

Bachelor's thesis

Bachelor of Engineering, Information and Communications Technology

2023

Joni Lehtinen

TECHNICAL REVIEW SETUP FOR AMAZON WEB SERVICES

– Assessing Amazon Cloud Computing Service
Configurations



Bachelor's Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2023 | 57

Joni Lehtinen

TECHNICAL REVIEW SETUP FOR AMAZON WEB SERVICES

This thesis provides a comprehensive technical review setup to audit various Amazon Web Services (AWS) services. The objectives were to describe the functionalities and features of important AWS services and to assess their cost, and security posture through automation and a manual visual auditing process. A thorough examination of official AWS documentation and relevant resources was conducted to gain a comprehensive understanding of the services and their security implications. Automation workflow was then setup to audit the cost and security configurations of the AWS services to identify potential vulnerabilities and misconfigurations with scanning tools. After the completion of the technical review, manual auditing can be conducted by reviewers to identify critical misconfigurations, cost issues, and vulnerabilities from a visual data template. Cloud services at present have hundreds of products and numerous rule sets available, meaning that establishing a functional work environment can be challenging. The simplicity of deploying cloud services can lead to the potential for mistakes, including incorrect configurations and security issues that may result in significant financial losses. Automation workflows are essential to enable the collection and auditing of critical configuration data through a reliable process.

Keywords: Amazon Web Services, cloud security, cloud configuration management, vulnerability assessment, misconfiguration detection

Contents

List of abbreviations	6
1 Introduction	8
2 Technical Review Foundation	9
2.1 Cloud Infrastructure	13
2.2 Microservices	16
2.3 Configuration Auditing Tools	18
3. Amazon Cloud Computing Services	21
3.1 Essential Services	22
3.1.1 Athena	22
3.1.2 CloudFormation	22
3.1.3 CloudTrail	23
3.2 Data Protection	31
4 Technical Review Assessment	32
4.1 Assessment Requirements	32
4.2 Service Assessments	34
4.3 Review process	44
4.4 Configuration Presentation and Auditing	46
5 Conclusion	51
References	52

Appendices

Appendix 1. Configuration Templates

Figures

Figure 1. Global Damages caused by cybercrime, from Ref. [6].	10
Figure 2. Directory data and file storage format.	11
Figure 3. AWS data gathering and analysis workflow.	12
Figure 4. Visualization format for auditing.	12
Figure 5. Linux malware yearly growth, from Ref. [9].	14
Figure 6. EC2 vulnerability scanning and patching workflow.	15
Figure 7. Microservice and monolith architectures, from Ref. [17].	17
Figure 8. Top cloud threats to audit with security tools, from Ref. [22].	19
Figure 9. Config flagging resource and auto-remediation workflow.	25
Figure 10. ECR Repository storage for application images, from Ref. [52].	26
Figure 11. EKS Workflow, from Ref. [53].	27
Figure 12. IAM Workflow, from Ref. [55].	28
Figure 13. Lambda events workflow.	28
Figure 14. Security Hub Core Workflow.	30
Figure 15. IAM cross-account role setup and scanning workflow.	34
Figure 16. Python argparse implementation to parse CLI arguments.	44
Figure 17. Running Python script with CLI arguments.	44
Figure 18. Configuration.ini file.	45
Figure 19. Account directory containing audited AWS services.	45
Figure 20. Regional directories for AWS services.	46
Figure 21. Technical review scan and API response result files.	46
Figure 22. Technical Review account IDs.	47
Figure 23. Technical review HTML display page.	47
Figure 24. Technical review HTML regional collapsible.	48
Figure 25. Technical review HTML scanner result collapsible.	48
Figure 26. Technical review HTML security scanner table.	49
Figure 27. Technical review HTML Lambda service review.	50

Tables

Table 1. Cloud environment definitions.	13
Table 2. Audit tools used for configuration assessment.	20
Table 3. Event types recorded by CloudTrail. From Ref. [42].	23
Table 4. Identity types recorded by CloudTrail. From Ref. [42].	24

List of abbreviations

ACL	Access Control List
AMI	Amazon Machine Image
API	Application Programming Interface
ARN	Amazon Resource Name
ASFF	AWS Security Finding Format
AWS	Amazon Web Services
CLI	Command Line Interface
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
EBS	Elastic Block Storage
EC2	Elastic Cloud Compute
ECR	Elastic Container Registry
EFS	Elastic File Storage
EKS	Elastic Kubernetes Service
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IP	Internet Protocol

IT	Information Technology
JSON	JavaScript Object Notation
KMS	Key Management Service
MFA	Multi-Factor-Authentication
NACL	Network Access Control List
S3	Simple Storage Service
SDK	Software Development Kit
SQL	Structured Query Language
STS	Security Token Service
URL	Uniform Resource Locator
VPC	Amazon Virtual Private Cloud

1 Introduction

The use of cloud computing is increasing at a rapid pace and end-user spending is expected to grow to more than one 1,000,000,000,000 USD during the current decade [1]. This large amount of growth does not come without issues, as cyberattacks targeting the cloud infrastructure are on the increase [2].

AWS (Amazon Web Services) is a public cloud service provider that hosts large data centers in many different countries, see **Table 1** in Section 2.1 for the public cloud definition. This makes it straightforward to set up computational and business operations in various parts of the world. Users do not have to face large upfront costs of buying hardware or setting up data centers which provides cost benefits and makes it easier to focus primarily on providing a business service. AWS offers ready-made services crafted to make operating in the cloud more streamlined. Cloud services can be used for different operations such as computation, data storage, securing and monitoring the cloud infrastructure [3]. In AWS, every service utilizes an identity to perform actions and interact with other services and the improper management of identity permissions and access control for services is a common issue.

Cloud based services can experience attacks such as enumeration of public cloud storage for data, privilege escalation due to misconfigured services and unauthorized access. Security configurations for critical services must be reviewed and verified to ensure that the services are operating correctly. Cloud environment designs often vary and understanding the context is important for a security review to be effective. Operating in the cloud can be costly and optimizing current costs is one of the focus areas for many businesses, as the cost of cloud infrastructure is expected to keep on rising [4].

This thesis will introduce essential AWS services, beneficial security and cost optimization checks and cloud security tools. The objectives of the thesis are to design and build an automated solution for conducting a technical review in AWS, to ensure that the infrastructure is optimized for efficiency and security and is compliant with best practices to protect against cyber threats and data breaches.

2 Technical Review Foundation

The plan for creating an automated technical review was split into four parts:

1. Recognizing relevant AWS services and finding processes, configurations, and best practices for an effective technical review.
2. Identifying the required data needed for analysis, how to collect, process and store the acquired data.
3. Gathering and analysis of the data with self-created or ready-made tooling to ensure cloud environment configurations are properly set up.
4. Creating a clear and easily understandable format for the analyzed data that can be easily shared among individuals.

Technical review is necessary for all cloud environments that handle important data and processes or have access to such environments. Workloads must be configured to emit the information necessary to support them, meaning status of processes and events that occur in the environment should be stored as log files [5]. Establishing processes for utilizing implemented tools and operations within the cloud environment is mandatory, for instance, where to find the location of saved computing logs and the proper procedures for conducting audits on them. It is recommended to create documentation for the technical processes, services and current tools.

Ensuring the proper functioning of the cloud environment through the implementation of various checks and monitoring procedures in case certain services are not running as configured is critical for maintaining the reliability and availability of cloud services. Cloud workload information is beneficial for troubleshooting failures, analyzing security and optimization of services or when the environment must be audited due to regulations. Cloud infrastructure can become expensive which means removing unnecessary storage and computing resources provides improvement to cost benefits. The primary feature of a technical review is ensuring that the cloud environment is configured to operate in a safe and secure way, as indicators for cybercrime complaints and losses are rising on a yearly basis [6]; see **Figure 1**.

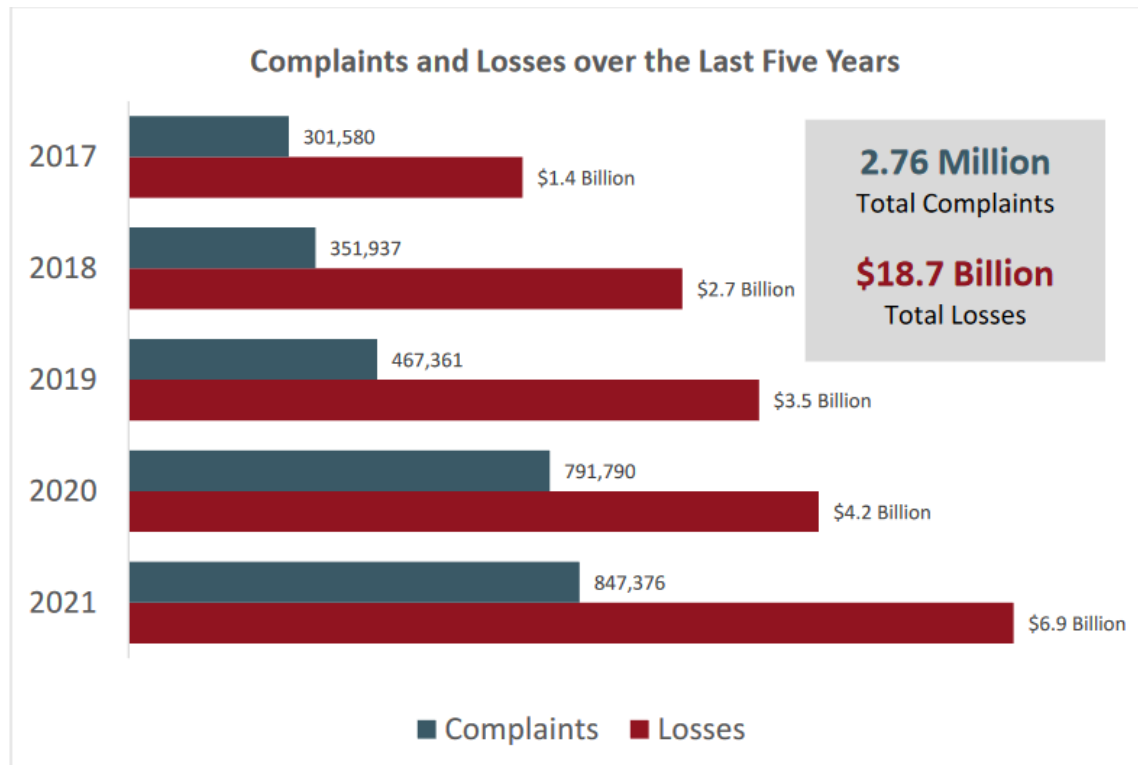


Figure 1. Global Damages caused by cybercrime, from Ref. [6].

Step 1 Recognizing relevant processes

Cloud environments contain multiple moving parts and identifying the processes that are necessary to be reviewed begins with researching which AWS services are essential and what configurations should be enabled or disabled. The documentation provided by AWS offers a comprehensive overview of the numerous services available, as well as detailed information on important security configurations and recommended best practices. More of beneficial information can be found from cloud security research papers, articles and guides written by professionals. These resources describe more in-depth what kind of security checks are ideal for different services and what can be identified from them.

AWS offers useful services that provide out of the box security, monitoring and configuration checks for which users can add more self-made configurations as needed. Many useful security auditing tools have been created for AWS, which allows for assessing important optimization and security configurations, and these will be discussed in **Section 2.3**.

Step 2 Identifying and labeling data

Identifying the right data to gather is important, as it needs to show if proper security controls are currently implemented and how the essential AWS services are configured. The gathered data has to be stored in case auditing is required later for an incident or other data analysis. Data also must be labeled properly so that it is easily identifiable. Data file names generated in the implementation of this technical review contain the AWS account ID, AWS service name, region where the service is hosted, configuration details and the current date. Data files are then structured into a directory path format from which it is easy to identify the data, as shown in **Figure 2** below.

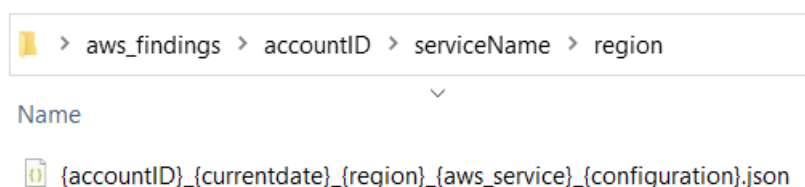


Figure 2. Directory data and file storage format.

Step 3 Gathering and analysis of the data

Data is gathered with AWS API (Application Programming Interface) by requesting data from various AWS services and then storing the response data as a JSON or YAML file. Analysis of the data is then performed with security tools and self-made parsing methods, which can be created using various scripting languages. Every cloud security tool presented in this technical review provides scanning capabilities for JSON or YAML input file formats.

Security tools generate the scan result file outputs, usually in JSON file format. The files containing the API request data are necessary to retain in case the data needs to be audited in the future. The described general workflow is depicted in **Figure 3**.

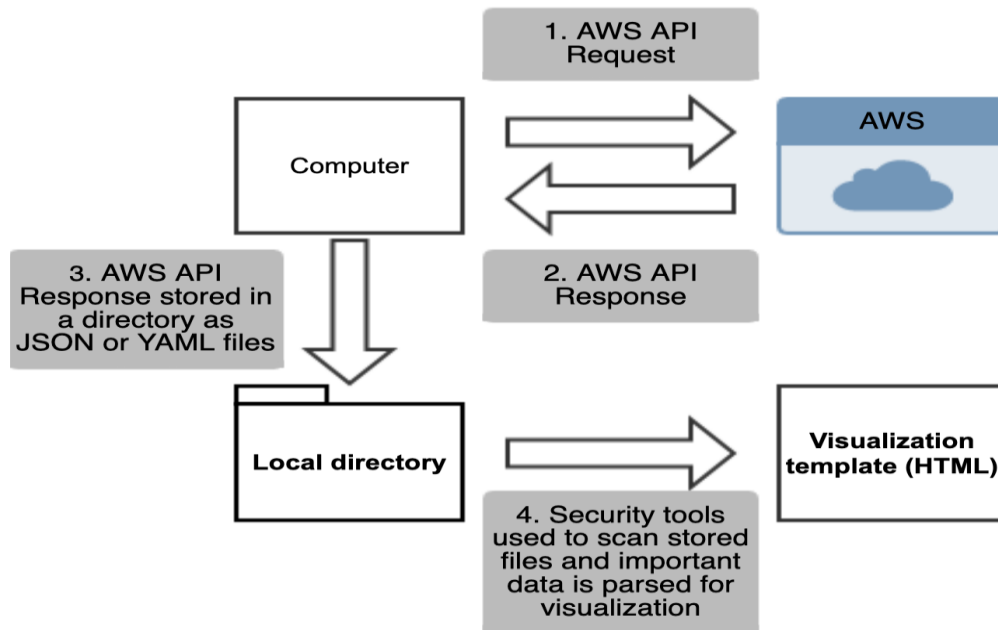


Figure 3. AWS data gathering and analysis workflow.

Step 4 Data presentation

Technical review data should be presented in an easy to comprehend format. The table format, presented in **Figure 4**, was chosen due to its ability to compactly display a large volume of data, facilitating the auditing process.

Severity	Confidence	CWE	Issue	Information	Code
HIGH	HIGH	https://cwe.mitre.org/data/definitions/327.html	Use of weak MD4, MD5, or SHA1 hash for security. Consider <code>usedforsecurity=False</code>	https://bandit.readthedocs.io/en/1.7.4/plugins/b324_hashlib.html	▶ Details

Figure 4. Visualization format for auditing.

Manual auditing is vital for an effective technical review due to varied AWS environment designs. Automatic scanners cannot answer important questions such as internet accessibility and resource necessity. Knowledge of the solution's requirements and design are required by the auditor.

2.1 Cloud Infrastructure

There are usually three different definitions for cloud environments: private, public, and hybrid cloud, presented in **Table 1. Cloud environment definitions.**

Table 1. Cloud environment definitions.

Cloud definition	Description
Private Cloud	Cloud infrastructure is operated only for an organization, meaning no general public access. Organization purchases the computing equipment for their own internal use. Infrastructure and computational resources are managed by either the organization or a third-party provider on premise or off premise. Infrastructure and service access is only available to internal users in the organization, or third parties granted access. Private clouds provide more security and privacy than public clouds but often have higher costs. [7]
Public Cloud	Cloud infrastructure is made available through the internet by an organization that sells cloud services to the general public. Resources are usually offered over an internet connection for a pay-per-usage fee to users. Public cloud vendors, such as Amazon and Microsoft manage the infrastructure, for example the servers and data centers. Users do not have to purchase the required hardware to use the service and can scale their cloud resource usage on demand. [7]
Hybrid Cloud	A hybrid cloud is a combination of at least one private cloud and at least one public cloud. Cloud providers in a hybrid cloud are unique entities but have the same standardized technology that enables data and application portability. In hybrid cloud, an organization provides and manages computing resources in-house and some out-house. For example, an organization hosts confidential customer data in the private cloud and less confidential data in the public cloud. [7]

Linux is the most widely utilized operating system for running services in the cloud [8] and is therefore expected to be subjected to security-related attacks more frequently in the future [9], as predicted by the increase in malware presented in **Figure 5**.

New Linux malware families, 2010-2020

Number of new Linux malware families per year (Source: Intezer)

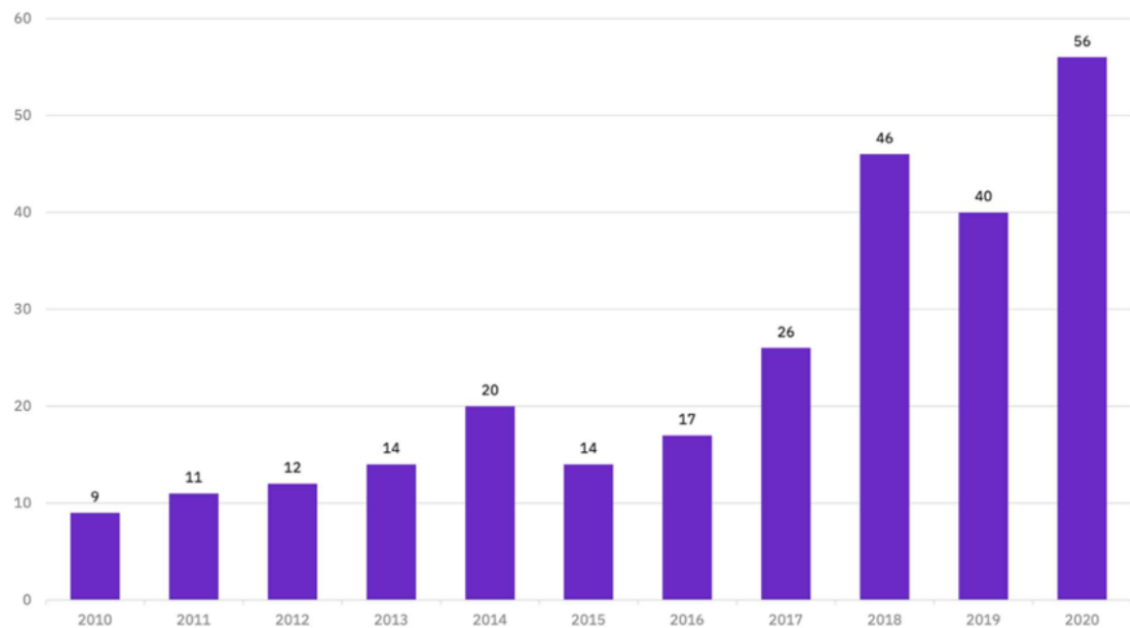


Figure 5. Linux malware yearly growth, from Ref. [9].

Linux is used because it offers a lot of customization, great security and flexibility for optimization. The amount of malware created for Linux is still extremely low, when compared to the current most popular computer operating system Windows.

Malware, such as viruses and ransomware, can infect a cloud system and compromise data. Computer systems should undergo regular security updates, and if security issues are identified, there should be alert systems or automated remediation processes in place. AWS offers services such as Systems manager and Inspector which can be used to automate patching of computing instances and to ensure the best security practices are followed.

AWS Inspector continuously scans EC2 computing instances for vulnerabilities [10] and can be integrated with System manager service which can handle the patching operations based on specified automation runbooks [11]; see **Figure 6**.

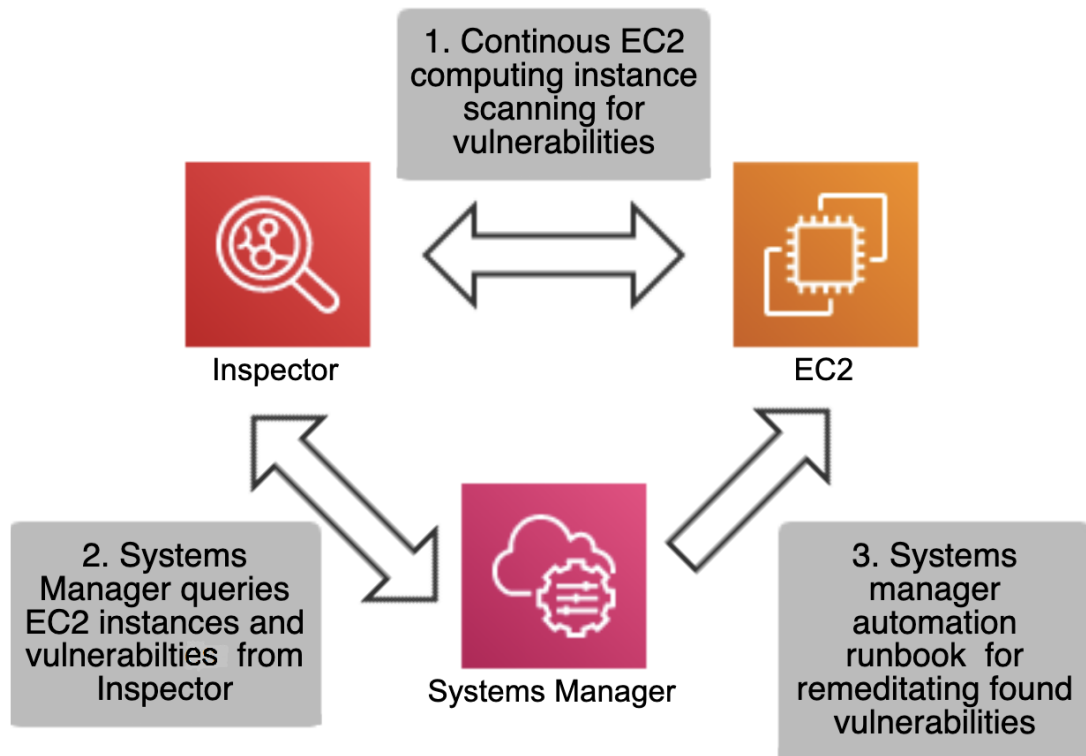


Figure 6. EC2 vulnerability scanning and patching workflow.

Cloud infrastructure security vulnerability findings should be centralized to one specific location where every security incident is reported to. AWS provides a service called Security Hub which can be used to store security findings from multiple sources [12]. The Security Hub is further discussed in **Section 3**.

Monitoring and alerts need to be set in case critical security issues are found related to current active configurations, to enable fast incident remediations. Processes should be created for sending critical and high-severity alerts to operations or support teams who can investigate the issue causing the alert.

Preventing supply chain attacks by adversaries is an important precaution to take. Supply chains attacks are targeted toward suppliers that provide services to different organizations. Compromising the supplier means attacks can be targeted towards the organizations that use the service [13]. This can be achieved by creating multiple layers of security, using zero trust security mindset and MFA (Multi-Factor-Authentication). Zero trust framework works against threats by assigning the least privileged access needed to perform operational tasks [14] and assuming that every connection and endpoint is a threat [15]. MFA requires users to log in with extra authentication, such as a code from mobile phone authenticator application. MFA is necessary to be implemented for extra protection in case user credentials are stolen.

Access to cloud environments should be restricted to only specified account IDs, and IP (Internet Protocol) addresses. Continuous auditing is required for API requests to ensure that the environments don't have any events regarding unauthorized or malicious operations.

Open-source programming libraries used along with cloud operations are required to be assessed for security issues and to ensure that they are created by known trusted parties.

2.2 Microservices

In the past, monolithic approaches were utilized in software, in which processes were closely interdependent and operated as a one big service [16]. This presented a significant risk of failure due to the reliance on a single process. In contrast, microservices address this issue by implementing applications composed of independent components, which reduces the likelihood of system failure, see **Figure 7**.

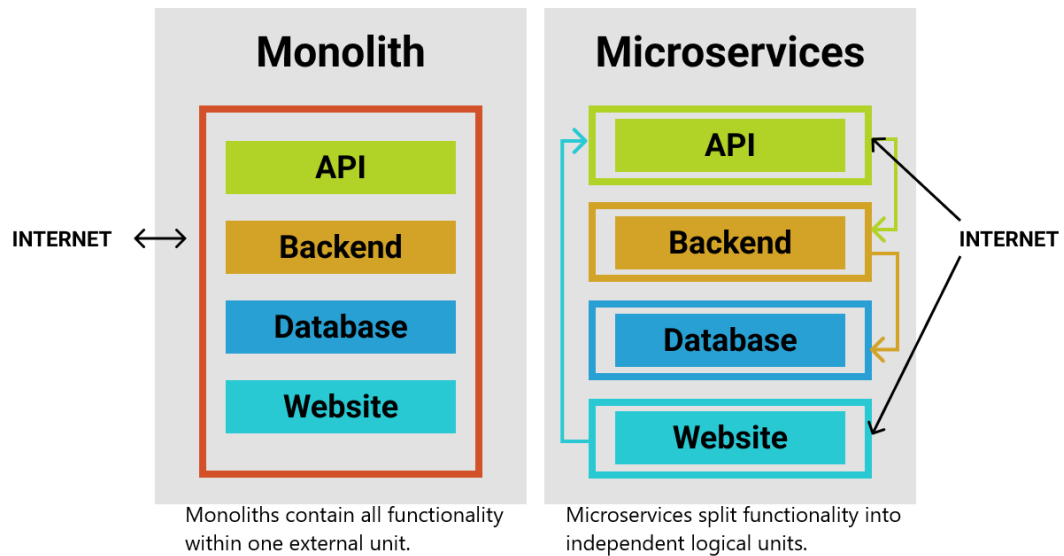


Figure 7. Microservice and monolith architectures, from Ref. [17].

Containers are often used in microservice architecture because they allow for easy isolation, and scaling of applications. Container is a standard way to package application's code, different configurations, and dependencies into a single object, which can run consistently in any computing environment. Containers share the operating system kernel with other containers and applications and is more lightweight than virtual machines, which are used to run a fully virtual operating system. Containerization is useful for developers and operations teams to help in managing and automating development and the deployment of software. Containers enable infrastructure as code workflow, by specifying required infrastructure in a simple configuration file and deploying it as many times as needed. This makes it useful for managing microservice architecture, which consists of a large number of independent components.

Docker is an open platform for building, developing, and running applications in containers and it is widely used in cloud environments [18]. Docker is one of the most popular container platforms and provides all required tooling to enable workflows using containers. Docker also provides detailed documentation on the best practices for running containers in a secure manner [19].

Container orchestration services, such as Kubernetes, are needed when running many containers in a high availability and scalable workflow across a distributed cluster of computing nodes. Kubernetes is a tool for automating deployment, scaling, and management of containerized applications. It is designed to assist developers and IT (Information Technology) operations teams in deploying and managing applications in a consistent and reliable manner, with features such as load balancing, self-healing, and updating [20]. AWS provides documentation on the best practices for running Kubernetes applications in EKS (Elastic Kubernetes Service) [21].

Securing microservice platforms that are used in AWS is important as they are going to be critical targets for misconfigurations and vulnerabilities. Best practices are necessary to follow to ensure the security and efficiency of microservices used in cloud environments.

2.3 Configuration Auditing Tools

Several ready-made security tools that were created by professionals and the open-source community are used to audit AWS services for security issues and misconfigurations. These tools are usually designed to be run from a CLI (Command Line Interface). In this setup of technical review, scanning tools are used through scripts created with the Python programming language and by utilizing the subprocess Python module.

The subprocess Python module allows developers to run CLI commands directly in Python scripts. Tools that are used in the technical review are free to use, and most of them have paid options that include more scanning capabilities. Scanning tools are described in **Table 2**. The focus of the security tooling is on the main cloud threats that have been researched by the Cloud Security Alliance through interviews with various companies operating in the cloud [22]. The threat risk ratings are displayed as survey scores in **Figure 8**. For each issue, The Survey Score was given between 1 to 10, with 10 being the highest possible conceived risk.










Survey Results Rank	Survey Average Score	Issue Name
1	7.729927	 Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts
2	7.592701	 Insecure Interfaces and APIs
3	7.424818	 Misconfiguration and Inadequate Change Control
4	7.408759	 Lack of Cloud Security Architecture and Strategy
5	7.275912	 Insecure Software Development
6	7.214493	 Unsecure Third Party Resources
7	7.143066	 System Vulnerabilities
8	7.114659	 Accidental Cloud Data Disclosure/ Disclosure
9	7.097810	 Misconfiguration & Exploitation of Serverless & Container Workloads
10	7.088534	 Organized Crime/ Hackers/ APT
11	7.085631	 Cloud Storage Data Exfiltration

Figure 8. Top cloud threats to audit with security tools, from Ref. [22].

Table 2. Audit tools used for configuration assessment.

Tool	Description
Bandit	Designed to find common security issues in Python code. Utilized to inspect Python code for passwords, authentication keys and code risks [23]. Applied to scan AWS Lambda functions that contain Python code.
Boto3	SDK (Software Development Kit) for AWS. Enables sending API requests to AWS programmatically using Python as the programming language [24] (see A1: Boto3 example). Boto3 is operated to retrieve configuration data from AWS for auditing and security scanning purposes.
Cfn-lint	Linter for CloudFormation templates (see A1: CloudFormation JSON template). Validates CloudFormation templates against the AWS CloudFormation Resource Specification, and ensures values given for resources are following best practices [25].
Checkov	Security scanner for CloudFormation templates (see A1: CloudFormation JSON Template). Provides scan results for misconfigurations and information on how security remediations can be applied [26]. Checkov is utilized to audit security issues in CloudFormation templates.
CloudSplaining	IAM (Identity and Access Management) security assessment tool that identifies violations of least privilege and generates a risk-prioritized report [27]. Applied to identify security problems within IAM configurations.
Parliament	Analyzer for IAM Policies (see A1: IAM Policy). Identifies problems related to bad policy patterns and incorrect configurations in IAM Policies [28].
Prowler	Security tool that can perform many security checks for assessing and auditing AWS environments and resources. Features more than 240 best practice security checks and is highly configurable [29]. Prowler is employed to check various security configurations in AWS services.
Pylint	Linter for Python code. Pylint is utilized to display errors and warnings found in Python scripts that are retrieved from AWS Lambda [30].
Trivy	Vulnerability scanner for various DevOps environments. Operated for scanning Docker images that are hosted in AWS and to provide a vulnerability report for each scanned image [31].

3. Amazon Cloud Computing Services

AWS is currently the world's largest cloud service provider [32] and provides renting of computing infrastructure, such as virtual computing servers, data storage and several managed services to help accomplish various business requirements. Cloud Infrastructure from AWS can be rented from various regions located in the world [33]. Regions specify where the data center hosting the service is in the world. For example, the AWS region named eu-north-1 is in Stockholm. Services include databases, computing capacity, resource monitoring and much more [34]. In addition, serverless services can be be rented, allowing users to run their own code and applications on infrastructure that is fully setup and managed by AWS [35].

VPC (Amazon Virtual Private Cloud) service is utilized to host a virtual network, which resembles a traditional network within AWS cloud environments. This allows users to launch and manage AWS resources, such as computing instances and load balancers, in a secure and isolated manner within a scalable network infrastructure [36]. By leveraging the features of VPC, users can ensure that their resources are isolated from other AWS customers and protected from external threats. Additionally, VPC allows users to configure and manage their network infrastructure to meet their specific needs.

AWS provides an API which allows programmatically to setup, modify, or get information about the current cloud infrastructure, making it possible to create automation workflows using code.

Physical security of the cloud environment, such as the data centers are handled by AWS, but the security of user run applications and cloud environments are the customers responsibility [37]. AWS provides documentation for the well-architected six framework pillars of operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability. Well-architected framework pillars present the requirements for operating in the cloud and help in designing stable, secure and efficient cloud operations [38].

3.1 Essential Services

This section describes AWS services that were identified to be relevant for auditing configuration data or requiring a technical review to ensure that the proper settings are enabled or disabled.

3.1.1 Athena

Athena is an interactive query service that makes it easy to analyze log data in S3 (Simple Storage Service) buckets using standard SQL (Structured Query Language) [39]. Athena is serverless, so there is no infrastructure to manage, and you pay only for the SQL queries that are run. Athena allows users to investigate logs from S3 buckets for services such as CloudTrail by creating an SQL table. This table can then be queried with SQL statements, based on what log data needs to be audited (see **Appendix 1: CloudTrail Athena SQL Table and SQL query for CloudTrail Athena table**).

Athena is essential for log analysis as it does not require long times for downloading log data from AWS to the local machine and it scales well for vast amounts of data.

3.1.2 CloudFormation

CloudFormation provides Infrastructure-as-code templates for automating AWS resource creation and configuration [40]. CloudFormation templates can be created using JSON or YAML data formats (see **Appendix 1: CloudFormation JSON template**). These templates specify the AWS resources to be created and their configurations and are reusable for setting up the infrastructure which makes future cloud deployments more automated and streamlined.

3.1.3 CloudTrail

CloudTrail is the primary auditing service for AWS which monitors and records account activity across AWS infrastructure and stores the data as log files [41]. CloudTrail logs contain invaluable information which makes it possible to see what is happening across the whole AWS infrastructure. Log files are written in JSON format, with each event presented as a single JSON object (see **Appendix 1: CloudTrail event**). CloudTrail events can be written to objects in an S3 bucket, and they typically appear within 15 minutes of the API request. The logs are written as gzip-compressed JSON files, with all the events for the whole 15-minute period in one file. Every CloudTrail event has an event type and identity type. Event type describes what kind of API request was performed (see **Table 3: Event types recorded by CloudTrail**). Identity type describes which identity made the API call (see **Table 4: Identity types recorded by CloudTrail**).

Table 3. Event types recorded by CloudTrail. From Ref. [42].

Event type	Description
Management event	Entries for management and network (control plane) operations performed on the resources in your AWS account, such as security group configuration changes, IAM role permission adjustments, and VPC network alterations.
Data events	Entries for data request operations—such as Get, Delete, and Put API commands—performed on an AWS data plane resource.
Insight events	Entries that reflect unusual API activity in your AWS account in comparison to your historical API usage, such as excessive API calls in a short frame of time.

Table 4. Identity types recorded by CloudTrail. From Ref. [42].

Identity type	Description
Root	The request was made with your primary AWS account credentials. If you set up an alias for your AWS account, that alias will appear here instead.
IAMUser	The request was made with the credentials of an IAM user.
FederatedUser	The request was made by a user with temporary security credentials provided through a federation token.
AWSAccount	The request was made by a third-party AWS account.
AWSService	The request was made by an AWS service account. Many AWS services use service accounts to perform automated actions on your behalf.
AssumedRole	The request was made with temporary credentials obtained by using the AWS STS (Security Token Service) AssumeRole operation.

Config

Config enables assessment, auditing, and evaluating the configurations of AWS resources [43]. If a resource violates a rule, Config flags the resource and the rule as noncompliant, this action also generates a finding into Security Hub, as depicted in **Figure 9**. Config rules can be created using AWS Lambda functions, and AWS provides premade rules that can be easily enabled for different resource configuration checks. Config can apply auto-remediation actions using AWS Systems Manager Automation documents, which applies a remediation action on a non-compliant AWS resource to make it compliant again.

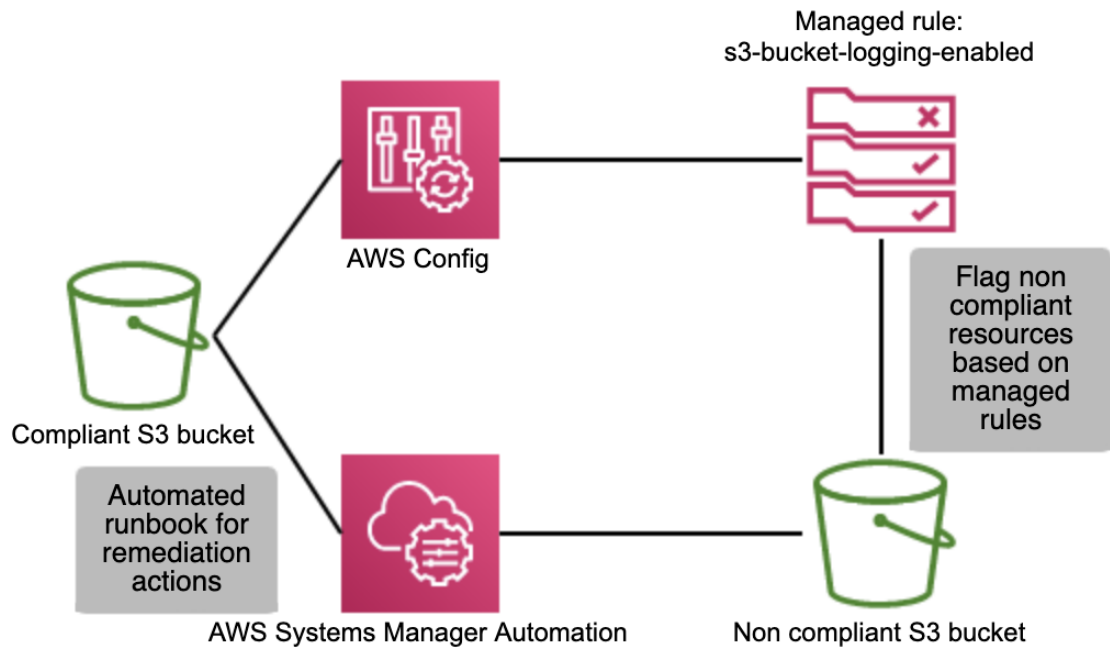


Figure 9. Config flagging resource and auto-remediation workflow.

Cost Explorer

Cost Explorer displays AWS service costs and usage over time, allowing to assess how the environment is optimized by price [44]. Cost data is updated daily, and users can select the desired time period for auditing purposes. Cost Explorer enables users to concentrate on the costliest services to reduce costs and audit for mismanagement or malicious activity if costs suddenly increase.

Elastic Cloud Compute

EC2 (Elastic Compute Cloud) provides virtualized cloud computing capacity and contains many functionalities to setup server configurations [45]. AMI (Amazon Machine Image) serves as the basis to configure the operating system and software when launching EC2 computing instances [46]. AMIs function as templates that contain the necessary specifications for the desired server. Users are able to choose from various EC2 hardware configurations based on their server requirements. The available hardware to choose from includes the CPU

(Central processing unit), GPU (Graphics processing unit), memory, data storage and network capacity configurations [47].

EC2 offers two types of persistent storage volumes in the form of EBS (Elastic Block Storage) and EFS (Elastic File Storage). EBS provides block level storage volumes that behaves like raw, unformatted block devices [48]. Block devices are hard disks where you read and write one block of data at a time. EFS provides serverless network file system which enables sharing of file data without provisioning or managing storage capacity and performance [49]. EC2 instances that lack a persistent storage volume will lose all of the stored data in the virtual machine when it is shutdown. EC2 instances can be protected with a security group that functions as a firewall and enables specification of allowed protocols, ports and source IP ranges that can reach the computing instances [50].

Elastic Container Registry

ECR (Elastic Container Registry) is a fully managed Docker container registry used for storing container images used in microservices [51]. Container images include packaged configurations that are used to build applications. These images can be retrieved and stored in repositories within ECR to streamline the workflow with containers used in microservices, depicted in **Figure 10**.

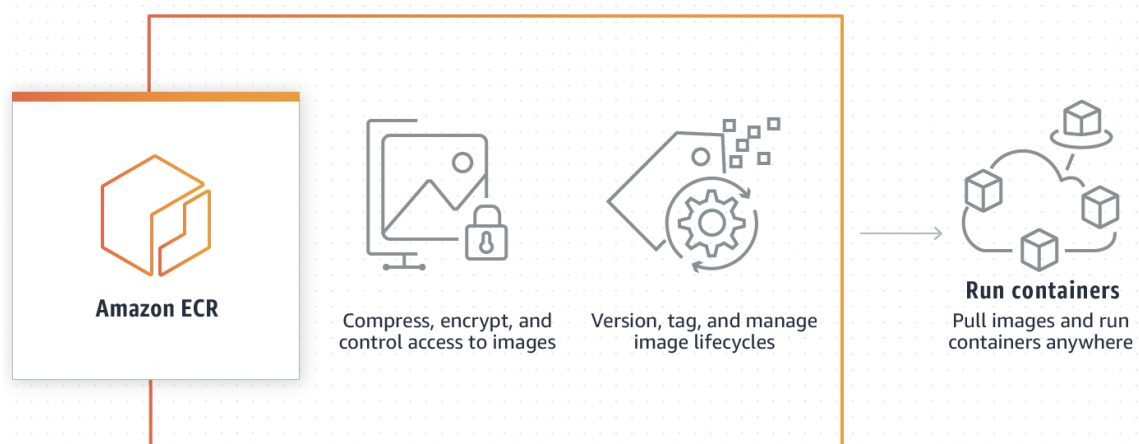


Figure 10. ECR Repository storage for application images, from Ref. [52].

Elastic Kubernetes Service

EKS is a fully managed service to run and scale Kubernetes applications in the cloud [53] and takes care of all the computing cluster setup and creation. EKS runs up-to-date versions of the open-source Kubernetes software and takes away the operational burden involved in running the Kubernetes control plane, as depicted in **Figure 11**. The downsides to managed services such as EKS is less control and customization options. EKS provides high availability and scalable workflows for cloud applications and is also integrated with many AWS services to provide scalability and security.

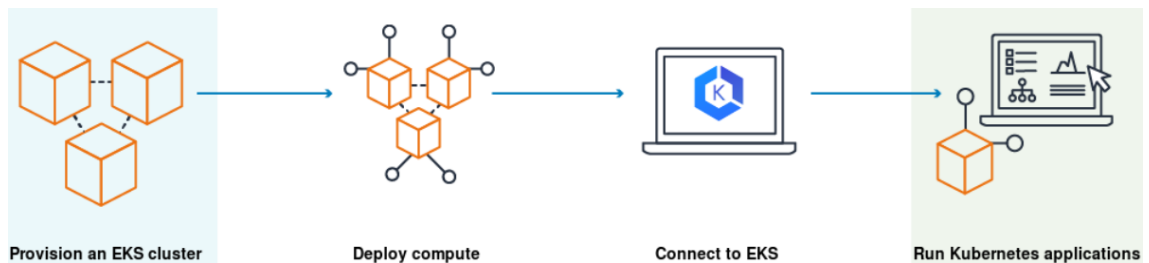


Figure 11. EKS Workflow, from Ref. [53].

Identity and Access Management

IAM specifies who can access which AWS services and resources, and under which conditions [54]. IAM contains users, groups, roles, and policies. User is an entity in AWS to represent the person that uses it to interact with AWS. Each IAM user has a unique name and a set of security credentials, which can be used to programmatically access AWS services.

Role is an IAM Identity that can be created in an AWS account that has specific IAM policies assigned to it. Roles can be used to delegate access to users, applications or services that do not normally have access to AWS Resources. IAM Role sessions have limited duration, which reduces the risk of credentials being compromised.

IAM group is a collection of IAM users, groups have permission attached to them and users in a group will receive all of the group permissions.

IAM policy describes API permissions that can be performed when that policy is used (see **A1: IAM Policy**). It is necessary to determine what users or roles need to do and then craft policies that allow them to perform only those tasks. IAM use case and workflow is described in **Figure 12**.

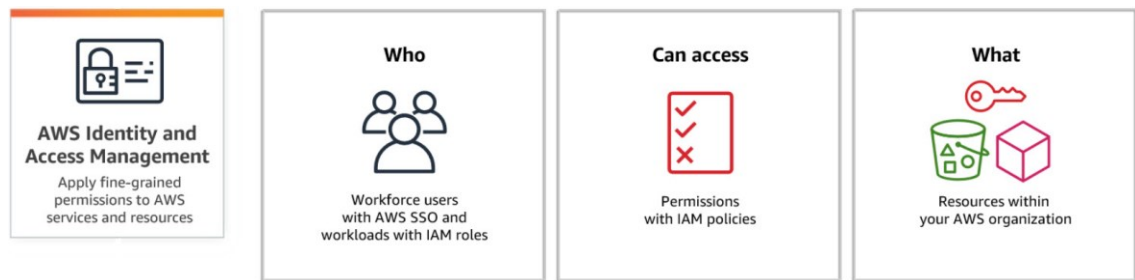


Figure 12. IAM Workflow, from Ref. [55].

Lambda

Lambda is a serverless event-driven computing service [56] that enables the execution of code for applications and backend services without the need to provision or manage self-hosted servers. It stores programmable scripts to run functions regarding AWS resources, which can be triggered upon the occurrence of various events. Lambda scripts can be implemented in several programming languages such as Python, and Go. **Figure 13** depicts the Lambda workflow.

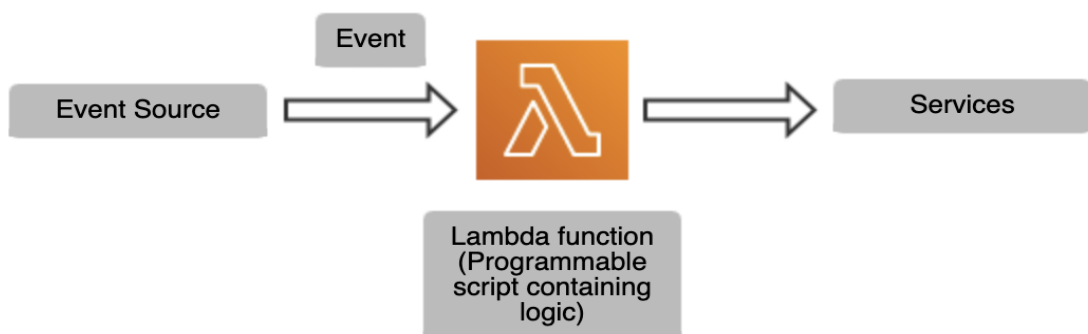


Figure 13. Lambda events workflow.

Simple Storage Service

S3 is an object storage service that is used for data storage in AWS [57]. S3 stores data as objects, which consists of object data and metadata into a bucket container. An object is uniquely identified within a bucket by a key name. Users can specify access controls to the bucket with the use of IAM and bucket policies.

Data in S3 buckets can be private meaning access through the public internet is restricted to the bucket data. Public access to an S3 bucket enables the retrieval of data stored in the bucket over the public internet. S3 public access is required when hosting services to users that need to access non-sensitive data, such as website content. Private buckets are used to securely store and manage confidential data, such as business-critical files or databases.

S3 provides various storage methods, which can be chosen based on how long the data should be stored and how quickly it needs to be accessed [58]. Storing the data in a colder storage, where it takes a longer time to retrieve the data is less expensive and can provide great cost benefits.

Security Hub

Security Hub provides a comprehensive and centralized view of the security state of the AWS environment and can be used to audit cloud resources for security industry standards and best practices [59]. Security Hub requires that Config is enabled in all accounts that have Security Hub enabled, this is because Config is used to send findings to Security Hub.

Security Hub can be integrated with various security tools and AWS services that provide scanning of security configurations, such as Inspector, which is utilized for vulnerability assessment of EC2 instances. Integration with Security Hub allows the service to send assessment data to Security Hub in ASFF (AWS Security Finding Format) format (see **A1: ASFF finding format**). Security Hub can then be used to display all of the security findings on a single page, simplifying the process to identify which security issues need to be addressed. The Security Hub core workflow is summarized in **Figure 14**.

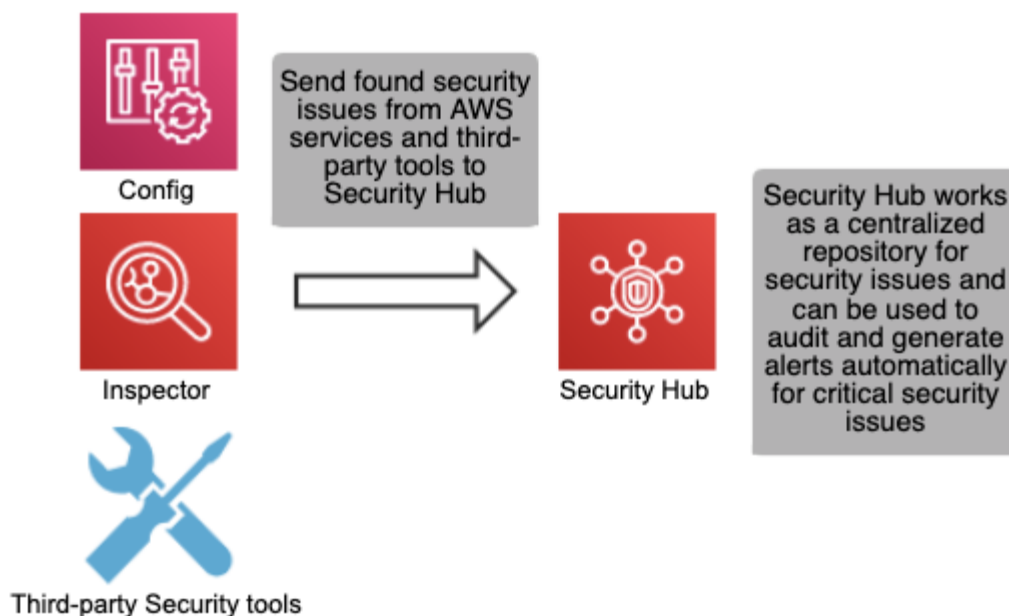


Figure 14. Security Hub Core Workflow.

3.2 Data Protection

Data security and management in cloud environments refers to measures taken to protect data and systems in a cloud environment from unauthorized access, breaches, and other types of attacks. Data protection and privacy regulations must be strictly adhered to avoid significant financial penalties for non-compliance.

Data breaches resulting from mismanagement can have large consequences, including monetary losses, loss of users, and legal issues that can lead to bankruptcy. To safeguard critical data, encryption is necessary, and AWS offers services such as the KMS (Key Management Service) for this purpose [60]. Encryption is the process of converting data into a secure, encoded format to protect it from unauthorized access. In a cloud environment, data can be encrypted when it is stored and when it is being transmitted.

Data retention must be properly configured to allow for the audit of older data as needed. Confidential data must be kept secure and not made publicly accessible. Different countries have varying regulations regarding the storage and processing of data, so it is essential to ensure that data is stored and handled in accordance within these regulations.

Data collected from the technical review of AWS environments requires exceptional care as it often contains sensitive information that could be exploited if it falls into the wrong hands. Overall, data security and management in cloud environments is critical to protect the data and systems from unauthorized access and breaches.

4 Technical Review Assessment

This section gives an overview of how the technical review process is done and what configurations are assessed for each AWS service presented in the previous section. The first subsection goes through requirements, the second subsection provides details of how the assessment is done and the third subsection displays how the manual review process is achieved.

4.1 Assessment Requirements

Technical review has requirements for the AWS IAM role that need to be fulfilled before the assessment can be done. The IAM role is required to have the necessary IAM read only policy permissions for multitude of AWS API requests to assess the technical configurations of AWS environments.

Assessing multiple AWS accounts requires creating a cross-account trust policy for the IAM Role that is used in the review process. Trust relationship means IAM entities can assume an IAM role under specified conditions, such as a specific IAM role from a different AWS account that is used for the technical review.

To make cross-account role assuming more secure, it is mandatory to use an external ID, and requiring that the assuming entity has MFA enabled. External ID is a string value that is required to be present in the API request when a third-party is trying to assume an IAM role and provides extra security.

The process to set up cross-account roles between account 111111111111 and 222222222222 is described below. Role A and Role B are in different AWS accounts. Role B needs to have a trust relationship to allow Role A to assume it.

Role B in account 222222222222 requires an IAM trusted relationship policy to allow sts:AssumeRole access from Role A in account 111111111111.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/RoleA"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "CustomExternalID"
        }
      }
    }
  ]
}
```

Role A in account 111111111111 requires an IAM trust policy sts:AssumeRole to assume Role B in account 222222222222.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::222222222222:role/RoleB"
  }
}
```

After trusted relationship between roles has been configured, the assume role operation can be done from the 111111111111:RoleA to 222222222222:RoleB. **AssumeRole** API request is used to retrieve temporary credentials for the role located in the other AWS account. Retrieved credentials can then be used to access the account resources, based on what IAM permissions the role has. The technical review IAM cross-account scanning process for multiple accounts is visualized in **Figure 15**.

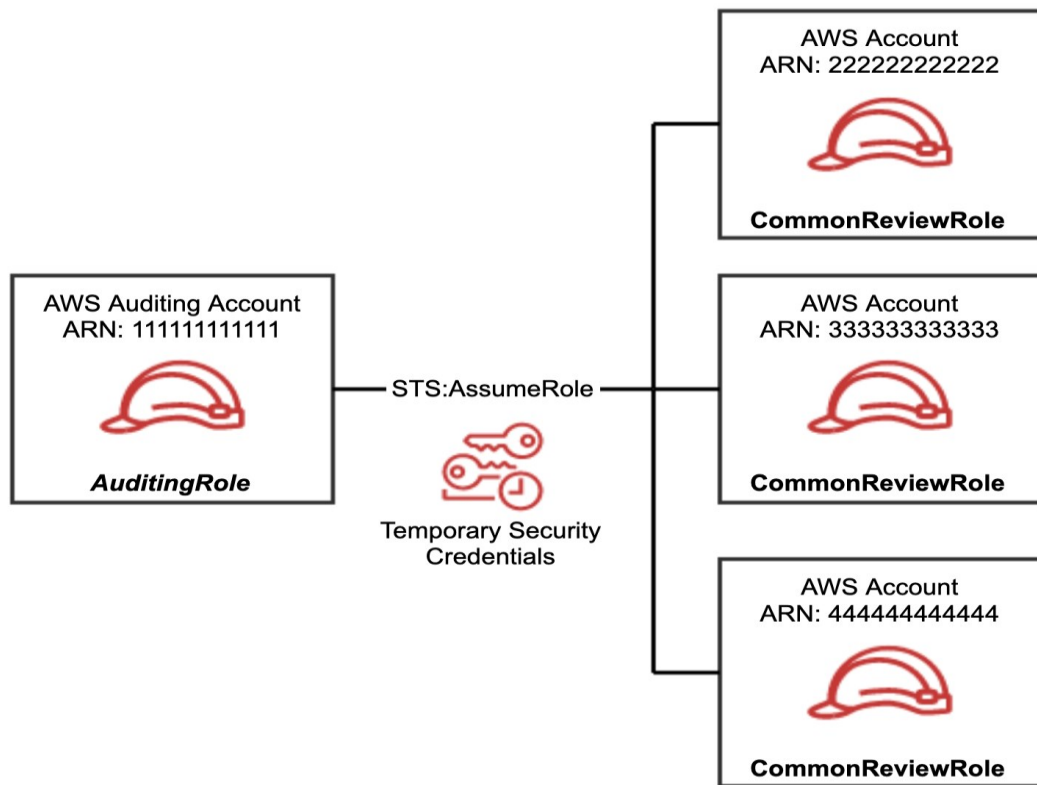


Figure 15. IAM cross-account role setup and scanning workflow.

4.2 Service Assessments

A programmable script was implemented with the Python programming language to run the required technical review steps automatically. In this section a numbered list is used to display technical review assessment steps for the AWS services. After the numbered list there will be more information about how and why these steps are done. Boto3 is used to send AWS API requests and is represented in the text with the function name, for example **ListStacks**. Prowler security tool checks are in lowercase format, such as **ec2_ami_public**. Data from the scan reports and requests are saved as JSON or YAML files for data parsing and further analysis. Data retrieved from the assessments will be shown to the reviewer in an HTML (Hypertext Markup Language) template and is described in **Section 4.3**. Every identified vulnerability and configuration issue should undergo a manual auditing process by a reviewer.

CloudFormation assessment

1. Download templates from AWS
2. Validate that the templates are configured properly
 - a. Scan and analyze templates
3. Ensure that no secrets are found in stack outputs
4. Termination protection is enabled for the required stacks

The process of analyzing CloudFormation templates involves downloading them from AWS. This is achieved through the use of **ListStacks**, which retrieves every CloudFormation stack, and **GetTemplate**, which retrieves each CloudFormation template. The templates are parsed from the response, then Checkov and Cfn-lint tools are run on each template. Checkov is used to scan for vulnerabilities and misconfigurations, while Cfn-lint verifies the use of valid resource values and best practices.

CloudFormation output values are return values that are optional and not obscured. They are available in clear text following stack operations. The **cloudformation_outputs_find_secrets** is used to audit that no secrets are present in the outputs. CloudFormation stacks play a vital role in cloud infrastructure, and deleting a stack removes all of the resources created by the stack. The **cloudformation_stacks_termination_protection_enabled** can be utilized to verify that termination protection is enabled, to prevent the accidental deletion of the stacks.

CloudTrail assessment

1. Trail is enabled for account
2. Audit configurations
 - a. Logging is enabled for every AWS region
 - b. Log data is not available to public
 - c. Log files are created with a digest file and encrypted at rest
3. Review account authentication and unauthorized activity API events

Enabling the delivery of CloudTrail events to an S3 bucket requires the activation of CloudTrail Trail on the account. The currently enabled trails for the AWS account can be inspected using the **DescribeTrails** function. To prevent attackers from using a region with no API request monitoring for malicious purposes, CloudTrail logging must be enabled for every region. Validation that logging is enabled for every region can be achieved using the check called **cloudtrail_multi_region_enabled**.

It is essential that the S3 bucket used for storing CloudTrail logs is not publicly accessible. Validating that the CloudTrail logs are not public can be accomplished with the use of **cloudtrail_logs_s3_bucket_is_not_publicly_accessible**. Log files should be created with a signed digest file, to ensure that the logs have not been modified. This is evaluated with **cloudtrail_log_file_validation_enabled**. In addition, log files should be configured to use server-side encryption at rest, which can be verified using **kms_encryption_enabled**.

Athena is used to evaluate CloudTrail logs stored in a S3 bucket. The API requests ConsoleLogin and AssumeRole are utilized for authenticating to AWS accounts. These requests are audited for legitimate IP addresses and to ensure that zero trust strategy is in use. Additionally, unauthorized API request responses are reviewed, as the activity could be malicious, such as attempting to delete an S3 bucket that contains essential information.

Config assessment

1. Configuration recorder is enabled in all AWS regions that have active resources
2. Retrieve the current status of Config rules

The Configuration Recorder detects changes in AWS resource configurations and records them. The operational status of the recorder can be assessed by using the **DescribeConfigurationRecorderStatus** request. It is necessary to validate the Config rules for the environment to ensure that they are active and can be located from the rules.

This task can be accomplished by using the **DescribeConfigRules** request to retrieve the status of the current Config rules, which can then be reviewed manually. For instance, the Config rule that is used to audit status of S3 buckets public access, **S3_BUCKET_LEVEL_PUBLIC_ACCESS_PROHIBITED** should be confirmed to be in active state to ensure that Config automatically assesses this configuration.

Cost Explorer assessment

1. Fetch costs of AWS services in a monthly timeframe.
2. Audit the costs of AWS services.

GetCostAndUsage request is used to fetch the cost of services and can be given a time period as parameter, such as how many months to check the costs for. An analysis of the costs associated with the use of AWS services over a specified period can provide valuable insight into the cost efficiency of the environment. This information can also be used to identify if malicious activity is ongoing in case the costs have suddenly risen, for example, due to malicious EC2 computing instances used for crypto mining operations.

Elastic Cloud Compute assessment

1. Security configurations
 - a. Security groups
 - b. Network access control list
 - c. EC2 instances have IAM roles assigned for accessing AWS resources
 - d. Secrets reviewed
2. Instance configurations
 - a. AMI Images
 - b. EBS and snapshots
3. Cost optimization

The security groups function as virtual firewalls to control incoming and outbound traffic. Auditing security group settings is done with the use of two different checks. First **ec2_securitygroup_allow_ingress_from_internet_to_any_port** is used to get every open computer port from the EC2 instances. Second check **ec2_securitygroup_default_restrict_traffic** is used to verify that the default security group of every VPC restricts all traffic. Firewall configurations are then audited manually based on how restricted the EC2 instance traffic should be.

NACL (The Network Access Control List) serves as a firewall to control inbound and outbound traffic within VPC subnets. The process of auditing ingress ports in EC2 subnets is done with the use of **ec2_networkacl_allow_ingress_any_port** and a manual review to determine if any unnecessary ports remain open. EC2 instances should not be directly exposed to the internet by having a public IP as this increases the attack surface. Ensuring that the EC2 instances do not have a public IP is verified with **ec2_instance_public_ip**.

EC2 instances should use IAM roles to access AWS resources instead of encoding AWS keys into API calls, to reduce risk of credentials exposure. EC2 IAM roles usage is assessed with **ec2_instance_profile_attached**. Hardcoded secrets should not be used in EC2 user data as they can be used to gain lateral access to other services and is audited with **ec2_instance_secrets_user_data**.

AMIs contain data about the computing environment that is used to host processes and should not be available to the public internet. AMIs can contain vulnerable information and allow threat actors to find exploits related to the system configurations. This is examined with **ec2_ami_public**.

EBS volumes and snapshots should be encrypted and not available to the public to protect data from unauthorized access. Encryption is validated with two checks **ec2_ebs_snapshots_encrypted** and **ec2_ebs_volume_encryption**. Public access to snapshots is audited with **ec2_ebs_public_snapshot**.

To optimize costs, various methods can be employed such as identifying unused EC2 instances, monitoring instances to check if they are utilizing all of the hardware resources, and then downgrading to lower cost instances. Additionally,

the removal of unused EBS snapshots can provide cost savings, as data storage in AWS incurs expenses.

The process of auditing EC2 instances involves utilizing the **DescribeInstances** request to obtain a list of the currently active instances, which are then manually assessed to determine if the instances are required to be in a running state. Monitoring should be enabled as it provides insight into hardware usage metrics and enables reviewers to determine if hardware can be downgraded for cost optimization.

To delete EBS snapshots that are no longer required, a **DescribeSnapshots** request is sent, and then each retrieved snapshot will be reviewed to determine if those are not attached to any AMIs, EKS clusters or managed by AWS services. Snapshots that are no longer deemed necessary are then deleted to achieve cost benefits.

Elastic Container Registry assessment

1. Get every container image from ECR
2. Scan for vulnerabilities in container images

In order to perform the scanning, it is necessary to retrieve every application image from the ECR repository. The **DescribeRepositories** request is used to obtain information on every repository that exists within the ECR. Following this, the **ListImages** request can be utilized to acquire the URL (Uniform Resource Locator) of every image that is present in the repositories.

Trivy tool is utilized for the purpose of image vulnerability scanning. This tool can operate with a specific image URL and is applied to execute scans for each image URL that is retrieved. Vulnerabilities that are found with the tool need to be audited and fixed in container or application development.

Elastic Kubernetes Service assessment

1. Retrieve AWS EKS clusters
2. Scan Kubernetes workloads and configurations

The process of retrieving and analyzing EKS configurations is done by utilizing the **ListClusters** request. For each cluster that is retrieved, the **DescribeCluster** request is used to obtain information regarding the cluster. It is critical that EKS clusters run on a Kubernetes version that continues to receive support and security updates.

Security configurations within Kubernetes workloads that operate in EKS must be analyzed for vulnerabilities. Assessment is required for every component that runs within a Kubernetes cluster, including container images and cluster configurations. The examination of Kubernetes workloads involves authenticating into the cluster and downloading every image and configuration file from it. Retrieved images are then assessed for vulnerabilities with the use of Trivy, while configuration files are evaluated using Checkov.

Identity and Access Management assessment

1. Retrieve account authorization details
2. Assess IAM entities, security and policies
3. Ensure root account security
 - a. MFA is enabled
 - b. Not used for normal cloud operations
 - c. API key is not enabled for the root user
4. Audit user configurations
 - a. Strong passwords enabled
 - b. Access keys reviewed
 - c. Unnecessary credentials removed

GetAccountAuthorizationDetails request is utilized to retrieve information about all IAM users, groups, roles, and policies in the AWS account. The CloudSplaining IAM security scanner is run against the IAM account authorization details file, which generates an HTML report for conducting an IAM security assessment. The results obtained from CloudSplaining are used to ensure that roles, users, groups, and policies are operating under a zero trust strategy. IAM policy assessment is done by running the Parliament policy analyzer on IAM policies retrieved from the IAM Authorization details file to ensure that policies do not contain security errors.

Hardware MFA should be used to enhance the security of the root account and prevent breaches. This is audited with **iam_root_hardware_mfa_enabled**. API keys should not be used for root users, as compromising these keys give full unrestricted access to the whole environment. This can be audited using **iam_avoid_root_usage**. Additionally, root users should not be used for interacting with AWS services, as this undermines the efforts towards using IAM and its access control capabilities. Assessed with **iam_no_root_access_key** method.

To provide an additional layer of security in case of IAM user credential theft, it is necessary to enable Multi-Factor Authentication for every user. This can be verified with the use of **iam_user_mfa_enabled_console_access**. It is also recommended to have a strong password policy to minimize the risk of account compromise. Password policy can be reviewed to ensure user passwords are at least 14 characters with **iam_password_policy_minimum_length_14**.

IAM access keys should be regularly rotated to limit the duration of the credentials used to access your resources, in case they are compromised without your knowledge. This is examined with check **iam_rotate_access_key_90_days**. In addition, it is recommended to remove credentials that are no longer required. This can be assessed using **iam_disable_90_days_credentials**.

Lambda assessment

1. Download Python Lambda scripts
2. Scan and analyze Python functions for security errors and warnings

To conduct an analysis of Lambda scripts, it is necessary to download them from AWS. This can be accomplished by utilizing the **ListFunctions** request, which returns the ARN (Amazon Resource Name) of each Lambda function. For each function ARN, the **GetFunction** request is utilized to download the function code.

The Lambda Python code files are then scanned using the Bandit scanner to detect any critical or high security issues that may be present in the Python code. Furthermore, the code files are evaluated using Pylint to identify any errors or warnings that could potentially affect the functionality or security in an unfavorable manner.

S3 assessment

1. Audit Simple Storage Service configurations
 - a. Public and user access disabled
 - b. Bucket and IAM policies used instead of access control lists
 - c. HTTPs requests enabled
 - d. Encryption enabled

To ensure the confidentiality of data, it is necessary to disable public and AWS user access to private buckets. Bucket public access is assessed with check **s3_bucket_public_access**. To prevent non-intended users from gaining write access, access control policies must be implemented. This is audited using **s3_bucket_policy_public_write_access**.

For buckets containing critical data, MFA delete should be implemented. MFA delete requires the use of an additional MFA key for deleting objects in a bucket, and this can be checked using the **s3_bucket_no_mfa_delete** method. S3 ACL

(Access Control List) should be disabled and replaced with IAM and bucket policies, which can be assessed with the use of **s3_bucket_acl_prohibited**.

HTTP (Hypertext Transfer Protocol) is unencrypted and can therefore expose sensitive information in clear text over a network. As a result, HTTPS (Hypertext Transfer Protocol Secure) should be used instead of HTTP when transferring data over a network between S3 buckets, and this can be verified with check **s3_bucket_secure_transport_policy**. Additionally, default encryption should be enabled at rest for all buckets to protect data, which can be inspected with **s3_bucket_default_encryption**.

Security Hub assessment

1. Ensure Security Hub is enabled
2. Retrieve and review configurations
 - a. Security findings
 - b. Enabled standards
 - c. Enabled products

Security Hub needs to be enabled on the AWS account to obtain necessary security findings. This can be assessed with the **DescribeHub** request and verifying that the Security Hub ARN is present in the response. To retrieve security findings, the **GetFindings** request is used. Any critical security findings that are identified should be immediately audited and resolved.

Enabled standards refer to the security standards that are currently activated in the AWS account. These standards can be obtained by utilizing the **GetEnabledStandards** request. An example of such a standard is the AWS Foundational Security Best Practices by AWS Security, which enables automated security checks that detect whether the AWS account or its deployed resources are aligned with security best practices.

Products describe the AWS services that send security results to Security Hub and can be obtained through the **ListEnabledProductsForImport** request. Inspector is an example of an AWS service that is capable of sending results to Security Hub if any security issues are detected.

4.3 Review process

Technical review process is implemented as a Python script, where it is possible to specify which AWS services to scan with CLI arguments. Python was selected as the scripting language because it is widely supported, contains many useful programming libraries and has a SDK for interacting with AWS services. Python also offers an abstracted multiprocessing library that allows the review to progress faster by running the security scan processes in parallel. CLI argument parsing is implemented with a Python library called argparse, see; **Figure 16**.

```
parser.add_argument('-l', '--lambdafunctions',
                    action='store_true',
                    help='Creates table of Lambda pylint & bandit scan results')

parser.add_argument('-cf', '--cloudformation',
                    action='store_true',
                    help='Creates table of Cloudformation cfn-lint & checkov scan results')
```

Figure 16. Python argparse implementation to parse CLI arguments.

Example of running the Python script in the CLI, can be seen in **Figure 17**. Technical review is run in the CLI by specifying `-l` and `-cf` which correspond to Lambda and CloudFormation. This command stores true values for `-l` and `-cf` for which the Python script then executes the specified technical review configuration checks automatically.

```
python3 aws_technical_review.py -l -cf -roles
```

Figure 17. Running Python script with CLI arguments.

The CLI argument `-roles` is used to specify to audit every role in the `configuration.ini` file instead of the current account that is authenticated to AWS.

Configuration.ini is a file that stores variables related to the technical review, such as the roles to audit in the account_arns section, displayed in **Figure 18**.

```
[accounts]
account_arns: [
    "arn:aws:iam::111111111111:role/ReadOnlyRole",
    "arn:aws:iam::222222222222:role/ReadOnlyRole"
]
[roles]
cross_role_name = AWS-Technical-Review-Role
[technical_review]
folder_name = aws_findings
[multiprocessing]
core_amount = 3
[regions]
default_region = eu-west-1
[cloudtrail]
amount_of_months = 6
cloudtrail_log_bucket_region = eu-west-1
```

Figure 18. Configuration.ini file.

Each account that is included in the technical review will have a technical review directory created for it. This directory contains all of the services that were audited and the directory structure for the reviewed services is depicted in **Figure 19**.

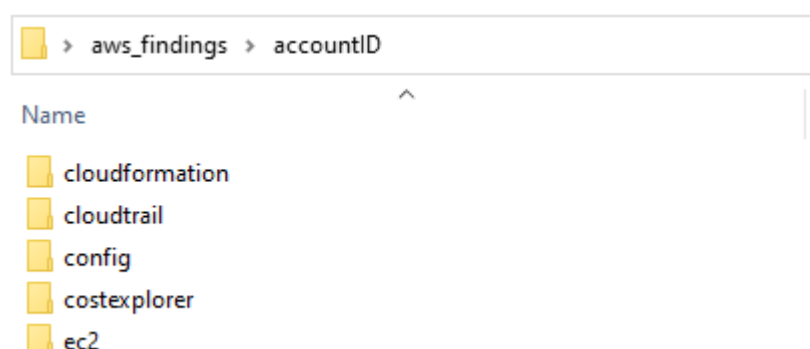


Figure 19. Account directory containing audited AWS services.

Each service directory has a subdirectory based on what AWS regions were audited. AWS region specifies in which region the data center that is used to run the service is located at, regional subdirectories are shown in **Figure 20**.

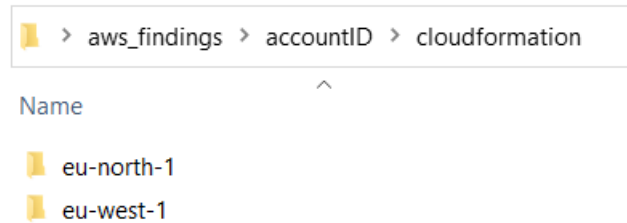


Figure 20. Regional directories for AWS services.

Region specific directories contain all of the auditing files and results from that region, such as the results from code scanners and response data from API requests, displayed in **Figure 21**.

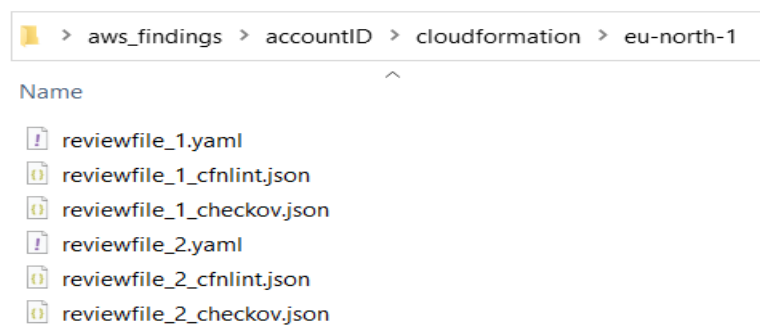


Figure 21. Technical review scan and API response result files.

4.4 Configuration Presentation and Auditing

Parsed AWS service data is gathered into a single-page browser-based web application that uses HTML and JavaScript, making it easy to switch between accounts to check all of the account specific configurations. Data representation is done in HTML by using collapsible elements and tables, which makes it simple for splitting the data into sections and displaying vast amounts of information. Additional Python scripts were used to generate the HTML collapsible and tables from the parsed configuration data.

HTML review template was designed to be easy to use, enabling simple switching between multiple accounts and have the data presented and visualized for easier auditing. AWS account IDs that are being reviewed will be displayed on the left side of the page, and by clicking an account ID, the page is switched to display details of the AWS service configurations that were found for that AWS account ID, see **Figure 22**.

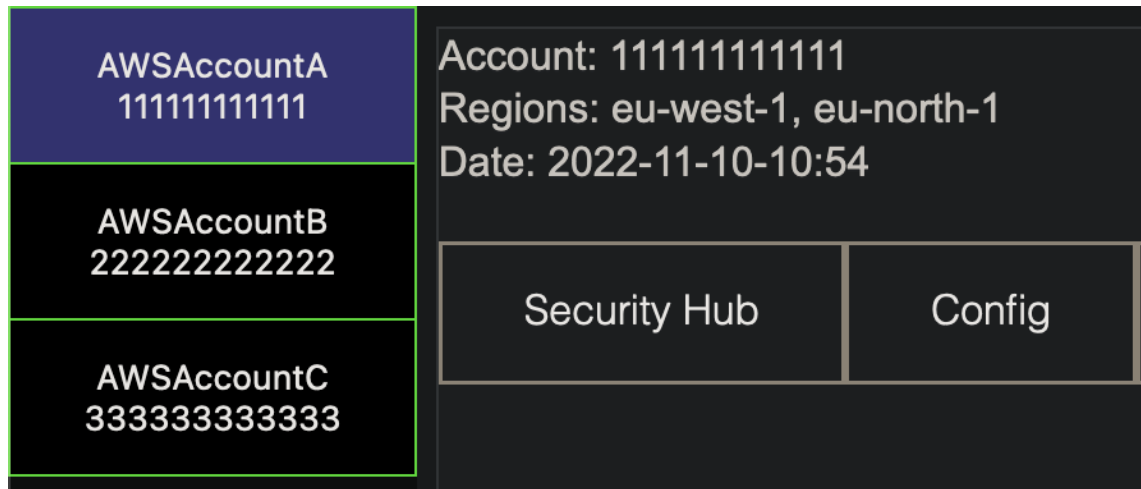


Figure 22. Technical Review account IDs.

Based on what arguments were given to the Python script, the single page application will display configurations for the AWS services and the regions that were specified, see **Figure 23**. AWS service will only be displayed if any configurations are found for the AWS services and if none was found from any of the specified regions, the service will not be displayed on the technical review.

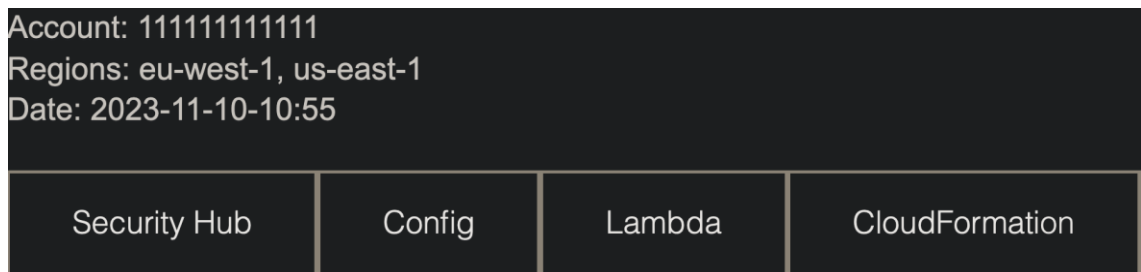


Figure 23. Technical review HTML display page.

Clicking on an AWS service in the HTML page will reveal all of the regions where configuration data was found. This allows the reviewer to audit what results were found on a regional basis and where to apply remediation actions, see **Figure 24**.

Security Hub	Config	Lambda	CloudFormation
eu-west-1			
us-east-1			

Figure 24. Technical review HTML regional collapsible.

Region collapsible contains specified scanner results and configuration details in a collapsible format for more information, depicted in **Figure 25**. This shows that the security scanners found issues regarding the service in the specific region.

Security Hub	Config	Lambda	CloudFormation
eu-west-1			
cfn-lint			
Checkov			

Figure 25. Technical review HTML scanner result collapsible.

Clicking an HTML collapsible that contains results from a scanner or parsed configuration data will be displayed in a table format. Tables contain details such as the severity and description of the issues found in the technical review, Table format is shown in **Figure 26**.

The screenshot shows a web interface for a security scanner named 'Checkov'. It displays a table of findings for a stack named 'database-stack'. The table has five columns: CheckID, CheckName, Resource, Code, and Severity. There are three rows of findings, each with a 'Details' link and a severity level.

CheckID	CheckName	Resource	Code	Severity
CKV_AWS_96	Ensure all data stored in Aurora is securely encrypted at rest	AWS::RDS::DBCluster.DBCluster	Details	HIGH
CKV_AWS_162	Ensure RDS cluster has IAM authentication enabled	AWS::RDS::DBCluster.DBCluster	Details	MEDIUM
CKV_AWS_157	Ensure that RDS instances have Multi-AZ enabled	AWS::RDS::DBInstance.DB	Details	LOW

Figure 26. Technical review HTML security scanner table.

Technical review process is achieved by clicking an account ID and then going through the services and regions to find what issues are found by the security and configuration scanners.

This gives a method to audit multiple AWS accounts with a straightforward way to switch between accounts through the single page application and begin to improve the security and cost posture of the AWS environments.

The output generated by security scanners was designed to be conveniently reviewed from a collapsible HTML object, which allows for easy navigation of code and enables the identification of specific lines that require auditing and remediation, see **Figure 27**.

The screenshot displays the AWS Lambda console interface for a service in the eu-west-1 region. It shows the results of a security scan for a Python script named 'python-script.py'. The scan results are categorized into 'Lint-results' and 'Scan-results'. The 'Scan-results' section shows a table of findings:

Severity	Confidence	CWE	Issue	Information	Code
HIGH	HIGH	https://cwe.mitre.org/data/definitions/327.html	Use of weak MD4, MD5, or SHA1 hash for security. Consider <code>usedforsecurity=False</code>	https://bandit.readthedocs.io/en/1.7.4/plugins/b324_hashlib.html	249 250 251 <pre>m = hashlib.md5() m.update(ip_json)</pre>

Figure 27. Technical review HTML Lambda service review.

Establishing systematic procedures is necessary to manage vulnerabilities found from the security scanners in the auditing process. Critical and high vulnerabilities must be addressed and resolved with a sense of urgency. It is crucial to minimize attack vectors to the greatest extent possible, thereby reducing the overall vulnerable surface.

5 Conclusion

This thesis provided a comprehensive technical review setup for auditing various AWS services, offering insight into their functionalities, features, and security implications. AWS services present potential cost issues and security risks if they are not configured and monitored properly. Scanning tools implemented in this technical review proved effective in identifying misconfigurations, cost issues, exposed resources, and potential vulnerabilities in the AWS environment.

Operation in the cloud should be carefully planned, not a rushed decision that includes careless configurations and architecture that will lead to security or cost issues in the future. It is necessary to have robust security measures in place to protect against cloud breaches, including strong access controls, automatic security scanners and comprehensive security training for the employees.

Cloud providers offer hundreds of different services, created for higher abstraction level and to streamline many business requirements. Automated processes are necessary to assess the critical services that are currently used in the cloud environment as they save time and offer benefits when assessing security and costs of cloud environments.

AWS offers various beneficial services to streamline the process of auditing for security issues, and these should be enabled when possible. It is important to check that monitoring and alerting are set properly to service the environment in case of security breaches and service outages. Automated processes help in reducing human errors and provides an overall better management strategy when implemented properly.

In conclusion, AWS technical review is a crucial step for organizations using AWS to ensure the performance, security and efficiency of their infrastructure. By conducting a thorough review, organizations can identify and address potential security issues, inefficiencies and improve their business operations.

References

- [1] "IDC Forecasts Worldwide "Whole Cloud" Spending to Reach \$1.3 Trillion by 2025." (September 14, 2021). IDC, Needham Mass. Retrieved December 18, 2022, from <https://www.idc.com/getdoc.jsp?containerId=prUS48208321>
- [2] "All Eyes on Cloud | Why the Cloud Surface Attracts Attacks." (October 17, 2022). SentinelOne. Retrieved January 12, 2023, from <https://www.sentinelone.com/blog/all-eyes-on-cloud-why-the-cloud-surface-attracts-attacks/>
- [3] Amazon Web Services. (2022). What is AWS. Retrieved August 24, 2022, from <https://aws.amazon.com/what-is-aws/>
- [4] "How to respond to the increasing costs of cloud: a CIO guide." (August 10, 2022). IBM, Julien Oleg Willard. Retrieved August 26, 2022, from <https://www.ibm.com/blogs/internet-of-things/increasing-cloud-costs-cio-guide/>
- [5] Amazon Web Services. (2022). AWS CloudTrail Documentation. Retrieved July 13, 2022, from <https://docs.aws.amazon.com/cloudtrail/>
- [6] "Internet Crime Complaint Center IC3." (2021). Federal Bureau of Investigation. Retrieved June 18, 2022, from https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf
- [7] Sumit Goyal, "Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review", IJCNIS, vol.6, no.3, pp.20-29, 2014. DOI: 10.5815/ijcnis.2014.03.03
- [8] "The state of Linux in the public cloud for enterprises." (March 5, 2019). Red Hat. Retrieved July, 22, 2022, from <https://www.redhat.com/en/resources/state-of-linux-in-public-cloud-for-enterprises>
- [9] "2020 Set a Record for New Linux Malware Families.", (February 24, 2021). Intezer. Retrieved July, 13, 2022, from <https://www.intezer.com/blog/cloud-security/2020-set-record-for-new-linux-malware-families/>
- [10] Amazon Web Services. (2022). What is Amazon Inspector. Retrieved July 15, 2022, from <https://docs.aws.amazon.com/inspector/latest/user/what-is-inspector.html>

- [11] Amazon Web Services. (2022). AWS Systems Manager Patch Manager. Retrieved June 12, 2022, from <https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html>
- [12] Amazon Web Services. (2022). How Security Hub works, Retrieved June 16, 2022, from <https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-get-started.html>
- [13] "Supply Chain Attacks." (2023). Microsoft. Retrieved February 12, 2023, from <https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/supply-chain-malware?view=o365-worldwide>
- [14] "Zero Trust Field Guide." (2023). IBM. Retrieved February 12, 2023, from <https://www.ibm.com/cloud/architecture/content/field-guide/zero-trust-field-guide/>
- [15] "Zero Trust." (2023). IBM. Retrieved February 12, 2023, from <https://www.ibm.com/topics/zero-trust>
- [16] Amazon Web Services. (2022). Microservices. Retrieved November 6, 2022 from <https://aws.amazon.com/microservices/>
- [17] "Breaking up Monolith Architectures." (2023). Orkes. Retrieved February 18, 2023, from <https://orkes.io/content/blog/Breaking-up-Monolith-Architectures-Part-One>
- [18] Docker Inc. (2022). Docker Get Started. Retrieved December 8, 2022 from <https://docs.docker.com/get-started/>
- [19] Docker Inc. (2022). Docker Best Practices. Retrieved December 8, 2022 from <https://docs.docker.com/develop/dev-best-practices/>
- [20] The Linux Foundation. (2022). Kubernetes Concepts. Retrieved January 12, 2023 from <https://kubernetes.io/docs/concepts/>
- [21] Amazon Web Services. (2023). AWS EKS Best Practices. Retrieved February 16, 2023 from <https://aws.github.io/aws-eks-best-practices/security/docs/>
- [22] Cloud Security Alliance. (2022). Top Threats to Cloud Computing Pandemic Eleven. Retrieved August 24, 2022 from

<https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-pandemic-eleven/>

[23] PyCQA. (2022). Bandit: A Python Source Code Security Analyzer, GitHub Repository. Retrieved July 19, 2022 from <https://github.com/PyCQA/bandit>

[24] Amazon Web Services. (2022). Boto3 Documentation. Retrieved June 7, 2022 from <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

[25] Amazon Web Services. (2022). Cfn-lint: CloudFormation linter, GitHub Repository. Retrieved July 11, 2022 from <https://github.com/aws-cloudformation/cfn-lint>

[26] Bridgecrew. (2022). Checkov: A Static Analysis Tool for Infrastructure-as-Code, GitHub Repository. Retrieved September 21, 2022 from <https://github.com/bridgecrewio/checkov>

[27] Salesforce. (2022). Cloudsplaining: AWS IAM Security Assessment Tool, GitHub Repository. Retrieved August 12, 2022 from <https://github.com/salesforce/cloudsplaining>

[28] Duo Labs. (2022). Parliament: AWS IAM Linting Library, GitHub Repository. Retrieved August 24, 2022 from <https://github.com/duo-labs/parliament>

[29] Prowler. (2022). Prowler: Cloud Security Best Practices Assessment Tool, GitHub Repository. Retrieved June 20, 2022 from <https://github.com/prowler-cloud/prowler>

[30] PyCQA. (2022). Pylint: Python linter, GitHub Repository. Retrieved August 21, 2022 from <https://github.com/PyCQA/pylint>

[31] Aquasecurity. (2022). Trivy: DevOps Vulnerability Scanner, GitHub Repository. Retrieved October 11, 2022 from <https://github.com/aquasecurity/trivy>.

[32] Gartner (2023). Cloud Infrastructure and Platform Services. Retrieved January 19, 2023 from <https://www.gartner.com/reviews/market/public-cloud-iaas>

[33] Amazon Web Services. (2022). AWS Global Infrastructure. Retrieved June 13, 2022 from <https://aws.amazon.com/about-aws/global-infrastructure/>

- [34] Amazon Web Services. (2022). About AWS. Retrieved June 9, 2022 from <https://aws.amazon.com/about-aws/>
- [35] Amazon Web Services. (2023). Serverless Architecture. Retrieved January 22, 2023 from <https://aws.amazon.com/lambda/serverless-architectures-learn-more/>
- [36] Amazon Web Services. (2023). Amazon VPC Documentation. Retrieved Jan 11, 2023 from <https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html>
- [37] Amazon Web Services. (2023). Shared responsibility model Documentation. Retrieved February 6, 2023 from <https://aws.amazon.com/compliance/shared-responsibility-model/>
- [38] Amazon Web Services. (2022). AWS Well Architected Framework. Retrieved July 28, 2022 from <https://docs.aws.amazon.com/wellarchitected/latest/framework/the-pillars-of-the-framework.html>
- [39] Amazon Web Services. (2022). Athena Documentation. Retrieved November 6, 2022 from <https://docs.aws.amazon.com/athena/>
- [40] Amazon Web Services. (2022). CloudFormation. Retrieved August 26, 2022 from <https://docs.aws.amazon.com/cloudformation/>
- [41] Amazon Web Services. (2022). CloudTrail Workflow Documentation. Retrieved November 28, 2022 from <https://docs.aws.amazon.com/awscloudtrail/latest/userguide/how-cloudtrail-works.html>
- [42] “Best practices for monitoring AWS CloudTrail logs.” (September 25, 2020). Datadog, Justin Massey and Jonathan Epstein. Retrieved October 6, 2022 from <https://www.datadoghq.com/blog/monitoring-cloudtrail-logs/>
- [43] Amazon Web Services. (2022). AWS Config. Retrieved September 2, 2023 from <https://docs.aws.amazon.com/config/>
- [44] Amazon Web Services. (2022). Cost Management. Retrieved August 4, 2022 from <https://docs.aws.amazon.com/cost-management/latest/userguide/what-is-costmanagement.html>

- [45] Amazon Web Services. (2023). EC2 Documentation. Retrieved January 8, 2023 from <https://docs.aws.amazon.com/ec2/>
- [46] Amazon Web Services. (2023). AMI Documentation. Retrieved January 8, 2023 from <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html>
- [47] Amazon Web Services. (2023). EC2 Instance types. Retrieved January 8, 2023 from <https://aws.amazon.com/ec2/instance-types/>
- [48] Amazon Web Services. (2023). EC2 EBS Documentation. Retrieved January 9, 2023 from <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html>
- [49] Amazon Web Services. (2023). EC2 EFS Documentation. Retrieved January 9, 2023 from <https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html>
- [50] Amazon Web Services. (2023). EC2 Security Groups. Retrieved January 10, 2023 from <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html>
- [51] Amazon Web Services. (2022). ECR Documentation. Retrieved October 19, 2022 from <https://docs.aws.amazon.com/ecr/>
- [52] Amazon Web Services. (2022). ECR Description. Retrieved October 19, 2022 from <https://aws.amazon.com/ecr/>
- [53] Amazon Web Services. (2022). EKS Documentation. Retrieved December 13, 2022 from <https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html>
- [54] Amazon Web Services. (2022). IAM Documentation. Retrieved July 23, 2022 from <https://docs.aws.amazon.com/iam/>
- [55] Amazon Web Services. (2022). IAM Getting Started. Retrieved July 23, 2022 from <https://aws.amazon.com/iam/getting-started/>
- [56] Amazon Web Services. (2022). Lambda Documentation. Retrieved October 29, 2022 from <https://docs.aws.amazon.com/lambda/>
- [57] Amazon Web Services. (2022). S3 Documentation. Retrieved November 17, 2022 from <https://docs.aws.amazon.com/s3/>

[58] Amazon Web Services. (2022). S3 Features. Retrieved November 17, 2022 from <https://aws.amazon.com/s3/features>

[59] Amazon Web Services. (2022). Security Hub Documentation. Retrieved June 27, 2022 from <https://docs.aws.amazon.com/securityhub/>

[60] Amazon Web Services. (2022). Key Management Service. Retrieved December 17, 2022 from <https://docs.aws.amazon.com/kms/latest/developerguide/overview.html>

Configuration templates

Boto3 example.

```
# Importing the required Python module boto3, to use its features
import boto3

# Creating CloudWatch boto3 client, that allows to use API requests related
to CloudWatch.
cloudwatch_client = boto3.client('cloudwatch')

# Sending API call called describe_alarms() through the created CloudWatch
boto3 client. This will return a JSON response that contains every alarm
in CloudWatch
describe_alarms_json_response = cloudwatch_client.describe_alarms()

# Boto3 functions often have parameter options that you can use to create
functions to fit your needs. Example for describing a single alarm based
on its name.
describe_alarms_json_response = cloudwatch_client.describe_alarms(
AlarmNames=['CustomAlarmName']
)
```

IAM Policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UsedToDescribeWhatPolicyIsUsedFor",
      "Effect": "Allow",
      "Action": [
        "securityhub:DescribeHub",
        "securityhub:GetFindings",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets"
      ],
      "Resource": "*"
    }
  ]
}
```

CloudTrail API request event [14].

User Alice (userName) made an API request to create a new user(eventName) named Bob(requestParameters).

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAAAAAAAAAAAAAAAAAAAA",
    "arn": "arn:aws:iam::111111111111:user/Alice",
    "accountId": "111111111111",
    "accessKeyId": "AKIAAAAAAAAAAAAAAAAAAAAA",
    "userName": "Alice"
  },
  "eventTime": "2020-09-21T10:31:20Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "CreateUser",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "1.2.3.4",
  "userAgent": "console.amazonaws.com",
  "requestParameters": {
    "userName": "bob",
    "tags": []
  },
  "responseElements": {
    "user": {
      "path": "/",
      "userName": "bob",
      "userId": "AIDBBBBBBBBBBBBBBBBBBB ",
      "arn": "arn:aws:iam::111111111111:user/bob",
      "createDate": "Sep 21, 2020 10:31:20 AM"
    }
  },
  "requestID": "604e7549-4ea4-4185-83b0-acff4e462d27",
  "eventID": "600e50af-0a2c-4352-95a8-7b813c744072",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111111111111"
}
```

ASFF Security Hub finding format.

```
{
  "SchemaVersion": "2018-10-08",
  "Id": "containerName:containerTag/cveId",
  "ProductArn": "arn:aws:securityhub:awsRegion::product/aqs/aqs",
  "GeneratorId": "generatorid",
  "AwsAccountId": "awsaccountid",
  "Types": [ "Software and Configuration Checks/Vulnerabilities/CVE" ],
  "CreatedAt": "iso8601Time", "UpdatedAt": "iso8601Time",
  "Severity": {
    "Product": "trivyProductSev", "Normalized": "trivyNormalizedSev"
  },
  "Title": "Trivy found a vulnerability to cveId in container
containerName",
  "Description": "cveDescription",
  "Remediation": {
    "Recommendation": {
      "Text": "More information on this vulnerability is provided in
the hyperlink",
      "Url": "cveReference"
    }
  },
  "ProductFields": { "Product Name": "Trivy" },
  "Resources": [
    {
      "Type": "Container",
      "Id": "containerName:containerTag",
      "Partition": "aws",
      "Region": "awsRegion",
      "Details": {
        "Container": { "ImageName" : "containerName:containerTag"
},
        "Other": {
          "CVE ID": "cveId",
          "CVE Title": "cveTitle",
          "Installed Package": "packageName:installedVersion",
          "Patched Package": "packageName:fixedVersion"
        }
      }
    }
  ],
  "RecordState": "ACTIVE"
}
```

CloudTrail Athena SQL Table.

```

CREATE EXTERNAL TABLE cloudtrail_table(
  eventVersion STRING,
  userIdentity STRUCT<
    type: STRING, principalId: STRING,
    arn: STRING, accountId: STRING,
    invokedBy: STRING, accessKeyId: STRING,
    userName: STRING,
    sessionContext: STRUCT<
      attributes: STRUCT<
        creationDate: STRING>,
      sessionIssuer: STRUCT<
        type: STRING, principalId: STRING,
        arn: STRING, accountId: STRING,
        userName: STRING>>>>,
  eventTime STRING, eventSource STRING, eventName STRING,
  awsRegion STRING, sourceIpAddress STRING, userAgent STRING,
  errorCode STRING, errorMessage STRING, requestParameters STRING,
  responseElements STRING, additionalEventData STRING,
  requestId STRING, eventId STRING, readOnly STRING,
  resources ARRAY<STRUCT<
    arn: STRING,
    accountId: STRING,
    type: STRING>>,
  eventType STRING, apiVersion STRING,
  recipientAccountId STRING, serviceEventDetails STRING,
  sharedEventID STRING, vpcendpointid STRING)
PARTITIONED BY (`region` string, `timestamp` string)
ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
    's3://aws-cloudtrail-logs-${account_arn}-
q21d11cs/AWSLogs/${account_arn}/CloudTrail'
TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2022/01/01,NOW',
  'projection.timestamp.type'='date',
  'projection.region.values'='eu-west-1',
  'projection.region.type'='enum',
  'storage.location.template'='s3://aws-cloudtrail-logs-${account_arn}-
q21d11cs/AWSLogs//${account_arn}/CloudTrail/${region}/${timestamp}')

```

SQL query for CloudTrail Athena table.

```
SELECT
*
FROM cloudtrail_table
where timestamp >= '2022/06/01' and timestamp <= '2022/09/30'
and region in ('eu-west-1')
and eventname in ('ConsoleLogin', 'AssumeRole')
and useridentity.type in ('IAMUser', 'FederatedUser', 'AWSAccount',
'AssumedRole')
```

CloudFormation JSON template.

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation template for s3 bucket

Resources:
  S3Bucket:
    DeletionPolicy: Retain
    Type: 'AWS::S3::Bucket'
    Description: Creating Amazon S3 bucket from CloudFormation
    Properties:
      AccessControl: Private
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256
      VersioningConfiguration:
        Status: Enabled
Outputs:
  S3Bucket:
    Description: Bucket Created using this template.
    Value: !Ref S3Bucket
```