 2013). They extended the TOE-framework (Technology-Organisation-Environment) developed by Tornatzky et al. in the nineties, and used it to investigate the adaptation factors. They concluded from the results that the balance of the expected Cloud governance risks and advantages clearly guides the decision making in organisations on when and how to adapt to the Cloud Computing. This thesis tries to identify some of the risks decision makers sees in Cloud and how IT-governance can be improved to mitigate those risks by increasing the observability of the Cloud environment.

The Cloud platform market is dominated by a few large players. Figure 2.2 shows the market shares of the eight largest companies providing Cloud services. Number of the services each one of the providers have on offer keeps constantly growing. For example AWS has more than 200[1] services on offer at the time of writing this thesis (may 2020), where as in January 2019 the number was 183.

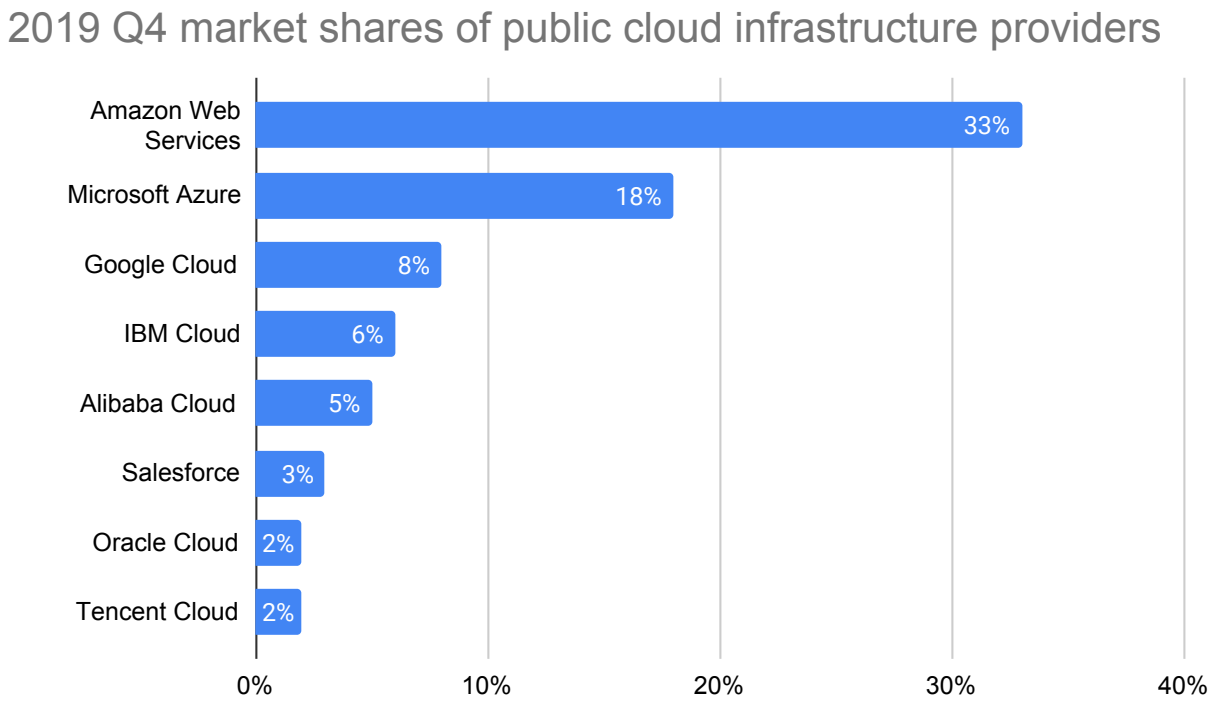

Figure 2.2: Market shares of leading public Cloud providers in fourth quarter of 2019 (source: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/)

2.1 Cloud Computing Characteristics

United States National Institute Of Standards and Technology (NIST) defines the term ”Cloud Computing” based on five essential characteristics, three service models, and four deployment models (Mell, Grance, et al., 2011). Deployment models in NIST’s definitions are Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud. This thesis will concentrate solely on building software systems on the Public Cloud. According to NIST the Public Cloud is public accessible, can be operated by private company, academic, or

[1]https://www.techradar.com/news/awscomplete-list-of-amazon-web-services

government organisation and the infrastructure resides in providers premises. Three service models NIST uses are:

- Infrastructure as a Service (Iaas). Users are responsible for configuring their own virtual infrastructure resources such as deploying compute capacity, patching operating systems and defining networks. Underlying hardware and means to configure infrastructure are provided.
- Platform as a Service (PaaS). Platform to run applications is provided. Users do not need to worry about operating system or underlying resources, but it is their responsibility to build and deploy software on top of the platform.
- Software as a Service (SaaS). Software is run on Cloud platform and provider of the service is responsible for maintenance of the Software. User might have access to some application specific configurations, but does not have access to operating system or infrastructure level configurations.

The palette of services large Public Cloud providers offer comprises of all three service models. As for the five essential characteristics of Cloud Computing NIST lists the following:

- On-demand self-service.
- Broad network access.
- Resource pooling. Customers use Cloud Providers computing storage and network resources from shared pool. Resources are often virtual although sometimes it is possible to reserve actual physical resources too.
- Rapid elasticity.
- Measured service.

”Rapid elasticity” and ”On-demand self-service” are two of the essential characteristics from NIST that make Cloud Computing and especially Public Cloud offerings very compelling for organisations when assessing platform options for the software infrastructure. The easiness of configuring and adding new resources to your environment. That is probably where the greatest risk lies too. When deployment of a new database, adding an object storage, or exposing existing resource to the whole internet requires just a few mouse clicks. It is something that can be easily done by a person who is lacking the relevant knowledge to properly evaluate security and compliance aspects of the IT-infrastructure.

2.2 Cloud Governance Risks

The Open Web Application Security Project (OWASP) listed top ten Cloud security risks in 2010 (Chebrolu et al., 2010) as follows:

- R1: Accountability & Data Risk
- R2: User Identity Federation
- R3: Regulatory Compliance
- R4: Business Continuity & Resiliency
- R5: User Privacy & Secondary Usage of Data
- R6: Service & Data Integration
- R7: Multi-tenancy & Physical Security
- R8: Incidence Analysis & Forensics
- R9: Infrastructure Security
- R10: Non-production Environment Exposure

Most relevant risks from OWASP's list for this thesis are R1, R2, R3, R4, R5 and R8.
In 2013 University of Sheffield researchers conducted a survey (Dutta et al., 2013) targeting IT experts with substantial professional experience with Cloud technologies. They gathered 39 potential Cloud computing risks from literature review and systematic analysis and build Cloud computing risks onthology based on those. They used social media service LinkedIn to find the respondents for the survey. Based on the answers they calculated risk scores for each potential risk with formula:

$$\sum[W * (Probability + Impact + Frequency)]$$

The score for specific risk is calculated by summing individual respondents scores for that risk. W is the coefficient for individual respondents score that gets value 0, if the correspondent does not consider the risk to be Cloud related and value 1 if the respondent considers the risk to actually be Cloud computing related. Probability, Impact, and Frequency are scores that the respondent gave to those dimensions for the risk event. The probability score does not mean mathematical probability, instead it indicates how probable the respondent thinks the risk is. Probability score 2 meaning high probability, 1 medium probability, and 0.5 low

probability. Similarly impact dimension was scored with three values, 2 for high impact, 1 for medium impact and 0.5 for low impact. Five discreet values from 0.25 to 2 were used for scoring frequency dimension. Researchers ordered risk events based on the scores and their list of top ten Cloud computing risks are:

1. Privacy of enterprise or customer data is jeopardised in the Cloud.
2. Inconsistent data protection laws adopted by different countries where Cloud data are generated and stored.
3. Difficult for user companies to change Cloud vendors even in the case of service dissatisfaction (also known as vendor lock-in).
4. User companies lack disaster recovery and contingency plans to deal with unexpected technical issues in Cloud environment.
5. Enterprise data re-migration difficulties at the end of the Cloud contract.
6. Inadequate user training/knowledge on Cloud services and usage.
7. Cloud applications become temporarily unavailable or out-of-service.
8. Increasing hidden costs due to non-transparent operating models in the Cloud.
9. Denial-of-Service (DoS) attacks in the Cloud environment.
10. Unauthorised access to enterprise data/applications in the Cloud.

This thesis concentrates mostly on the mitigation of the risks 1, 4, 6. and 10. In their predictions of cyber security threats for 2020 Trend Micro Research lists configuration mistakes as one of the most prominent sources for risks in Public Cloud environments (*Trend Micro Security Predictions for 2020* 2019). Trend Micro Researchers see the risk in usage of the third party vendors who are not used to the Cloud environments. Improperly configured access management to storage services imposes an obvious attack vector. Rise of the serverless computing and container services enable easy deployment of application code that have access to sensitive data. Every developer can easily configure their own computing environment and deploy their own code without the need to think about server instances, network infrastructure, routing and firewalls. Removed need to maintain server and network infrastructure makes it easy to overlook security concerns (Brandis et al., 2019)

.

2.3 Identity and Access Management

One of the key areas of Cloud security is the Identity and Access Management, IAM for short. Cloud environments provide easy access to the resources and configurations from anyplace with network access. Robust way to control who can access which resources is essential. It is also important to identify actors in the environment to be able to generate the trace of changes in the configurations with the information of who is behind each change. Gonzales et al. proposed a framework for studying and developing solutions for authentication and authorisation in the Cloud in their 2013 paper aptly named "A framework for for authentication and authorisation credentials in Cloud computing" (Gonzalez et al., 2013). Their framework concentrates on different types of credentials and their relation to different kind of deployment models, service types and Cloud entities. They define four key concepts in Cloud computing authentication: Entities, Attributes, Identities and Credentials. **Entities** consists of higher level actors and objects in the Cloud environment, such as Cloud providers, customers, users or services. **Identity** uniquely specifies a single entity. **Attributes** are some information about the entity and attributes are assigned to the entities by grouping them into credentials. **Credentials** are groupings of attributes and can be shared by multiple entities or be assigned to a single entity. For example when some Cloud resource should be made available for multiple users it is possible to either define an credential for each individual user or share the credential between the group of users. Credentials are classified into three categories, metadata, authorisation and obligation. **Authorisation** gives access to some other entity. **Obligation** defines consists of attributes defining a set of rules and policies that the entity have to comply to. **Metadata** can be any information about the entity, not belonging to the other categories. Credential as a set of attributes can belong to multiple categories. One credential can for example be a document that gives access to some resource and contain information about the resource as metadata. Their framework is presented in the form of a chart which is recreated in Figure 2.3

In 2014 Yang et al. presented an architecture for the identity and access management (Yang et al., 2014). Their framework consists of four key components. **Cloud Resource Provider** (CRP) is responsible of providing both computing Cloud services such as computing and network infrastructure, and resources that user have uploaded into the Cloud, such as files, data stored in database and applications. **Identity Management** (IDM), manages both users and their identities. IDM uses multifactor authentication process. **Policy Management** is responsible for user authorisation and management of the attributes of the entities. **Resources Engine and Policy Decision-making** REPD is the component that handles users requests and works as an intermediator between user and other components. REDP consists

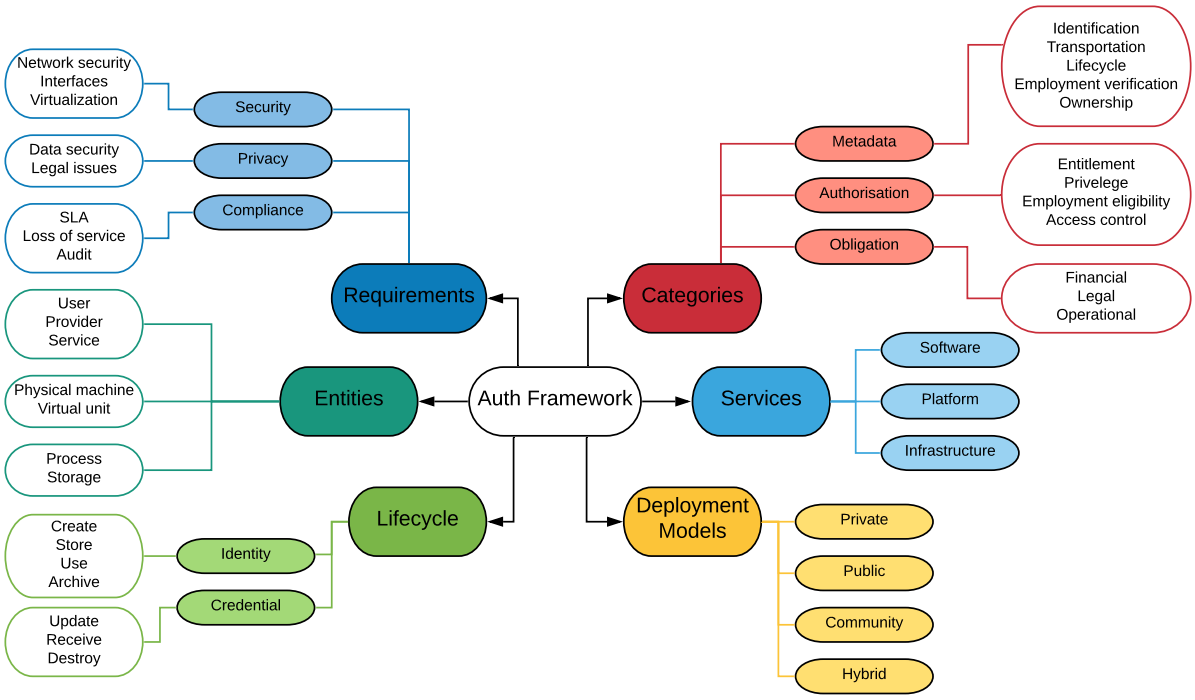

Figure 2.3: Authentication Framework

of three components, Policy Inquiry Service, Resources Engine, and Policy Decision-making. Workflow proposed in their paper goes as follows.

1. User requests access and REPD redirects the request to the IDM.
2. IDM handles the authentication and validates the users credentials against its database.
3. If authentication succeeds, users identity information is returned to REDP.
4. Policy Inquiry Service is used to queried for users priveledges.
5. Policy Decision Making is used to decide the if the user has access to the service based on the identity, users priviledges, and resources access policy.
6. If the access is granted. CRP is used to access the resource.

2.4 Automated Configuration and Security Auditing

Trend Micro Research recently published a white paper on risk assessment and mitigation in Public Cloud infrastructures (Swimmer et al., 2020). One of the key takeaways from the conclusions section of their paper is that people responsible of the security of the Cloud

environment should find ways to analyse the configurations and deployment scripts. When the number of used services and developers building the environment grows it becomes a tedious job to keep up with all the configurations and their effect on the overall security of the system. One security hole in a configuration of one of the services might be enough to compromise the whole system. Of course it is important to try to have several layers of security measures, so that possible breaches can be isolated as much as possible, but sometimes it is hard to see how everything is connected in complicated environments. One promising new development in the area of configuration auditing is the ability to monitor the configurations programmatically.

In recent years AWS as well as other Public Cloud providers have developed automatic mechanisms to detect security risks in configurations clients have made (Cook, 2018). Previously the use of automated formal methods for proofing the correctness of automated data processing systems, have been restricted to limited, well defined problem spaces like microprocessor design or safety systems in aerospace industry. It has not been feasible to use these kinds of methods to find security vulnerabilities or other problems in complex systems such as web services, industry automation, or support services for enterprises. Building a custom model to base formal reasoning on in complex systems have been too large of an investment. The rise of Cloud services with well defined API's have enabled these methods to be used in a more general manner to benefit diverse customer base. AWS security team has been using formal reasoning while developing their own services and now they also offer several services for customers that include automated formal reasoning techniques to guide decision making and to help detect possible errors in configurations (Backes et al., 2019).

2.5 Accountability

With the rise of Cloud computing the question of the data security related accountability have become more prevalent. Cattedu et al. from A4Cloud project proposed a model for Cloud computing accountability in their 2013 paper (Catteddu et al., 2013). A4Cloud Definition of Accountability: *"Accountability for an organisation consists of accepting responsibility for the stewardship of personal and/or confidential data with which it is entrusted in a Cloud environment, for processing, storing, sharing, deleting and otherwise using the data according to contractual and legal requirements from the time it is collected until when the data are destroyed (including onward transfer to and from third parties). It involves committing to legal and ethical obligations, policies, procedures and mechanisms, explaining and demonstrating ethical implementation to internal and external stakeholders and remedying any failure to act*

properly." A4Cloud accountability model is based on three layers, *attributes*, *practices*, and *mechanisms and tools*. Literature reviews were used to extract main attributes of the accountability: responsibility, liability, transparency, observability, verifiability, sanctions, provision of assurance and satisfaction of obligations. This paper touches the transparency and verifiability attributes but concentrates mostly on the observability aspect of the accountability. According to the paper organisation needs to have a set of behaviours to be accountable and *practices* layer of accountability model comprises of these behaviours. Practices are split into four categories: practices to define the governance model, practices to implement the model, ways to describe the practices and to show the compliance to regulations, and practices to compensate the failure to properly implement the practices. Mechanisms and tools can be for example software and manual procedures and processes that support the accountability. Felici et al. defines four accountability scenarios in public Cloud based on the actors involved. (Felici et al., 2013) In 2010 Wang et. al, argued that the Cloud storage services should be publicly auditable (Wang et al., 2010). They also suggested that there should be trusted third party entities that would have enough expertise to assess the data security of the Cloud storage services. Data owners could then shift the burden of auditing to those entities.

2.6 Observability in the Cloud

Although an attempt to define observability in the accountability context is provided in Section 2.5 there does not seem to be a clear definition for the word in the research literature in the wider context of IT systems and Cloud Governance. Nevertheless Observability is often mentioned in conference talks, white papers, technical blog posts and tutorials. There seems to be quite a clear understanding in the community of distributed systems specialists of what Observability means in this context. Very simple explanation for the concept of observability can be for example: *The easier it is to make observations of the state and health of the system the greater it's observability is.* Other way to explain observability in the context of distributed Cloud based systems is to describe the elements that can be used to make the system more observable. In the following sections three key concepts of observability: Monitoring, Logging, and Tracing will be explained.

2.6.1 Monitoring

Monitoring distributed Cloud based systems status differs from monitoring the system with the same intended use running on a limited number of servers in a traditional data centre. Traditionally the usage of compute, memory, and storage resources of individual server

instances are important metrics for triggering alarms and in monitoring systems status as whole. When a server runs out of resources that usually entails critical interruptions and errors in system's operation. In Cloud monitoring a single instance still provides important metrics, but instead of triggering an alarm that requires human intervention when some resource is running low, it is usually possible to trigger an automatic scaling task that adds resources to the execution pool. In Cloud environments human action might be required when Autoscaling group is approaching it's maximum allowed instance count or if scaling task fails to add another instance into the resource pool for some reason. Of course a requirement for this kind of approach to work is that individual components and the the overall architecture of the system are designed to support automatic scaling. Cloud environment requires a new kind of approach for the metrics collection and reassessing the situations that actually require human intervention and building alarms based on those.

2.6.2 Logging

In the case of monolithic applications running on single server instances it was often enough to collect logs into the local storage of that instance possibly on duplicated storage and take backups of the local storage. When the system is build from multiple individual services running on varying type of platforms log collection gets a lot more complicated. The application might not run in a server instance the local storage of which the developer would even have access to write or read the logs from. Or the the application might run on an automatically scaled virtual server the local storage of which is lost when the server fails or gets determined as unhealthy and a new server is automatically deployed to replace it. Even if there is a permanent storage for single applications/servers to store their log data it can become really cumbersome to find the relevant logs in the system that consists of larger amount of said applications, servers, and services.

Log collection needs to be centralised, logs should be archived and it should be possible to restore them for further processing and investigation, log collection system should support automated analysis and machine learning, log storage solution needs to scale, support for various log formats, log integrity needs to be preserved, access to the log data needs to be controlled and audit trail for the log access needs to be maintained (Marty, 2011). When the number of components in a distributed system grows the importance of convenient access to the log data grows with it. It becomes paramount to have the means to efficiently filter and query log events, analyse and find the root cause problems. Functional tooling for log analyses becomes even more necessary when it is not possible to pinpoint the origin of unwanted and unexpected behaviour to single service or line of code, but more comprehensive analyses of the events relationships needs to be conducted to find the conditions causing problems in

the system operation (Zhu et al., 2015).

General Data Protection Regulations (European Parliament, 2016) gives some restrictions on the usage and storage of the log data. This thesis does not concentrate on the details of the legislation, but GDPR needs to be taken into account when designing log collection solutions. White paper published by Elastic company (*GDPR Compliance and The Elastic Stack* 2020) gives an overview in layman's terms of what needs to be taken into account when considering GDPR compliancy. Some of the key takeaways from the Elastic company paper are: do not collect personal information without a legitimate need for it, do not store personal information longer than is needed, and if personal information is collected and stored, make sure that the collected information is stored securely. In the log collection context it is good to assess the need for writing personal information into the log data. If the log data does not contain any information that can be used to identify a person then all the GDPR concerns are void. In reality it might be difficult to control the content of the logs and prevent that any personal data from getting into the logs by mistake, hence it is good to handle the log data with care.

2.6.3 Request Tracing

When handling of the end users request or processing some scheduled task, consists of execution path that includes multiple different services, it might become laborious to find the point of failure or source of latency, when things do not work as planned. Request tracing is a technique in which every incoming request gets a distinctive identification code that is carried through the whole execution path. Every application along the path should include the request id when writing logs. Request ids can be used to combine log data from individual applications to build a comprehensive picture of the handling of both individual requests and general working of the system as a whole. Statistical data based on the request traces can be used for example to find bottlenecks in the system (Zhang et al., 2009).

3 Customer Case

In the example customer case a consultancy company have been tasked to improve the observability of a Cloud environment as a part of a governance contract. The customer has chosen Amazon Web Services (AWS) as their Cloud platform provider. The customer is going to use multiple vendors to build their system and to migrate existing services into the Cloud environment. The governance teams responsibility is to design and build top level Cloud architecture, common network infrastructure, identity and access management, and to define security principles, and a way to enforce security best practices.

The customer requires all log and monitor data to be collected into a centralised storage. This includes all the logs that can be collected from modern Cloud platform, access logs, network flow logs, application logs and logs of calls to the service endpoints of the platform provider. Service endpoint call logs can be used to monitor application teams activities and be alerted for security risks caused by misconfigured services. For security and compliance reasons log collection and storage should be implemented in a way that allows application teams to write logs into centralised storage, but do not allow them to delete or overwrite log data. Log data needs to be encrypted on transit and at rest.

Centralised log collection also allows building a comprehensive index of all the log data that is gathered from the system. When all the log data can be queried through single endpoint and possibly through single user interface.

Separation of concerns (Hürsch and Lopes, 1995) and the least privilege principle (Schneider, 2000) were chosen as the main design principles in overall systems architecture. Multi-account model supports both of these goals and it is also recommended in AWS's Well-Architected Framework design guide (Fitzsimons et al., 2018).

Figure 3.1 shows how the system is divided into different accounts:

Master account is responsible for consolidated billing, organisation level configurations and system wide identity and access management.

Business continuity account is used for backups, and log archives.

Operations account is used for monitoring and auditing.

Application accounts are used by development teams to build actual features and services using AWS services, programming languages and frameworks they see best fitting for their needs. Development, testing, quality assurance and production environments are split on to their own accounts

This kind of environment sets up various challenges in observability point of view. Differ-

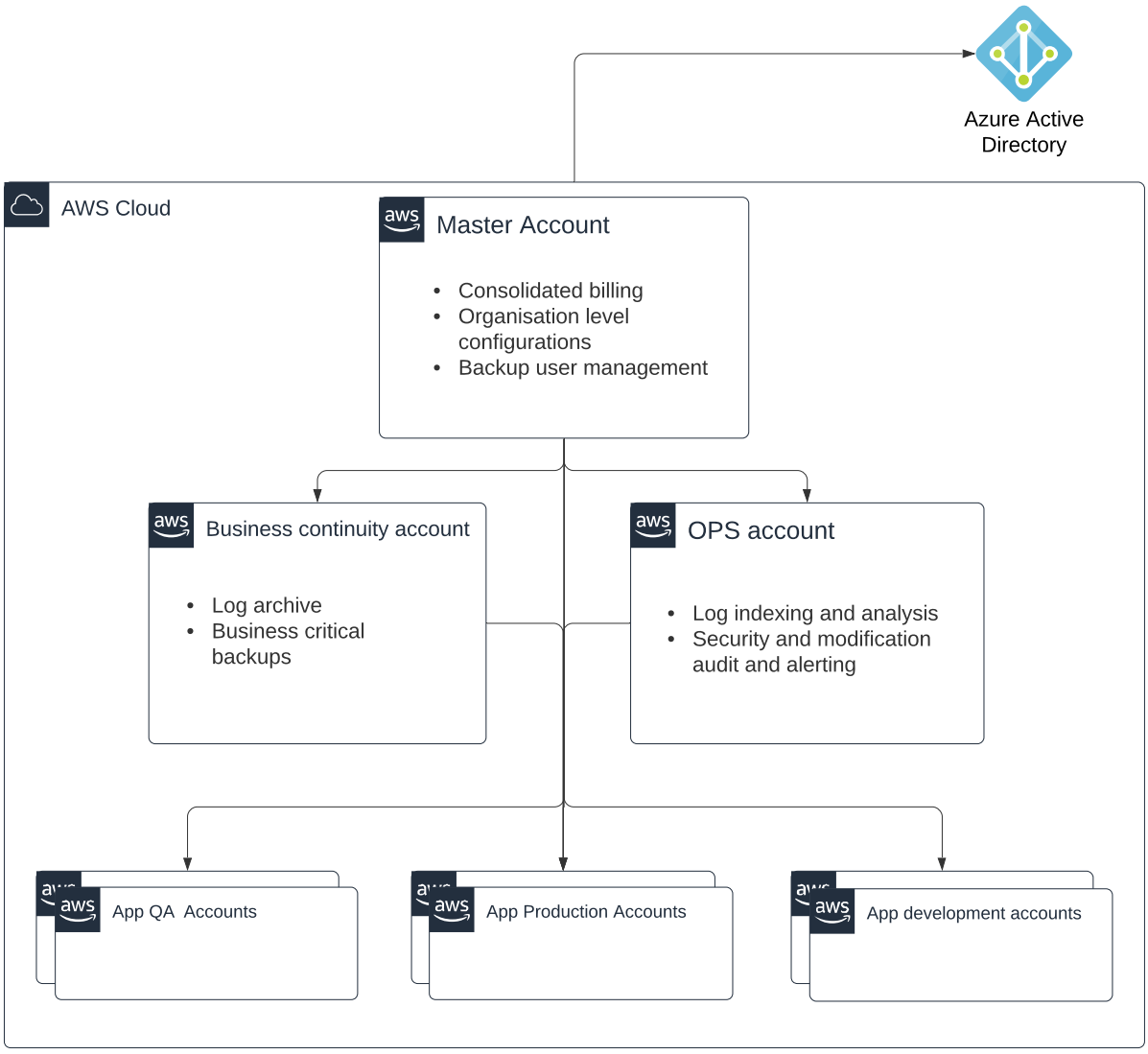

Figure 3.1: Account structure

ent interest groups have different needs regarding monitoring, log analysis, debugging and alerting. Operations and security team need to monitor systems overall health, availability and external threads. Application developers have to be able to debug individual errors. Webscale's responsibility as the governance provider requires monitoring infrastructure and security configurations, developers actions and API calls made to AWS service endpoints. On one hand it is paramount to make the whole system as observable as possible, and the other hand the access to gathered data should be as restricted as possible. Centralised solution needs to provide a top level overview of the systems health and support detailed views into individual components states.

3.1 Application Logs

The customer's system is going to include multiple individual applications written in multiple programming languages using multiple different frameworks and logging tools. The customer does not want to restrict the development teams choices of the used technologies and services as long as they comply with governance security rules.
One big challenge in gathering the log data for centralised processing is to ensure that downstream services can interpret the data correctly. One way to make such interpretation easier is to require applications to write log data in a uniform format. Most of the commonly used logging libraries can be configured to use different logging formats such as unstructured text files with tab or comma separated values or structured JSON or XML files. A problem with unstructured formats is to ensure that application's logging solution was correctly configured and all fields are written in the correct order. For downstream processing unstructured log might seem valid, but there is a risk that some fields get interpreted incorrectly to indicate something they are not indicating. Structured formats are preferable as long as the different logging libraries can be configured to write logs using the same schema. Another possibility is to transform log data into a unified format before it is submitted to analytics and monitoring systems.

3.2 Audit Trail and Security Alerts

As the customers IT governance needs to overlook the Cloud environment, or rather multiple environments on multiple Cloud accounts, that are going to be built by an unknown number of vendors, it is essential to be able to monitor the environments efficiently. Developer actions need to be recorded and archived for auditing purposes. It should also be possible to automate the monitoring and alerting so that any malicious or accidentally harmful actions can be spotted and counter actions be taken to mitigate the damage.
Likewise, monitoring external threats needs to be automated as much as possible. Monitoring data needs to be collected centrally and must be easily accessible in one place for all accounts. When the number of accounts grows it is just not viable to have individual monitoring solutions for each account. Alerts need to be categorised and prioritised. Security events that require immediate human intervention need to be sent immediately. Events related to compliancy and configuration specification and security events that do not require immediate actions can be aggregated into a single alert message and sent for example once a day. Monitoring a system that constantly sends warnings of low priority events can become

annoying and in the end defeat the purpose of the whole alerting system when it becomes impossible to pay attention to every message.

4 Solution

This chapter describes technical solutions and documented guidelines to help example client meet the observability needs.

4.1 Cloud Governance Model

There is a constant stream of headlines in the media about data leaks from this or that Cloud provider's storage systems. After reading the article a bit further it usually becomes clear that the leak is caused by a configuration error by the customer of the Cloud provider and not by a faulty service from the provider.

One way to mitigate such architectural configuration errors is to hire a team of experienced infrastructure architects and administrators and only allow them to handle configuration of any Cloud resources. This kind of approach can easily lead into a stiff development process and prevent the utilisation of the inherent agility of the public Cloud environments. Better approach is to give application developers the responsibility of configuring the Cloud resources they need and let the experienced infrastructure team to support the application teams.

In order to efficiently support multiple application teams it is best to automate as much as possible. Modern Cloud providers have their own solutions to define infrastructure as a code and there are some third party solutions that support multiple Cloud vendors. Infrastructure team should take responsibility of deploying the fundamental resources, which most application teams need and which are most vulnerable for configuration errors, such as network infrastructure. When deployment process is properly scripted it is easy task to build an environment for a new development team from scratch. Besides scripting their own tasks the infrastructure team can provide application teams snippets of scripts to deploy commonly needed resources using best practices. Besides supporting application teams in automating deployment of Cloud resources auditing developers actions is also needed. Cloud providers have managed services to collect audit trails of the api requests done to the configuration API's which can be used to record every change to the system. For example AWS has a managed service called CloudTrail for collecting audit trail log data.

4.2 Centralised Log Collection

To be able to efficiently utilise the log data collected from various applications in complicated distributed environment it is essential that the log data can be centrally accessed. To comply with auditability requirements it is also essential that there is a centralised archive for the log data. Sending logs to the archive needs to be straightforward for the application teams and it can not hinder their workflow too much. AWS's managed solution for collecting, storing and examining logs is CloudWatch. For log collection CloudWatch is probably the easiest solution from a developers point of view when building applications into AWS. For most of the AWS's managed services enabling log collection to CloudWatch logs is just few configuration parameters and for virtual computing free agent with configuration instructions are provided. For log inspection CloudWatch is an OK solution for small systems with only a few applications and a moderate amount of log data, but can become tedious to use when there are multiple applications and a larger amounts of log data. When applications are divided into multiple accounts with developers access restricted only to the accounts they are developing for, using only CloudWatch can become very limiting. To utilise the best part of AWS's core logging solution, that is the easiness and reliability of log collection, logs are first collected in to CloudWatch Log streams. Application teams are advised to collect logs into the application accounts CloudWatch log streams. Every application should use its own log group and every instance of that application its own log stream. If they see it beneficial application teams can use CloudWatch also for other purposes like investigating log data of their own applications or automated alarms based on error count and so on.

4.3 Log Collection Pipeline and Archive

CloudWatch offers a subscription functionality for near real time streaming of log events for further processing and/or storage. This feature is used for sending the log data in to the Kinesis Data Stream in the OPS account. A single Kinesis Data Stream is used for all log groups to avoid the need to make changes into the business continuity account every time a new application development is started. CloudFormation template snippet is provided to application teams to create a subscription that will forward log events into the Kinesis Data Stream residing in the OPS account. Access policy of the Stream is configured to allow CloudWatch service in all of the accounts that are part of the customers Organisation to send log events into the Stream, but not to modify or delete data from the Stream. Kinesis Data Stream can support multiple consumer applications. Differing from the way AWS's

managed queue service SQS works, consumers do not need to remove data from the queue to know it is already processed, instead each consumer has it's own marker to state the point in the stream it has processed, so the consumers can work independently and in parallel.

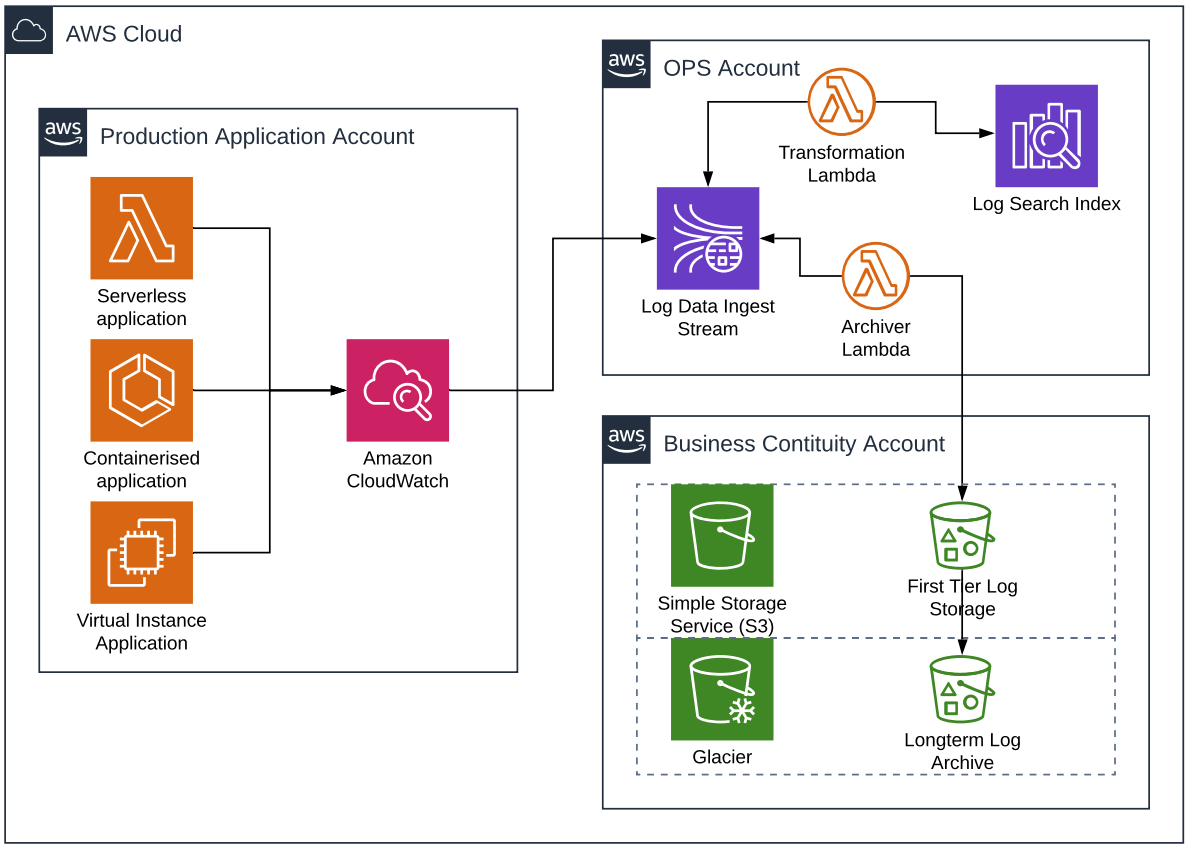

Figure 4.1: Log collection pipeline

Figure 4.1 shows the top level architecture of the log collection pipeline. There are two Lambda functions configured to process the data from the queue. One Lambda is used to write the data into Elasticsearch index and another one handles the archiving of log data. AWS's managed block storage service Simple Storage Service (S3), is an obvious choice for log archive. S3 offers practically unlimited storage space and has fast enough throughput for log archiving usage even for a very large system. Business continuity account was decided to be used to deploy the log archive bucket into. Log data is saved into a single S3 bucket using the name of the log group, stream, and the date when the log entry was written, as path part of the object names, for example:
`/application-1-group/instance-2-stream/2019/12/02/log-file-1`.
Access policy of a log archive bucket is configured to allow the archiver Lambda to write new

objects, but not to modify or delete the objects in the bucket. This is to ensure the log data integrity.

To improve system observability, the log data from multiple different applications and services needs to be easily searchable. Elasticsearch was decided to be used as the solution for the search index. Elasticsearch is an open source search index build using open source search platform Apache Solr which in it self is build on open source search engine Apache Lucene. Kibana is an open source data visualisation dashboard for elastic search. Amazon have productised Elasticsearch and Kibana in to their own managed service offering simply called Amazon Elasticsearch Service.

4.4 Log Index Placement and Access Model

Even though it should not, the log data may contain information that should be well guarded. Even if application developers are instructed not to write any sensitive information into the logs, it is practically impossible for governance team to enforce such limitations and control the content of the log data, hence it is always possible that some secrets or users personal information gets accidentally written into the logs. This Section describes the initial proof of concept design for the log index infrastructure and how the design evolved during the development. The initial design was mostly concentrating on the data security aspects of storing the data in the Elasticsearch cluster. Usability aspects and the need for a fine grained access management dictated that some compromises had to be made for the the data storage security in the final implementation.

4.4.1 VPC Placement

VPC or Virtual Private Cloud is the virtual private data centre, a service or actually collection of services in AWS that is build to offer extra layer of security in the public Cloud. Customers can deploy computing and storage resources into VPC and have control of network routing and security mechanisms between resources, AWS managed services, and the Internet. Network in a VPC can be divided into multiple subnets and subnets can be either public or private. Public subnets have routes to and from internet and private subnets are only visible inside the VPC. In a private subnet traffic to or from internet needs to be routed through a public subnet. VPC is not restricted to one physical location, instead the resources can be deployed on multiple availability zones, which are physically separate datacenters in close geographical proximity. Security groups are a set of firewall rules that can be assigned to the resources in VPC. Security Groups can be used to allow access to instances from other

security groups or IP ranges. One instance can have multiple security groups assigned for it and one security group can be assigned to multiple instances. AWS offers VPC the possibility to access some of their managed services through VPC endpoints. VPC endpoints are gateways that are deployed into the VPC and traffic to the IP ranges of the public endpoints of the managed service is routed to that gateway.

AWS offers two placement options for the managed Elasticsearch Service, in or outside of the VPC. With the added security benefits the in VPC placement seems obvious choice for the search index of log data. Even though development teams are advised not to log any secrets or users' personal data, there is always a chance that some sensitive information slips into the log data by accident.

Figure 4.2 shows the initial design of the log index architecture, it's placement, and con-

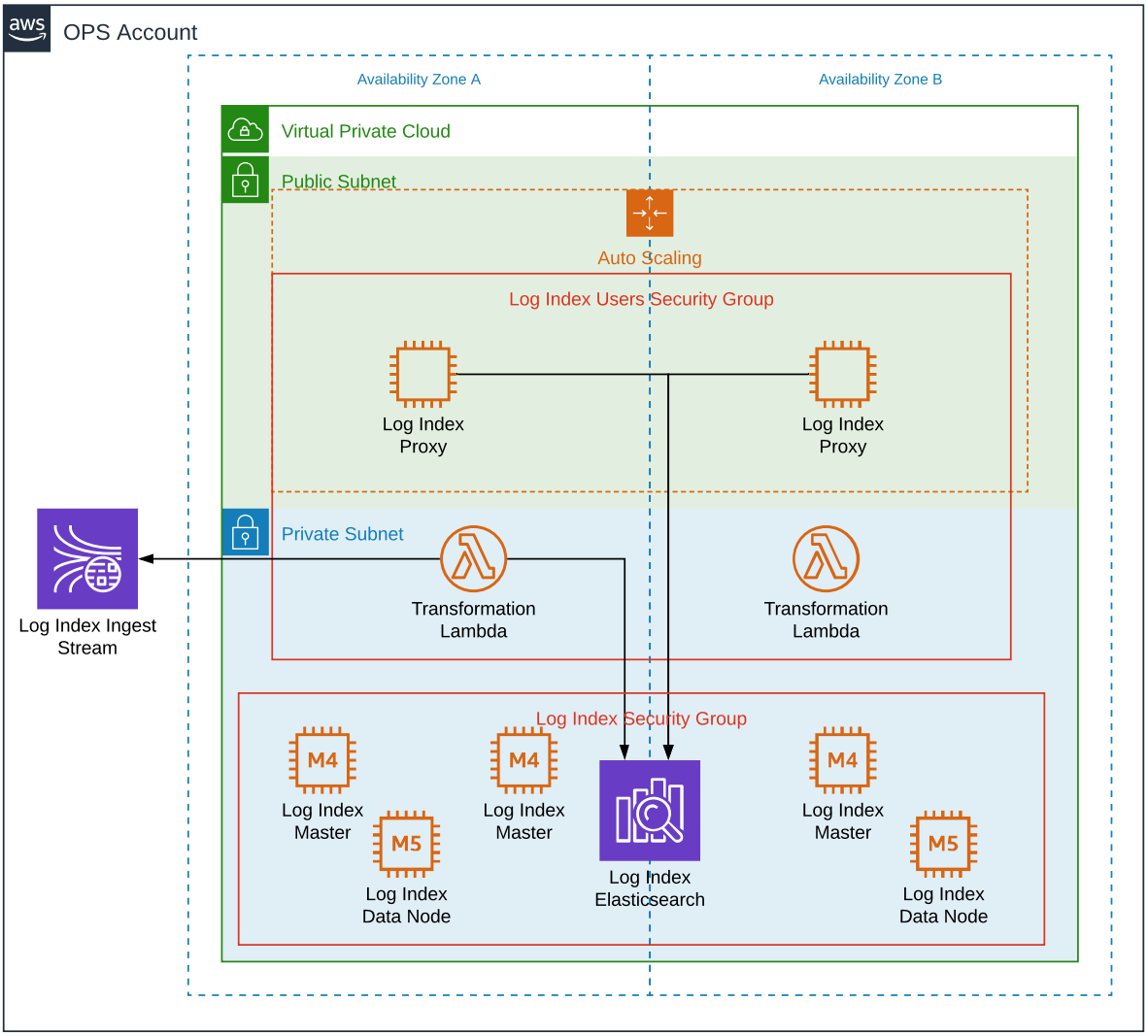

Figure 4.2: Log index placed in virtual private Cloud

nections to the related services. The idea was to place the Elasticsearch instances inside a private subnet and assign one security group for them. Transformation lambdas would run in the same private subnet, but use a different security group meant for the users of the log index. Inbound rule in Log Index security group would allow access only for that security group. Tranformation Lambdas would poll data from Log Index Ingest Kinesis Stream through VPC endpoint, that would allow private access to the stream from the VPC. Proxy instances would be deployed into the public subnet in the same VPC as the Elasticsearch instances as bastion hosts. Security group for log index users would be attached to the proxy instances to enable access to the index through them. Initial idea was to use SSH tunneling through the proxy instances to access log index from the Internet. Another security group was planned to be attached to the proxy instances which could have been configured to restrict access to proxy instances from few trusted IP addresses only.

Biggest problems with the initial design relates to the access management of the Kibana interface. There were few options how to implement and maintain access control with SSH. Easiest solution would have been to use AWS generated SSH keys attached to the proxy instance and distribute them to everyone who needs access to log index. From the security and access control point of view this approach would have been quite bad. It would have been difficult to keep track of all the individuals possessing the keys and would have required to generate and distribute a new key every time developer leaves the project. From the security perspective it would have been best to configure SSH access for every user individually, but this would have created too much maintenance overhead. Every time new developer starts in any of the projects, his or hers access credentials would need to be configured into the bastion host. It is also easy to forget to revoke access from developers when they are leaving a project. Middle ground solution would be to generate keys for every team, but that would still make it difficult to keep count of everyone who have access to the log index.

Customer also wanted to have an option to save logs from different systems into their own indexes and to restrict access to some indexes. SSH approach would not have allowed for a more fine grained access control to different indexes as the access control is based on security groups which work on the IP and the port number level. Everyone who has access to the bastion host has access to all the indexes in the Elasticsearch cluster. Also auditing individual developers actions would have been practically impossible and auditing access to the index would have had to rely solely on the access logs of the bastion host.

4.4.2 Federated Access

To make user and access management more convenient, something more easily configurable was needed. Also there was a preference to use managed AWS services for the access manage-

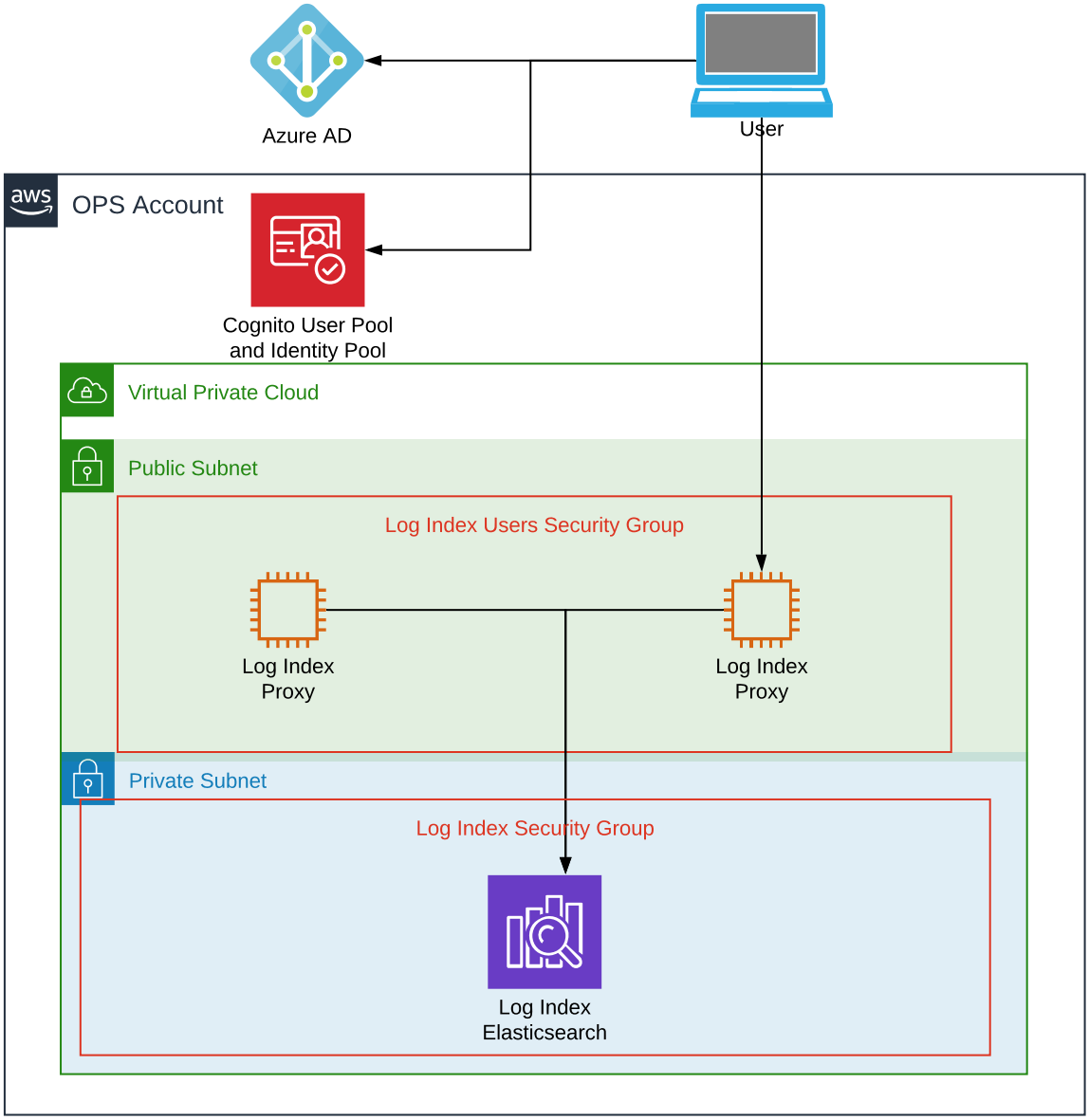

Figure 4.3: Authentication for log index placed in virtual private Cloud

ment. Kibana is served from AWS as a separate application through it's own http endpoint and it is not part of the AWS management console. Kibana authentication is not integrated into the authentication management of the Management Console. Kibana identity and access management requires usage of Cognito. Cognito is the AWS managed service or rather collection of managed services, which main components are Cognito User Pool and Cogntio Identity Pool. Cognito is geared more towards managing end users and their access to the AWS where AWS IAM service (short for Identity and Access Management) is the service that can be used to manage users who need to access the configuration endpoints of AWS

services and to manage programmatic access. AWS version of Kibana provides fairly easy integration with Cognito. When creating the Elasticsearch domain with Cloudformation, previously configured Cognito Idetity and User Pools can be given for the Elasticsearch service and let it configure Cognito client application for authentication and redirection to and from Cognito authentication when accessing Kibana. The example customer already have an Active Directory in use for managing their internal users and outside contractors. Their Active Directory is deployed in Azure Cloud and it already is used as federated identity provider for AWS console access. It was the preferred identity source to be used for Kibana authentication too. As Kibana is not part of the AWS Management Console, it does not recognise the same access token that is used for AWS console access so the same authentication flow that is already in place for other AWS resources can not be used. Cognito on the other hand can use Azure AD as an identity provider so it was decided to use Cognito to authenticate Kibana users and Azure AD as an identity provider for Cognito.
Initial design was changed so that the Bastion hosts were replaced with proxy instances running NGINX server. NGINX was configured to pass requests to Kibana and Cognito based on the request paths. This approach worked fine with Kibana and Cognito when trialed with users added to the Cognito User Pool. Problems rose when Azure AD was used as an identity provider and Cognito redirected authentication to the Azure AD client. It was impossible to configure Azure AD to redirect back to the Proxy address after successfull authentication. Security mechanisms in Azure allowed redirection only to the URL defined by the application that originated the authentication request which was the URL of the Cognito Application client. Next attempt was to configure NGINX to proxy Azure requests too so that all requests in the authentication flow would go through the proxy and redirection URLs in responses could be rewrote to point to the proxy. This approach might have worked otherwise, but cross reference rules of the Azure Client did not allow any requests where origin IP address did not belong to Azure.

4.4.3 Reassessing Log Index Placement

At this point it started to feel like VPC placement would complicate authentication flow too much and without a convenient solution for accessing Kibana, storing the log data into the Elasticsearch would become meaningless. Therefore assessing the sensibility of VPC placement was needed. One of the benefits of the VPC placement is the convenience of using Security Groups for access management, but when fine grained IAM based management is needed this security group based access management becomes redundant. After counting the plusses and the minuses, the VPC placement was ditched after all and the cluster was placed outside of the VPC. This made the access management clearer and usage of Kibana more

convenient.

4.4.4 Final Access Management Pattern

AWS does support IAM based access management per index in the Elasticsearch, but to our surprise Kibana will not work properly if access is restricted on the index level in IAM role policies. The access policy of the IAM role needs to allow http access to the whole Elasticsearch domain and fine grained access management have to be done using Elasticsearch's own access management. IAM role can be mapped to Elasticsearch access management systems external role entity and finegrained access policies can be attached to the external role. Fine grained access management in AWS Eleasticsearch implementation is so new feature that it does not have CloudFormation support and it can not be enabled for already running cluster. In order to get the fine grained management to work, old cluster had to be undeployed and new cluster needed to be deployed without CloudFormation. A Python script was created to handle the deployment so that the deployment process is easily repeatable and the cluster updatable. Log events are divided into indexes based on the id of the account the event originates from. IAM roles are used to give access to the specific index and the Kibana plugin. Roles can be assumed by authenticated Cognito users. Cognito is configured to use Azure AD federated identities and authentication flow.

Authentication Flow

Following describes the authentication flow to access the Kibana application and log data in Elasticsearch service, using Azure Active Directory as the federated authentication and identity source.

1. User makes a request for the Kibana Plugin.
2. Elasticsearch Service redirects the user to the Cognito app client.
3. User chooses to authenticate with Azure AD.
4. Azure authentication page opens.
5. When authentication succeeds Azure redirects back to Cognito
6. Cognito creates an external user into the Userpool and maps the user attributes from SAML claim. If the user already exists, Cognito updates user attributes if necessary.

7. Cognito Identitypool assumes an IAM role for the user based on the user's group id attribute, creates an access token for the user, and redirects back to Kibana.

8. Access to different indexes and Kibana/ELastisearch features is decided based on how the IAM role is mapped to different access policies in the Elasticsearch.

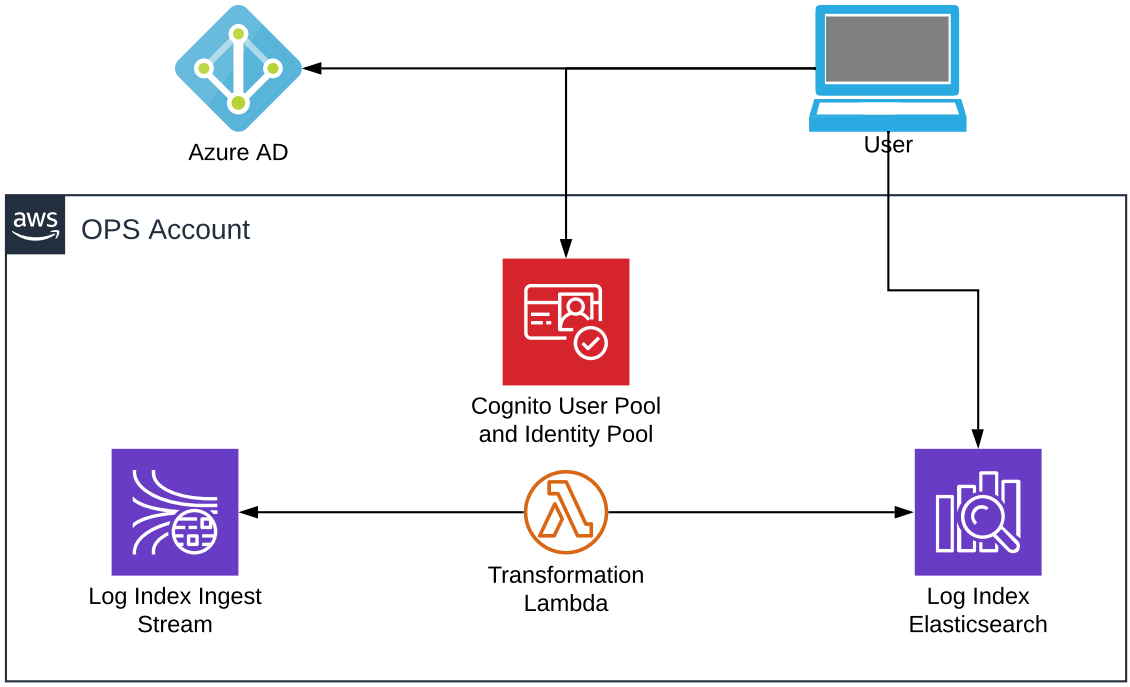

Figure 4.4: Log index without virtual private Cloud

4.5 Application Logging Guidelines

4.5.1 Logging Restrictions

Two main restrictions on what data can be collected into logs are systems security and handling of users personal data.

Security credentials

Logs should never contain any secrets like passwords etc. that can compromise systems security if they fall into the wrong hands. Access to the centralised log storage will be restricted

and monitored, but as there are multiple teams building the system and accessing the logs one has to pay attention to never log any secrets.

Personal data

When logs are collected from applications that handle users personal data, and especially when logs are collected into a centralised repository, extra care should be taken to not build an illegal person registry. EU's general data protection regulations (European Parliament, 2016) provide guidelines for considerations when collecting user related logs. Logs should not contain any data that can be used to identify the user. It is a good practice to never log request body as is, but always to parse only the relevant data and exclude everything that should not be written in to the centralised log storage.
Do not log:

- Request body
- Any passwords, API keys, certificates, etc
- Users personal information
- Full names
- Social security numbers
- Email addresses
- Postal addresses
- Phone numbers
- User's mobile device coordinates
- Credit card or banking information
- Anything that can be used to identify the person

The following can be logged when necessary: User's IP addresses, User names

4.5.2 Application Log Collection

All the production application logs need to be collected into the log archive and indexing service. Governance team will provide Cloudformation template snippet to enable subscription

of applications Cloudwatch log group into a centralised log handling system. Application teams responsibility is to collect all the logs into Cloudwatch and to enable the subscription. Application logs should be collected using asynchronous libraries and tools to minimize the effect on applications normal operation. In EC2 instances logs should be written in local files and the Cloudwatch agent should be used to deliver logs to CloudWatch. ECS and Lambda provides tooling to deliver logs to Cloudwatch. Collecting log data from third party libraries should be avoided as that is usually not very useful and can easily clutter the log files. All exceptions from third party libraries should be caught and logged as needed.
To be able to process logs downstream unified logging format is needed. As applications are written in different programming languages using various logging tools, it is not practical to force every application team to follow a very strict logging format, but it is best to try to unify the logging format as much as possible. When exact schema is difficult to guarantee it is best to use a semi-structured format. Wrong order, missing, or extra values in formats like CSV or TSV formatted logs can lead to log interpretation mistakes that are not easily noticed and may result in errors in analysis and automated actions based on log data. Most of the commonly used libraries can be configured to write logs in JSON or XML format, from which JSON is more lightweight and less fragile. If development teams preferred logging tool can not be configured to log all the defined fields, message field can be used to log a JSON object that includes those fields. Logging tools for different languages, that can be configured to write custom fields include:

- Javascript/Typescript: Winston[1]
- Java: LogBack[1] with logstash-logback-encoder[2]
- Python json-logging-python[3]

4.5.3 Recommended Log Schema

Log entries should provide enough context to be useful. In the case of ERROR or FATAL level log entries the goal is to make it as easy as possible to start solving the problem. Entries should answer at least the following questions:

- When the error happened?
- Where the error happened?

[1]https://github.com/winstonjs/winston
[1]http://logback.qos.ch/
[2]https://github.com/logstash/logstash-logback-encoder
[3]https://github.com/thangbn/json-logging-python

- What was the application trying to achieve when the error occurred?
- What was the root cause of the error outside of the application logic?
- Who or what initiated the call chain that caused the error?

```
{
  "timestamp": "number, required",
  "level": "string, required",
  "message": "string, required",
  "file_name": "string, required",
  "function_name": "string, required",
  "stack_trace": "string, not required",
  "x-amz-cf-id": "string, not required",
  "x-amzn-trace-id": "string, not required",
  "version": "string, not required",
  "session_id": "string, not required"
}
```

Figure 4.5: Suggested log schema

Figure 4.5 shows the suggested log schema in JSON format. More indepth explanations of the fields can be found below.

Timestamp (timestamp)

Although the Cloudwatch log entry contains timestamp of the time the entry was delivered into the Cloudwatch stream, application should add a timestamp of the time the log is written. Timestamp field should contain UNIX epoch time as a number.

Severity levels (level)

Log entry should always contain the severity level information. For debugging purposes the correct severity level information might not feel necessary, but for a centralised log analysis and alerting solution it is mandatory. Universally used log levels in descending severity order are FATAL, ERROR, WARN, INFO, DEBUG and TRACE. Further explanation for the usage of severity levels is provided in Subsection 4.5.4

Custom content (message)

Log entry should always contain the message field. The message field should describe the reason why the log entry is written and provide as much additional context as possible. When a log entry otherwise complies with the schema message field's type is free formatted string. If an application team decides to use a logging library that can not be configured to write custom field to fulfil the above described schema, the message field can always be used to write custom JSON strings. Downstream parsing of log events is configured to first read the value from the root of the log entry and if it is not present, then revert to interpreting the message field as a JSON object and to read the field from there.

Source code location (file_name, function_name)

Especially for debugging purposes it is important to make it as easy as possible to locate the spot in source code where the log entry originates from. Some logging libraries provide automation to write this information into a "logger" or a similarly named field, but for easier processing in downstream services it is better to separate the information into two fields.

Exception origin (stack_trace)

In case of ERROR and FATAL level logs a stack trace should be included in the log entry if the reason for the exception can not be easily derived from the data and told in the message field.

Request correlation and tracing (x-amz-cf-id, x-amzn-trace-id)

Correlation identification can be used for debugging and cost optimisation purposes in distributed system where one incoming request or event can cause a call chain that goes through multiple services. Different services in AWS add and/or updates request headers that can and should be used to correlate log entries from various applications and services. X-Amz-Cf-Id header uniquely identifies every request coming in to the system through CloudFront. This header is added to the request before it is forwarded to the origin. This can be used to correlate logs coming from request handling applications with Cloudfront access logs. Cloudfront does not add X-Amzn-Trace-Id header so X-Amz-Cf-Id header is needed to correlate the whole request chain.
X-Amzn-Trace-Id header is the best option for tracing requests in the AWS environment. It is added and updated by Application Load Balancer (ALB) by default and other services

such as API Gateway can be configured to handle the trace-id header. When request first arrives to ALB and X-Amzn-Trace-Id header is not present it is added with ”Root” field. When request where X-Amzn-Trace-Id is already present arrives to ALB, it either adds or updates the ”Self” field depending on if it is already present or not. X-Amzn-Trace-Id has the following format: Field=version-time-id, where ”Field” is the name of the field. ALB supports ”Root” and ”Self” field names. Applications can add custom fields and ALB preserves them as is. ”Version” is the version number, ”time” is the UNIX epoch time, and ”id” is the trace identification.

Source code version (version)

Version field can and should if possible be used for identifying the version of the source code that was running when the log entry was written. This can be a git hash or if the application team is using a unique version number for every deployment that can be used. Knowing the exact version of the source code helps especially in debugging and can also be used for various analysis purposes.

User session identification (session_id)

When request that causes the log entry to be written is part of an authenticated session the session id should be included in the log entry.

4.5.4 Severity Levels Explained

Following instructions were given to the application teams on how to use different severity levels when writing logs.

FATAL

is used for errors and exceptions that application can not survive and requires unexpected reboot. FATAL should not be used if application can handle the exception and continue normal operation.

ERROR

is for exceptions that disrupt the normal operation of the application and prevent processing the request in the desired manner. ERROR level log should be used when the application

can continue normal operation after the exception is handled even if the request that caused the exception can not be handled successfully.

WARN

is for warning about unusual behaviour or caught exceptions that does not prevent the application from continuing to handle the request. WARN should not be used to log about anticipated exceptions that are happening often and can be thought of as a part of normal operation.

INFO

level logs should be used sparingly to collect only relevant information regarding programs normal execution. INFO is the first level that should be collected in production environment so extra care should be used not to clutter logs with unnecessary information and to keep logging expenses on an acceptable level. Things that should be logged as INFO level entries include for example startup and configuration information when application starts and some important steps in program execution that do not occur too often. Depending on the application, for example start of the user session and database writes might be useful information to log. Information about every read request is usually not needed.

DEBUG

is for debugging purposes, which require a bit longer period to collect logs. In production DEBUG level log collection should be switched off by default, but can be used to collect logs of some specific problem that does not arise at development/testing environment. Depending on the application DEBUG level collection can be on all the time in the development/testing environment.

TRACE

can be used when a lot of data points needs to be logged, for example to trace every step of executing some complicated algorithm or if it is necessary to log every iteration in loop. TRACE level logs should not be collected for more than a short period of time even in testing/development environments. TRACE is meant for logging purposes that produce so much log data it can severely affect applications performance and incur serious storage expenses.

4.6 Centralised Security, Audit, and Compliancy Monitoring

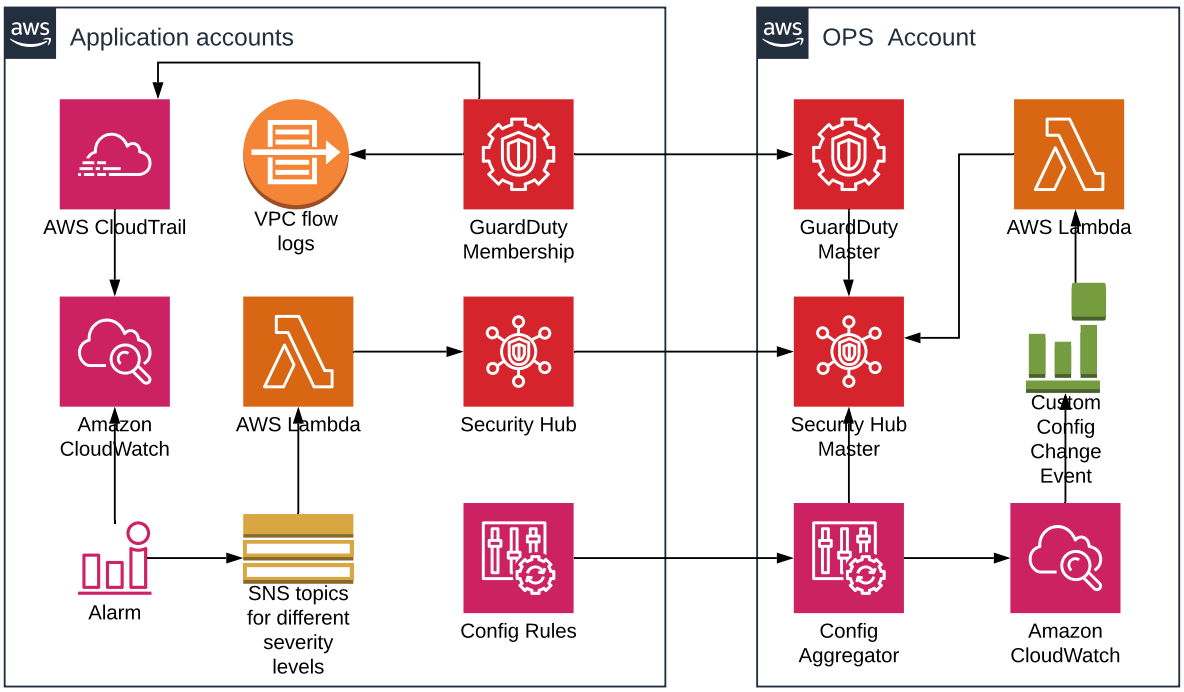

Figure 4.6: Architecture for centralised configuration, security and audit monitoring

Audit logs are collected using AWS managed service CloudTrail. CloudTrail can be enabled on the account basis and it collects by default logs of the calls made to the configuration APIs of almost every AWS service. CloudTrail can be used to collect records of all the developer actions in the AWS environment as all of the configuration changes happen through the same APIs, be it changes made through web user interface called management console, CloudFormation or third party infrastructure management tools, or AWS SDKs.

Config is the AWS managed service for continuous monitoring and auditing of how the configurations on the accounts comply with the governance rules set by the organisation. Config detector can be set to send CloudWatch events when it detects changes in the configurations that do not comply with the defined Config rules. GuardDuty is the AWS managed service for monitoring malicious and unauthorised activities in the customers Cloud Environments. AWS Security Hub is the service for centralised access to the findings of different security services in AWS. Security Hub can be used to configure alarms and monitor the findings form other security services.

In the example customers environment CloudTrail is enabled for every account by the gover-

nance team and application teams do not have access to CloudTrail configurations. CloudTrail findings are monitored automatically with GuardDuty which is also enabled in every account. There are also manually configured CloudWatch metrics and CloudWatch alarms based on those metrics based on the logs CloudFormation sends into CloudWatch. GuardDuty findings are collected locally by in every account and sent to the GuardDuty master residing in the OPS account. Notifications generated by the custom alarms based on CloudTrail logs are send into the SNS topic and Lambda Handler is configured to be invoked by the SNS topic. The Lambda handler forwards the Custom findings into the Security Hub in the application account. Config monitors application teams compliancy to the Configuration rules the governance team and AWS have defined. Some of the AWS provided rules are disabled as unfitting to the Environment and custom rules are used to fill in the caps in AWS rules and to provide environment specific rules. Each accounts Config sends its findings into the Config aggregator in the OPS account. In the OPS account GuardDuty Master and Config findings based on the default config rules provided by AWS are collected into the Security Hub. There are also CloudWatch metrics configured to monitor Configuration changes based on custom config rules. Custom config rules based findings can not be automatically forwarded into Security Hub, hence there is Lambda function processing them and adding them into Security Hub. Configuration monitoring data from Config is also collected into the Security Hub.

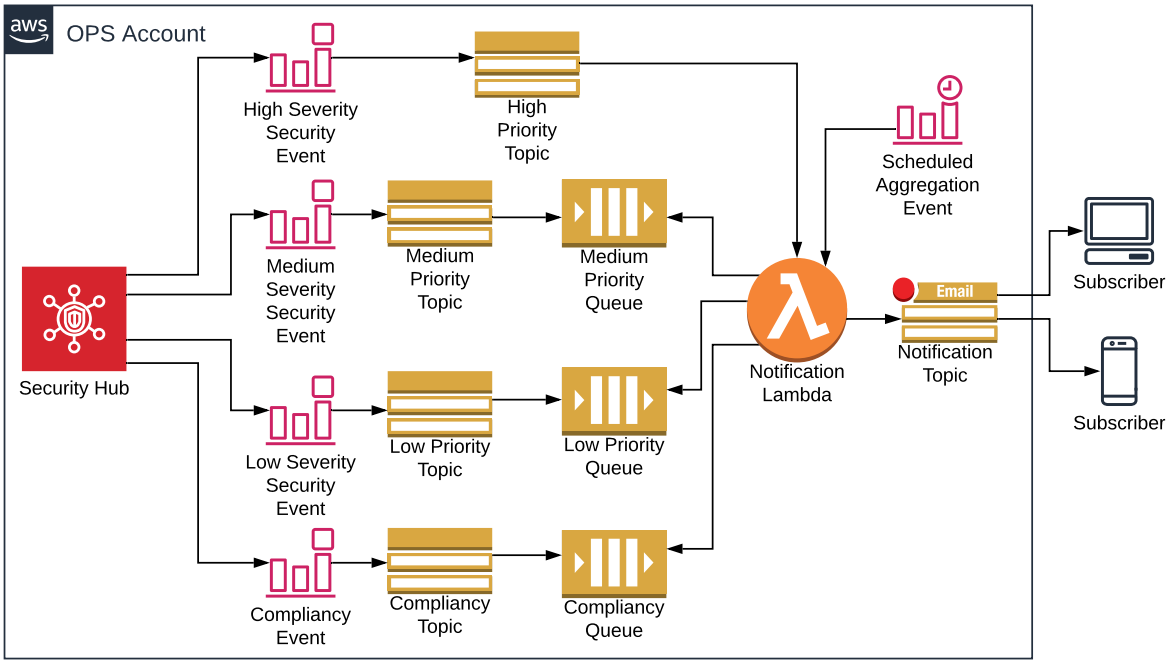

Figure 4.7: Architecture for centralised security alerts

Architecture for security and compliancy findings based alerts and notifications is described

in Figure 4.7 New findings in Security Hub generate events in AWS. In the example architecture those events are forwarded into the SNS topics. There are four SNS topics depending of the event's severity level and type. For security events there is high, medium, and low severity topics and for compliancy events only one topic. Medium and low priority security events and compliancy events are forwarded from SNS topics to SQS queues to wait for further processing. High severity SNS queue is configured to invoke the processing Lambda immediately. Lower severity security and compliancy events are aggregated daily from the SQS queues. Processor Lambda function formats the data from the events to more readable format, adds the information if the event is new or recurring one, generates links to the Security Hub so that it is easy to get to find the event in question, and forwards the generated message into yet another SNS topic. People responsible of governance and other interest groups, such as customers administration, can subscribe to that SNS topic to receive notifications about the Security Hub events via email. It could also be configured to send notifications via SMS message, but currently only email is used.

4.7 Centralised Audit Log Archive

Just like application logs the audit logs needs to be collected into the centralised archive. Figure 4.8 shows the overview of the architecture for collecting audit trail logs into the centralised archive. Every account is configured to collect CloudTrail logs and CloudTrail on every account is configured to send the logs into a single S3 bucket residing in Business continuity account. VPC flow logs and Config Detector findings are also collected on every individual account and forwarded into the S3 bucket residing the in business continuity account. Cloudtrail log bucket is encrypted with KMS key that has very restricted access policy for decryption. Currently no user can use the key to decrypt the data in the centralised CloudTrail archive, but users with administrator status can change the access policy of the key so that they can use it for decryption if a need arises. Logs of the accesses to the centralised CloudTrail bucket are collected into a different S3 bucket which is encrypted with a different key.

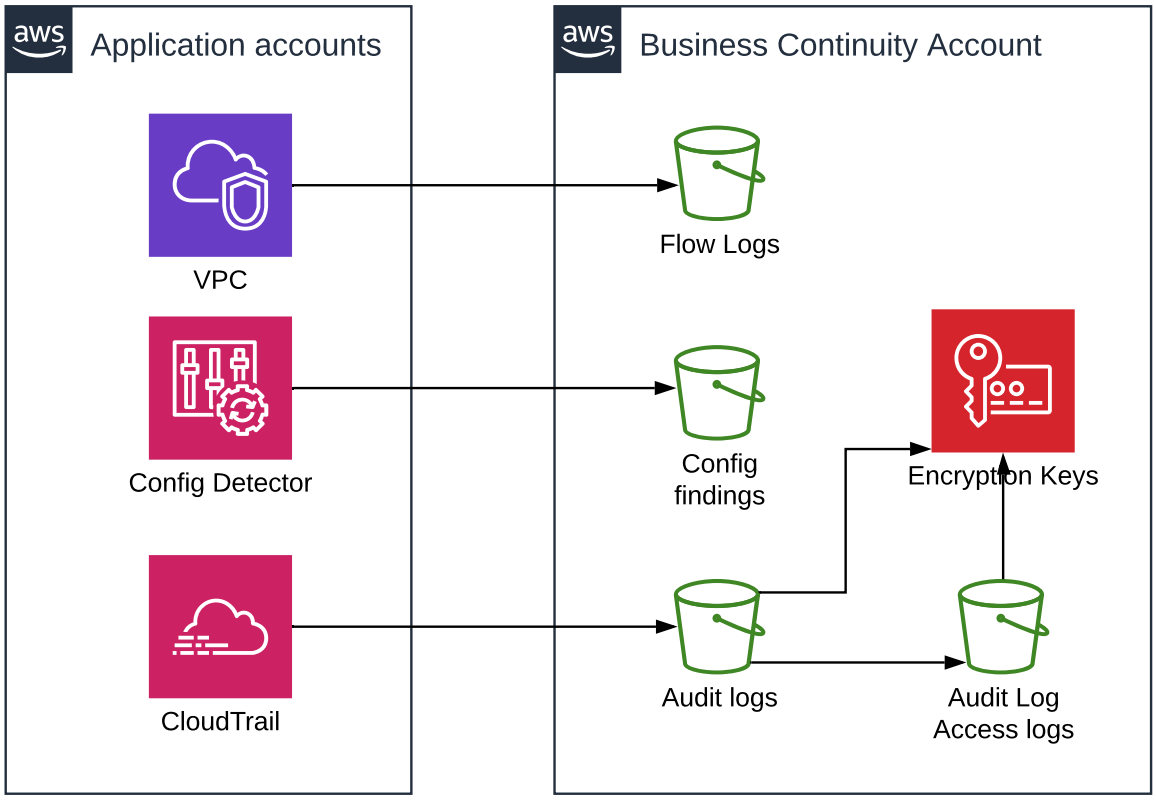

Figure 4.8: Architecture for centralised audit log archive

5 Conclusion

Having a comprehensive overview of what is happening in the computing environment is essential to succeed in IT-governance. Cloud providers offer many tools for making observations of the different services in customers environment. It is easy to make observations of individual services with the readily available tools, but getting an overview can be more difficult. Centralised search index for the application log data should greatly improves the usefulness of the collected data in an environment that consists of multiple accounts. Automated auditing of the configurations have already proven its value and many problems in configurations that might have caused issues later have been identified and fixed. There have not been any actual security breaches that the automated security monitoring system could have detected and created alerts, but there have been alerts for example from unauthorised access attempts, when some developer have forgot to open the VPN connection before trying to access the environment.

5.1 Evaluation of the Selected Approach and Used Technologies

The solution that was build to increase the observability of the Cloud environment of the example customer has already shown its value. Especially the tools used for the detection of shortcomings in the configurations have already given a lot of useful insights. Unfortunately the collection of application logs have only been started and it is not possible to properly evaluate the usefulness of the centralised log index at this point.
Deploying and managing the AWS version of Elasticsearch is not as straight forward as some of the other AWS managed services. Elasticsearch service is not as well documented as the core services in AWS. AWS also has a tendency to use the documentation as a kind of marketing material and it is often not clearly stated when something is not possible with the service. This lead to some misinterpretations when the log index was designed. It seemed clear that the access management could be done using IAM roles and that the VPC placement meant that the instances of the cluster would actually be placed inside the clients VPC. AWS does not provide dedicated tooling to automate the scaling of the Elasticsearch cluster. For the programmatic access native access management solutions of AWS, like IAM and SecurityGroups, are sufficient, but when fine grained access is needed for users accessing

the data through Kibana, the access management have to be done in Elasticsearch. Elasticsearch based access management can not be configured using CloudFormation, therefore the configuration needs to be done either manually, using Kibana, or with custom scripts, calling Elasticsearch API.

To reduce the maintenance overhead of the Elasticsearch cluster, it would be good to have automated scaling for the underlying computing and storage resources. By the time this thesis was finished the log indexing solution that was built for the example customer did not include any form of automated scaling of the computing or storage resources. The metrics reflecting the health and state of the Elasticsearch cluster are collected into CloudWatch and there are automated alarms based on the metrics. For example when the remaining free disk space gets under the threshold, it will trigger an alarm and the system sends notification to the governance team. Actual scaling still has to be done manually. It should not be too difficult to use the alarms to trigger the automatic scaling procedures in the future. Currently there are only a few applications sending log data into the Kinesis stream for indexing. As the number of applications and development teams grow it is to be seen how well the developers can comply to the logging guidelines. If the number of different log formats grows, the maintenance overhead of the transformation Lambda can become an issue.

Managed services for automatic monitoring and formal reasoning in AWS are still rather new and constantly evolving. Evolvement is a good thing, but changes in the behaviour of those services produced a few surprises during the development of the audit and security notification service. For example AWS changed the handling of the severity levels of the Config findings during the process. When the development started the Config finding were always classed as low severity level findings in SecurityHub and only security findings from had more severity levels. Therefore there was no need separate the handling of the different type of findings in notification service. Notifications about Config findings were sent out with the same priority as low level security findings. At some point AWS changed the behaviour and started to use more severity levels for the Config findings too. This required changes in the handling of different types. Even high severity level Config findings do not require as urgent reaction as high severity level security findings, therefore separate notification pipelines were needed for different types of findings.

5.2 Further Development

In the future it will be interesting to see how the log index can be utilised to serve the needs of different application teams. It would also be interesting to combine the application log data with different metrics collected from the system and build machine learning models to

detect anomalies and to predict the potential problems in the system.
Currently there is no automated mitigation procedures that could be launched based on the Security events, but that seems something worth investigating and prototyping in the future. There could also be a scheduled cleaning process or multiple processes which would undeploy resources that do not comply with the Config rules. Even better would be if AWS updates the Config service to support the rule checking at the time of the deployment, so that, if the configuration does not comply to the rules, the deployment would be prevented.

Bibliography

Backes, J., Bolignano, P., Cook, B., Gacek, A., Luckow, K. S., Rungta, N., Schaef, M., Schlesinger, C., Tanash, R., Varming, C., et al. (2019). "One-click formal methods". In: *IEEE Software* 36.6, pp. 61–65.

Borgman, H. P., Bahli, B., Heier, H., and Schewski, F. (2013). "Cloudrise: Exploring Cloud computing adoption and governance with the TOE framework". In: *2013 46th Hawaii international conference on system sciences*. IEEE, pp. 4425–4435.

Brandis, K., Dzombeta, S., Colomo-Palacios, R., and Stantchev, V. (2019). "Governance, Risk, and Compliance in Cloud Scenarios". In: *Applied Sciences* 9.2, p. 320.

Catteddu, D., Felici, M., Hogben, G., Holcroft, A., Kosta, E., Leenes, R., Millard, C., Niezen, M., Nunez, D., Papanikolaou, N., Pearson, S., Pradelles, D., Reed, C., Rong, C., Royer, J., Stefanatou, D., and Wlodarczyk, T. (May 2013). "Towards a model of accountability for cloud computing services". English. In: *Proceedings of the DIMACS/BIC/A4Cloud/CSA International Workshop on Trustworthiness, Accountability and Forensics in the Cloud*.

Chebrolu, S. B., Bansal, V., and Telang, P. (2010). *Top 10 cloud risks that will keep you awake at night*. URL: https://www.owasp.org/images/4/47/Cloud-Top10-Security-Risks.pdf (visited on 03/01/2020).

Cook, B. (2018). "Formal reasoning about the security of Amazon Web services". In: *International Conference on Computer Aided Verification*. Springer, pp. 38–47.

Dutta, A., Peng, G. C. A., and Choudhary, A. (2013). "Risks in enterprise cloud computing: the perspective of IT experts". In: *Journal of Computer Information Systems* 53.4, pp. 39–48.

European Parliament (Apr. 27, 2016). "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* 119. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG#d1e40-1-1 (visited on 01/26/2020).

Felici, M., Koulouris, T., and Pearson, S. (2013). "Accountability for data governance in Cloud ecosystems". In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Vol. 2. IEEE, pp. 327–332.

Fitzsimons, P., Shinn, B., Elmalak, S., and Potter, B. (2018). *Security Pillar, AWS Well-Architected Framework*. White paper. Amazon Web Services. URL: https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf.

GDPR Compliance and The Elastic Stack (2020). White paper. Elastic.co. URL: https://www.elastic.co/pdf/white-paper-of-gdpr-compliance-with-elastic-and-the-elastic-stack.pdf (visited on 01/26/2020).

Gonzalez, N. M., Rojas, M. A. T., Silva, M. V. M. da, Redıgolo, F., Brito Carvalho, T. C. M. de, Miers, C. C., Näslund, M., and Ahmed, A. S. (2013). "A framework for authentication and authorization credentials in Cloud computing". In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, pp. 509–516.

Hürsch, W. L. and Lopes, C. V. (1995). *Separation of Concerns*. Tech. rep. College of Computer Science, Northeastern University Boston, MA02115, USA.

Marty, R. (2011). "Cloud application logging for forensics". In: *proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, pp. 178–184.

Mell, P., Grance, T., et al. (2011). "The NIST definition of cloud computing". In:

Schneider, F. B. (Feb. 2000). "Enforceable Security Policies". In: *ACM Trans. Inf. Syst. Secur.* 3.1, pp. 30–50. ISSN: 1094-9224. DOI: 10.1145/353323.353382. URL: https://doi.org/10.1145/353323.353382.

Swimmer, M., Yarochkin, F., Costoya, J., and Reyes, R. (2020). *Untangling the Web of Cloud Security Threats*. White paper. Trend Micro Research. URL: https://documents.trendmicro.com/assets/white_papers/wp-untangling-the-web-of-cloud-security-threats.pdf.

Trend Micro Security Predictions for 2020 (2019). White paper. Trend Micro Research. URL: https://documents.trendmicro.com/assets/rpt/rpt-the-new-norm-trend-micro-security-predictions-for-2020.pdf.

Wang, C., Ren, K., Lou, W., and Li, J. (2010). "Toward publicly auditable secure Cloud data storage services". In: *IEEE network* 24.4, pp. 19–24.

Yang, Y., Chen, X., Wang, G., and Cao, L. (2014). "An identity and access management architecture in Cloud". In: *2014 Seventh International Symposium on Computational Intelligence and Design*. Vol. 2. IEEE, pp. 200–203.

Zhang, Z., Zhan, J., Li, Y., Wang, L., Meng, D., and Sang, B. (2009). "Precise request tracing and performance debugging for multi-tier services of black boxes". In: *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pp. 337–346.

Zhu, J., He, P., Fu, Q., Zhang, H., Lyu, M. R., and Zhang, D. (2015). "Learning to Log: Helping Developers Make Informed Logging Decisions". In: *Proceedings of the 37th Inter-*

national Conference on Software Engineering - Volume 1. ICSE '15. Florence, Italy: IEEE Press, pp. 415–425. ISBN: 9781479919345.