# Comprehensive Approach for Developing an Enterprise Cloud Infrastructure

Volodymyr Khoma[1], Aziz Abibulaiev[1], Andrian Piskozub[1], and Taras Kret[1]

[1]*Lviv Polytechnic National University, 12 S. Bandera str., Lviv, 79000, Ukraine*

**Abstract**

The modern world is witnessing an incredibly rapid adoption of cloud technologies across various sectors, making security in cloud environments extremely relevant. The problem lies in the lack of a comprehensive approach to ensure multilevel protection for cloud infrastructure solutions and the limitations of the existing "Security as Code" practice, which would provide effective protection in this environment and meet modern cybersecurity standards. This article aims to describe the process of implementing and deploying a comprehensive approach to building an enterprise cloud infrastructure and ensuring its security. The relevance of this research is confirmed by the constantly growing interest in cloud technologies and the need to ensure a high level of security during their usage. The aspects considered in the article can significantly facilitate the process of infrastructure development, improve its security, and help prevent potential threats. Therefore, the ultimate goal of the article is to implement a comprehensive approach to developing a cloud infrastructure solution and ensuring its security, using "Infrastructure as Code" and "Security as Code" practices. The results of this research can be used by organizations planning or already using cloud technologies to familiarize themselves with best practices for infrastructure development to enhance the security level of their information resources and prevent possible threats.

**Keywords**

Defense-in-depth, Infrastructure as code, Security as code, BCP, DevOps, AWS, ETL, software development cycle, cloud security threats, testing.

## 1. Introduction

In the context of modern trends and the development of both computing power and computing services, the use of physical servers and equipment is no longer the optimal or sole solution [1]. The constant and thorough development of computing resources over the last few decades does not allow us to predict the optimal amount of resources for computations. A company that acquired powerful physical servers and other technical resources a few years ago cannot be confident that the existing resources of these tools will be sufficient for further development, implementation, and support of the product. There arises a need for continuous improvement of methods and approaches to software development and infrastructure management, which is often not possible for an average product IT company due to the financial capabilities and technical competence of the staff.

One of the key elements of modern infrastructure is the transition from physical equipment to cloud solutions, which allows it to effectively address issues that physical computing resources cannot resolve. The large number of data centers worldwide, ensuring resource availability from anywhere, continuous improvement of existing technical tools and creation of new services, and the relevance of tools and services to user needs make cloud provider services the best choice in today's realities [2, 3].

CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2024, Kyiv, Ukraine
EMAIL volodymyr.v.khoma@lpnu.ua (V. Khoma); aziz.abibulaiev.mkbui.2022@lpnu.ua (A. Abibulaiev); andriian.z.piskozub@lpnu.ua (A. Piskozub); taras.b.kret@lpnu.ua (T. Kret)
ORCID: 0009-0009-2289-0968 (V. Khoma); 0000-0003-3037-8373 (A. Abibulaiev); 0000-0003-3037-8373 (A. Piskozub); 0000-0002-6333-3190 (T. Kret)

The article aims to analyze the use of modern approaches and technologies in the field of cloud infrastructure to overcome problems associated with the limitations of the relevance of computing power of physical servers in the long term; the limitation and complexity of infrastructure resource scaling implementations; the inconvenience and maladjustment of continuous data backup; the limited level of computational resource availability; the complexity of settings and configuration templates; the development of a comprehensive solution to protect the cloud infrastructure of a product IT company from threats such as DDoS attacks, port scanning, SQL injections, XSS-attacks, information leaks, and others [46].

The proposed solution aims to address such important aspects of cloud infrastructure security as effective access and privilege management; logical isolation of network resources and their configuration; continuous monitoring, event logging, and immediate response to anomalies and dangers; user notification mechanism about threats; system application vulnerability checks (posture checking); information confidentiality checks; data encryption tools; network traffic protection tools; network traffic filtering and control.

The use of the “Security as Code” approach in this context allows for the integration of security into DevOps processes, ensuring timely detection and correction of vulnerabilities. This approach not only strengthens the protection of users and data at the level of cloud infrastructure but also covers comprehensive monitoring and access management, network resource isolation, continuous monitoring, and rapid response to anomalies.

## 2. Comparison of Physical and Cloud Infrastructure

In this article, the authors focus on understanding the appropriate approach to deploying infrastructure and ensuring security through cloud provider services, using real-world experience from an IT company with an Extract, Transform, and Load (ETL) solution [2].

Nowadays, any work with infrastructure, deployment, and product operation must follow the processes of the Software Development Life Cycle (SDLC) [3]. Therefore, the authors used the SDLC with a focus on infrastructure deployment: planning and selecting a platform for deployment; defining requirements for resource capabilities; designing infrastructure architecture; configuring infrastructure; testing; and deployment.

Let’s start with the first point, namely the planning and selection of a platform for infrastructure deployment. Let’s consider some aspects and characteristics of physical and cloud servers that will help in choosing a platform.

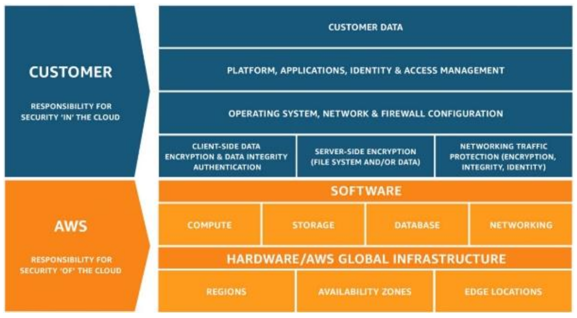
The rapid development of technologies renders physical technical resources obsolete over time, and Roy Longbottom [4] proves this by comparing the Cray-1 supercomputer, built in 1978, costing 7 million USD, with the modern Raspberry Pi, which costs 70 USD and is 4.5 times faster than the Cray-1. Therefore, renting technical resources from a cloud provider is a solution to this situation. The cloud provider is confident in the relevance of its technologies and has the capability, financial capacity, and interest in its developments in the field of computing technology, operating systems, and security. For example, Amazon Web Services (AWS) uses its development, the AWS Nitro System for EC2 virtual machines, which significantly improves the performance of the processor, virtual memory, and data storage, enhances system security and is a cheaper solution compared to other EC2 systems.

Physical computing resources are a company’s capital investment, where the owner tries to select the optimal amount of resources. However, it is quite difficult to predict the sufficient amount of resources for all cases and events in the company. On the other hand, maintaining a large amount of resources is expensive and not an optimal solution. This problem is solved by the cloud provider, where the customer can both obtain a sufficient amount and opt out of excess resources at any time, choosing a convenient payment option—hourly, annually, or reserving the necessary computing capacities. This allows for more flexibility in the software product testing process and predicting server loads. The complexity of ensuring the fault tolerance of computing servers lies in the need for constant monitoring, backup support, and automated recovery in case of failure. Without configured fault tolerance of the software product, there are high risks of errors and failures, which, in turn, can lead to financial losses, stopping the product and business as a whole. The use of cloud solutions simplifies these tasks, as most cloud service providers guarantee a high level of availability and uninterrupted operation thanks to distributed data processing centers in many places around the world, as well as backup mechanisms.

The scalability of the infrastructure solution is an important aspect of growing business tasks. Cloud providers offer automatic scaling, which allows increasing or decreasing resources according to needs, which is a more efficient and economical solution than the traditional purchase or upgrade of physical equipment.

In cloud systems users can easily configure computing power, storage, network settings, and other components, ensuring high flexibility and cost optimization through the Infrastructure as Code (IaC) tool.

Another important aspect when working with cloud resources is the shared responsibility for infrastructure security. The cloud provider is responsible for ensuring the security of the "cloud" itself: physical resources; software that ensures the operation of the global network, databases, and computing resources. The cloud service customer is responsible for ensuring the security of resources "in" the cloud: user data; customer-side data encryption; operating systems, firewall configurations, etc. The general model is shown in the figure (Fig. 1).

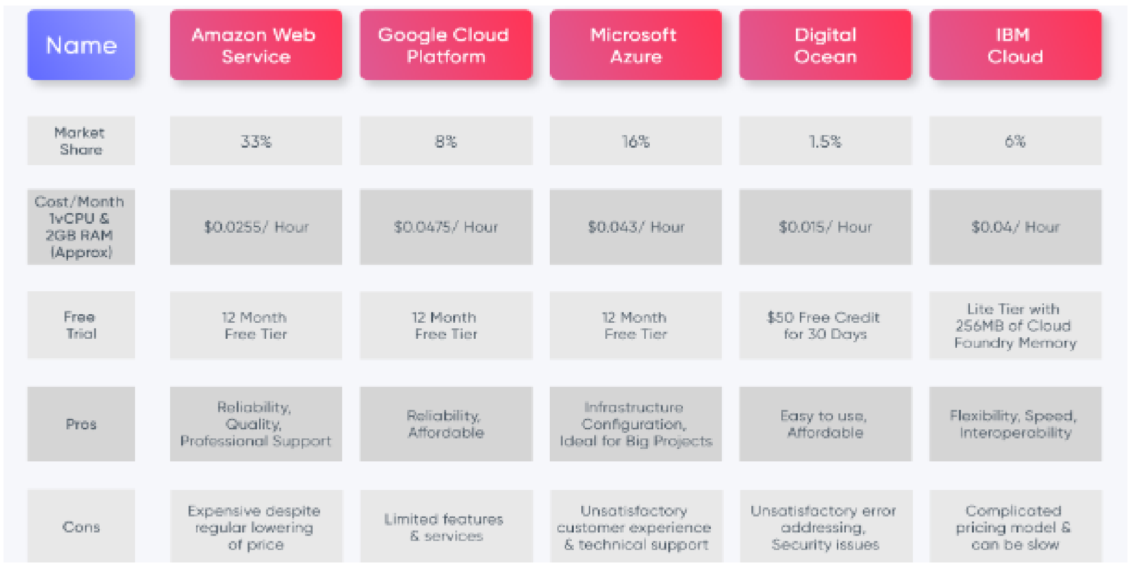

CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS
**Figure 1:** AWS Shared Responsibility model [5]

The convenience of properly distributing responsibilities between the cloud provider and the customer helps to clearly understand and define who is responsible for what, speeding up and facilitating the configuration of infrastructure and simplifying the customer's life. In systems with physical servers, the server owner is responsible for everything.

Considering the aforementioned aspects and characteristics of comparing physical and cloud servers, we chose the cloud platform.

## 3. Choosing a Cloud Provider

Among the main cloud service providers are Amazon Web Services, Google Cloud Platform, Microsoft Azure, and IBM Cloud, which offer their services to users worldwide. Recro [6], a developer company working in the Cloud Solutions field since 2015, analyzed the main cloud providers based on market share indicators, monthly costs for basic virtual machine configurations, free trial licenses for cloud resources, their advantages and disadvantages (Fig. 2):



| Name | Amazon Web Service | Google Cloud Platform | Microsoft Azure | Digital Ocean | IBM Cloud |
|---|---|---|---|---|---|
| Market Share | 33% | 8% | 16% | 1.5% | 6% |
| Cost/Month 1vCPU & 2GB RAM (Approx) | $0.0255/ Hour | $0.0475/ Hour | $0.043/ Hour | $0.015/ Hour | $0.04/ Hour |
| Free Trial | 12 Month Free Tier | 12 Month Free Tier | 12 Month Free Tier | $50 Free Credit for 30 Days | Lite Tier with 256MB of Cloud Foundry Memory |
| Pros | Reliability, Quality, Professional Support | Reliability, Affordable | Infrastructure Configuration, Ideal for Big Projects | Easy to use, Affordable | Flexibility, Speed, Interoperability |
| Cons | Expensive despite regular lowering of price | Limited features & services | Unsatisfactory customer experience & technical support | Unsatisfactory error addressing, Security issues | Complicated pricing model & can be slow |

**Figure 2:** Comparison of main cloud providers based on indicators [6]

Thus, we chose Amazon Web Services as the cloud provider, as the provider with the widest range of services and the status of the first and innovative leader among cloud providers.

## 4. Developing a Solution Architecture with BCP Implementation

The next step is to develop the infrastructure architecture with a focus on the requirements of the Business Continuity Plan (BCP), which were thoroughly described in [7].

The ETL solution is a tool for various business needs, where it is necessary to extract, transform, and load data from one environment to another, and this process can be scalable. It is a product that can help obtain a real picture of enterprise resource planning reporting, and performs the task of consolidating all data into a single system of values and details, ensuring their quality and reliability. Also, ETL is a mechanism for ensuring data audit, allowing even after their transformation to track the origin and exactly what each row of the table, text, video file, and other types of data were obtained from.

The customer's ETL solution consists of microservices and a relational PostgreSQL database. This means that at the level of cloud infrastructure and services, we need to ensure BCP, which is quite a critical component in the continuity of business processes and minimizing losses.

In determining the optimal platform for deploying our ETL solution, the authors of the article conducted a detailed analysis of the services offered by AWS, focusing on the capabilities they provide for microservice architectures. Based on [8], an analysis of platforms for microservices from AWS was conducted, namely—Elastic Container Service (ECS) and Elastic Kubernetes Service (EKS). EKS best meets the needs of the ETL solution due to its high level of management of virtual machines and effective load balancing.

In the context of load management, EKS demonstrates excellent capabilities, automatically distributing traffic between containers to optimize performance and service availability. This ability for effective load balancing is key to ensuring the resilience and efficiency of microservices ETL applications, which depend on flexible scaling and reliability in data processing.

Let's consider the mechanisms for ensuring BCP. For the PostgreSQL database, AWS Relational Database Service (RDS) was chosen, which provides reliability and convenience in managing databases. Using the AWS RDS solution, we will create an inter-regional, highly available cluster structure for the PostgreSQL database.

There are two types of database clustering—master-master and master-replica. The master-master relationship in database architecture allows two or more servers to function as primary (master), providing simultaneous writing and updating of data, which increases availability and provides distributed data management, and is important for systems with a high load level or to ensure uninterrupted operation. In the master-replica architecture, one server acts as the primary (master), and one or more other servers act as replicas (slaves). The primary server handles all writes and updates and the replicas synchronize these changes with the primary server, allowing them to respond to read requests. Such architecture increases read performance and ensures fault tolerance by distributing the load among several servers.

Based on [9], for the case of database clustering of the ETL solution, we will use a combination of master-master and master-replica relationships to ensure both uninterrupted operation and load distribution among databases (Fig.3).

BCP is defined by such indicators as Recovery Time Objective (RTO)—an acceptable measure in the amount of time before the system or application is fully restored and returned to normal operation after an unexpected interruption or disaster, and Recovery Point Objective (RPO)—a measure of data loss that is acceptable to the business. It should be noted that RTO and RPO are entirely individual indicators for each business [10]. Everything depends on the criticality of the information processed by the product, the degree of fault tolerance, and acceptable losses in case of failures.

With a fairly simple configuration, it is possible to achieve settings for both the repeatable backup process and database dumps. These processes precisely provide RTO and RPO in the architecture of the ETL solution.

In our case, these backups will be made with a period of 1, 6, 12 hours, 1, 7, 14 days. Therefore, we will be able to roll back to a working state with a loss of information exactly in 1 hour of work, which is sufficient in our case and meets the customer's expectations.

Summarizing the conclusions of the aforementioned aspects, the design of the ETL solution architecture was built, which is depicted in the diagram (Fig.3).

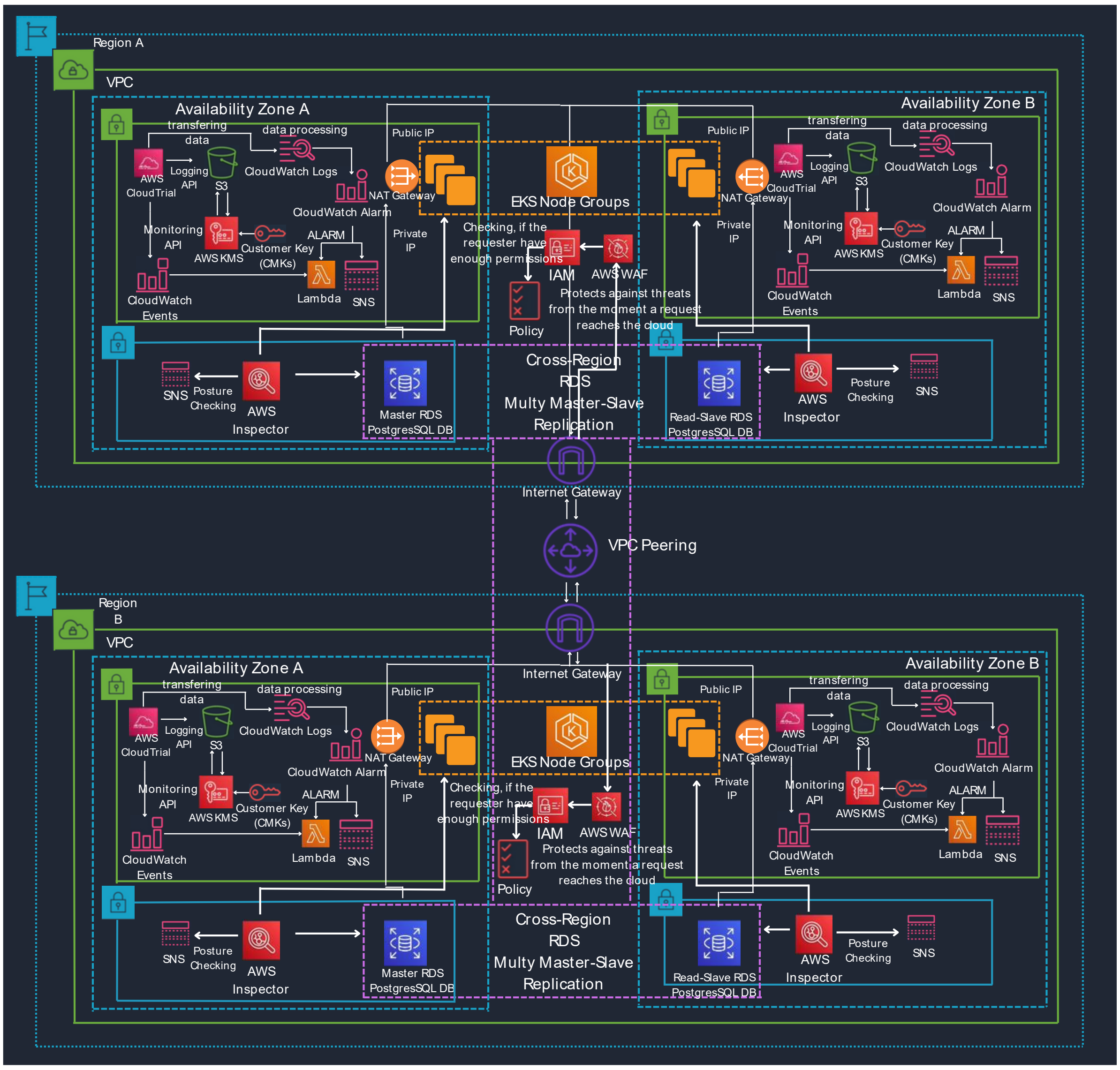


Figure 3: ETL application architecture

In any case, whether there are problems with the database instance in Region A or the availability zone, there will always be either Region B with the master database or a replica in another availability zone in Region A available. Transferring the roles of the main database to the replica happens very quickly and can be done in just a few mouse clicks, so business losses are limited to minutes.

The application itself is a Stateless resource, so scaling at the level of availability zones will be sufficient. The EKS cluster, like any Kubernetes product, can manage its resources and containers in pods using internal mechanisms, so the proper configuration will allow scaling and maintaining microservices in a continuous, working state.

## 5. Infrastructure Configuration Using IaC

The next step in the SDLC stages is infrastructure configuration, which, in the case of the architecture proposed in Fig. 3 can be implemented using IaC, a powerful and flexible tool for configuring infrastructure in cloud environments. One can agree with the author's opinion in the article [11] that "...the deployment time of infrastructure using Infrastructure as Code strategies average 60 seconds with the use of 'automation', while standard 'manual' deployment strategies take an average of 600 seconds, regardless of the cloud service provider". IaC allows for control and management of infrastructure, ranging from the configuration of the virtual machine or server itself (defining the type of operating system, the number of processors, memory, and storage capacity) to the full deployment of the product on this infrastructure, including setting up SSL/TLS encryption, automation of microservices launch.

The author's work [12] highlights Terraform, Pulumi, and other more cloud-native solutions like CloudFormation in AWS among the tools for implementing Infrastructure as Code. Terraform and Pulumi are called universal. It should be added that, in our opinion, Pulumi is not yet a stable enough tool for a production-ready solution, so users still prefer Terraform. The mentioned tools have a quite serious documentation base, so they are not very complicated to learn.

According to the comparative analysis of Terraform and AWS CDK tools [13], one can fully agree with the aspects that were compiled into a table form, which was supplemented by our observations and is presented below (Table 1):

**Table 1**
Comparative Characteristics of Cloud-Native IaC with Terraform

| Feature | AWS CDK | Terraform |
|---|---|---|
| Performance | Synthesizes code into CloudFormation format, affecting deployment and update times. | Performs better in all test cases due to the ability to interact directly with the underlying AWS APIs. |
| Learnability | Easy to install and start with. The abstraction layer reduces complexity and the amount of code needed for implementation. | Quick installation and easy to start with. The DSL requires time to learn and adapt. Sometimes describing infrastructure requires a lot of repetitive code (which can also be minimized using a modular approach to writing code). |
| IDE Support | Using GPL allows for better IDE integration and support, as well as receiving immediate feedback and errors in tool implementation. | Good tool development support, code navigation, and syntax highlighting are available. |
| Community Support | AWS CDK offers extensive and well-structured documentation, serving as a valuable resource for users. | A popular tool with many materials and real user cases created by the community, with electronic blogs, articles, and extensions to many IDEs. |
| Testability | Supports Jest, unit testing, and snapshot testing. Simplicity and speed in test development and testing itself. | Supports Terratest and integration tests. Comparative complexity in writing tests and their implementation with AWS CDK, but Terraform tests are better at detecting errors. |
| Static Code Analysis Capability | The abstraction layer and the possibility of "smart" default settings reduce the time to detect misconfigurations. Using ESLint and Prettier for code formatting and linting. | Everything is configured by the user, leaving room for errors and deficiencies. Uses a built-in formatting tool, code linting with TFLint. |

The conducted analysis allows choosing a tool for implementing IaC according to the needs—a universal tool or a cloud-native solution. The authors of the article prefer Terraform as a universal tool with a large documentation base, providers, and examples of working infrastructure code.

Here is an example of configuring the creation of a VPC with Internet Gateway for infrastructure using code from the official Terraform documentation source [14]:

```
data "aws_region" "current" {}
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"
  region           = data.aws_region.current.id
  tags = {
    Name = "main"
  }
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "main"
  }
}
```

The next part of the code creates the Public Subnet and the necessary resources to make the subnetwork:

```
resource "aws_subnet" "public_subnet" {

  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.1.0/24"
  map_public_ip_on_launch = "true"

  tags = {
    Name = "main"
  }
}

resource "aws_route_table" "public_subnet" {
  vpc_id           = aws_vpc.main.id
  route              = {
  cidr_block = "10.0.1.0/24"
  gateway_id   = aws_internet_gateway.gw.id
 }
  tags = {
    Name = "main"
  }
}

resource         "aws_route_table_association"
"public_subnet" {
  subnet_id      = aws_subnet.public_subnet.id
               route_table_id              =
aws_route_table.public_subnet.id
}
```

The last part of the provided Terraform code is for creating a Private Subnet, adding a Network Address Translation (NAT) Gateway, and other necessary things to make the private subnetwork

```
resource "aws_eip" "example" {
  domain = "vpc"

  depends_on = [ aws_internet_gateway.gw ]
}

resource "aws_nat_gateway" "private_nat" {
  allocation_id = aws_eip.example.id
  subnet_id     = aws_subnet.example.id

  tags = {
    Name = "Private Subnet gw NAT"
  }
  depends_on = [aws_internet_gateway.gw]
}

resource "aws_subnet" "private_subnet" {

  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.0.2.0/24"

  map_public_ip_on_launch = "false"

  tags = {
    Name = "main"
  }
}

resource "aws_route_table" "private_subnet" {

  vpc_id           = aws_vpc.main.id
  route              = {
     cidr_block = "10.0.2.0/24"
                       allocation_id         =
aws_nat_gateway.private_nat.id
  }
  tags = {
     Name = "main"
  }
}
```

```
resource         "aws_route_table_association"
"private_subnet" {
         subnet_id                            =
aws_subnet.private_subnet.id
                route_table_id                =
aws_route_table.private_subnet.id
}
```

With this configuration code, a VPC, Internet Gateway, Public and Private subnets, NAT Gateway—used for accessing the Internet from the Private subnet, Route tables, and associations of these tables with subnets are created. This all allows for a configured network for further deployment of necessary resources. Ensuring the security of the infrastructure is no less important aspect of the continuous and secure operation of the business process. The article [12] extensively discusses the approach to ensuring security that has gradually emerged with the development of IaC—Security as Code (SaC).

## 6. Infrastructure Security Using the "Security as Code" Approach

The existing "Security as Code" implementation approach by O'Reilly [15], in our opinion, is quite limited and incomplete in terms of security solutions. "Security as Code" should be a comprehensive approach that ensures security at all levels of the product's life and the infrastructure as a whole.

Security as code is an approach to ensuring the security of cloud infrastructure that allows controlling tools and measures, security policies, infrastructure and applications settings, authentication and authorization rules, and access restrictions through code, adding flexibility and convenience in management and continuous improvement of security.

Let's consider the problems any modern enterprise planning to use cloud resources for building infrastructure might face, and the solutions a cloud provider can offer, in terms of creation, configuration, and support, as code. Without effective access and privilege management, uncontrolled access to cloud environment resources can cause significant damage to the infrastructure and is a threat to the integrity, confidentiality, and availability of processed information. Without a proper means of delegating access to certain resources, a system engineer cannot granularly control the access an employee might have. In such a case, they have two options: grant the employee access to all resources or deny access to all resources. This is where the problem of either Insufficient Permissions (not enough access rights) or Redundant Permissions (excessive access rights) arises.

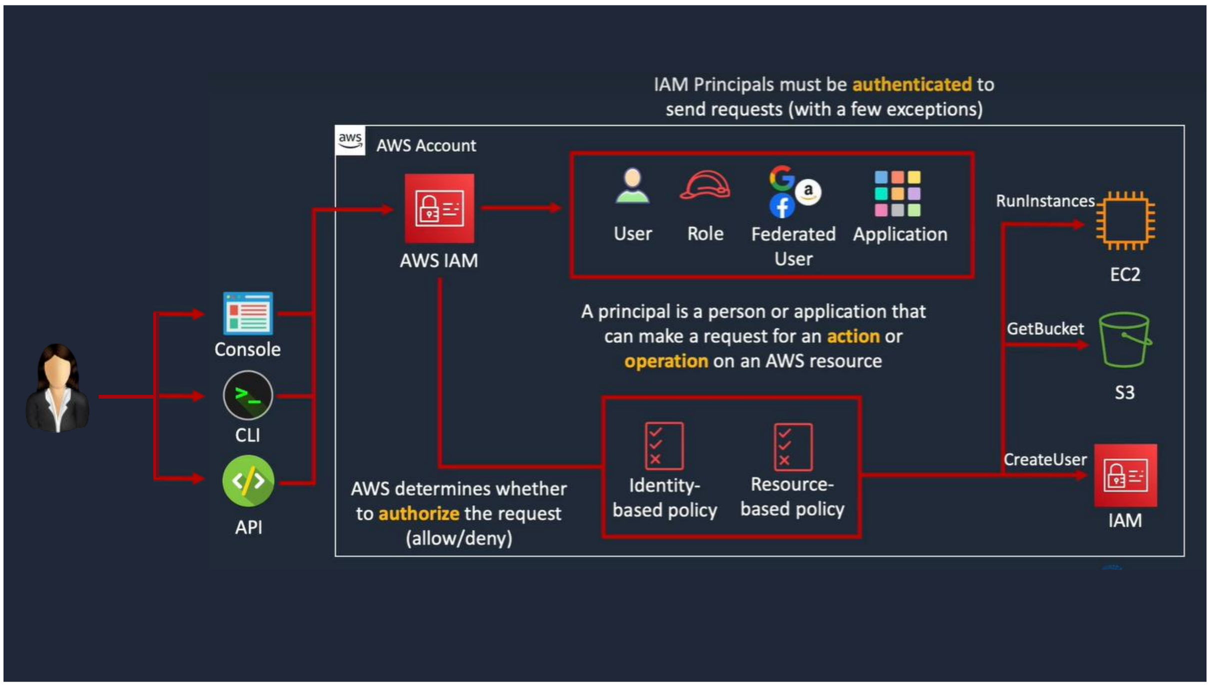

IAM Principals must be authenticated to send requests (with a few exceptions)
AWS Account
AWS IAM
User
Role
Federated User
Application
RunInstances
EC2
GetBucket
S3
CreateUser
IAM
Console
CLI
API
A principal is a person or application that can make a request for an action or operation on an AWS resource
AWS determines whether to authorize the request (allow/deny)
Identity-based policy
Resource-based policy

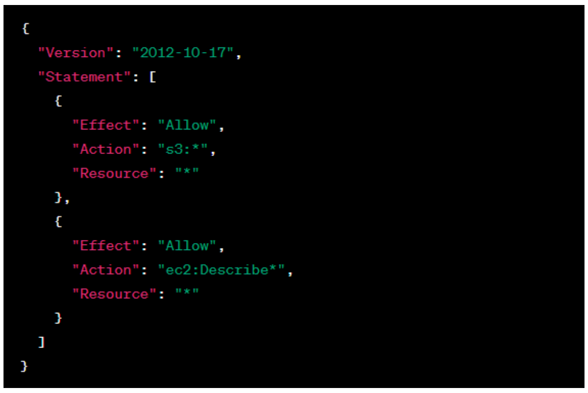
**Figure 4:** IAM operation demonstration

IAM offers us a fundamentally new solution (Fig. 4). An employee can authenticate using the web console (AWS Console), utilizing the Application Programming Interface (API), and tools from the AWS Command Line Interface (CLI), and within the IAM environment, assume the status of a user, role, federated user, or application. By assigning policies (Identity and Resource-based) to the user, they can gain access to various resources such as IAM, S3, and EC2, performing actions on them by sending requests—creating an IAM user, retrieving a file from the S3 object storage, and launching a virtual machine in the EC2 environment. This mechanism provides flexible access management to a particular service within the cloud provider's environment.

Authentication can occur through login-password schemes for web access and using ACCESS_KEY and SECRET_ACCESS_KEY for programmatic use, for example, using the boto3 client for Python or AWS CLI.

It can be said that the main component of the IAM service is the security policy, provided to the entity as an Identity-based policy or Resource-based policy. Security policies are blocks of code that define the following attributes: policy name; description of privileges; permissions; and subject.

These attributes allow IAM to determine whether a user has permission to act on the resource they are requesting. Let's provide an example of a policy that grants a user administrator access to the S3 object storage and the ability to view existing virtual machines (Fig. 5).



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    }
  ]
}
```

**Figure 5:** Example of an Identity-based policy, as code

Moreover, IAM allows the use of roles to perform specific tasks with cloud resources. Combining the use of roles, resource policies, EC2, and S3 services can organize uninterrupted access of a virtual machine to objects in storage. This practice is used in many solutions, including websites.

IAM also offers tools for controlling privilege and access grants—Permission Boundaries—which control the list of permissible privileges that can be granted to a user and a role. For example, a user who has full access to IAM but only read access to S3 cannot create an IAM user with privileges to write and edit the content of the S3 object storage, ensuring high-level protection even in the event of potential data compromise.

Without centralized control of software, services, and IT devices, the problem of "Shadow IT" [16] arises. The cloud platform allows controlling and centralizing all processes using IAM policies. Without sufficient permissions, using IT department-uncontrolled programs, services, and solutions becomes much more challenging. Proper IAM implementation prevents information leakage.

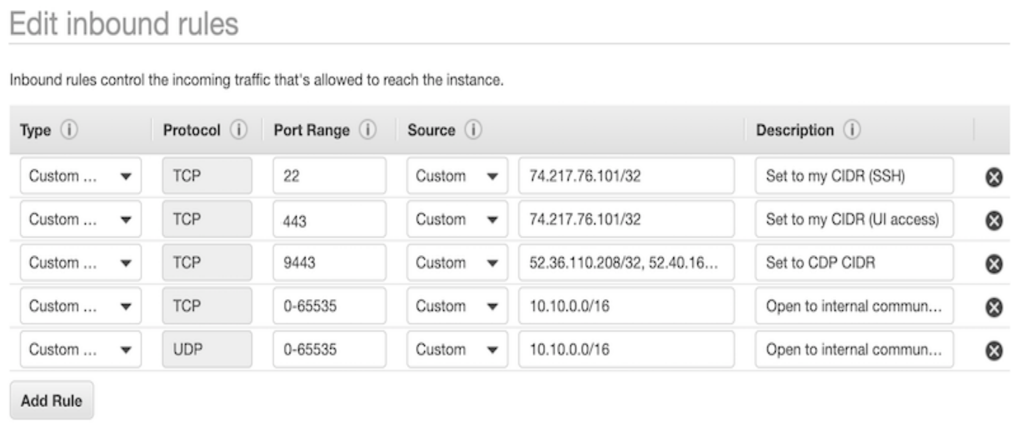
The implementation of the IAM service operation defines the main idea of the Security as Code methodology because it is through code that a clear and flexible process of privilege and access management to various cloud provider resources can be implemented.

## 7. Logical Infrastructure Isolation

Resource isolation from external interference is a current issue requiring special attention. In the AWS cloud environment, VPC is responsible for the logical isolation of the infrastructure. A VPC can be divided into isolated subnets from each other, being public (Public Subnets) and private (Private Subnets).

Traffic isolation is achieved using Security Group mechanisms and Network Access Control Lists, where security groups operate at the level of virtual machines/groups of virtual machines, and the network access control list at the subnet level (public and private).

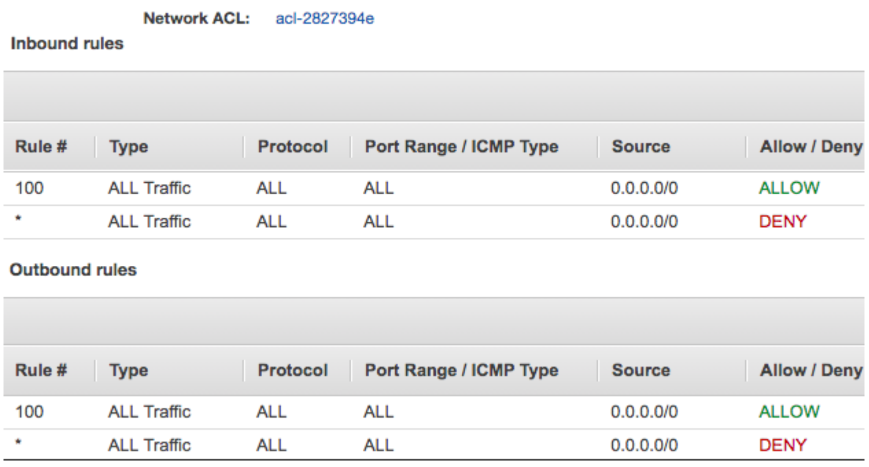
Moreover, network access control lists allow explicitly denying traffic—Explicit Deny, and security groups—implicitly—Implicit Deny. The difference in approaches to traffic denial is that in explicit denial, we must block traffic with rules (Fig. 6), and in implicit, the approach is "everything not allowed is forbidden" (Fig. 7).


Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

| Type | Protocol | Port Range | Source | | Description |
|---|---|---|---|---|---|
| Custom ... | TCP | 22 | Custom | 74.217.76.101/32 | Set to my CIDR (SSH) |
| Custom ... | TCP | 443 | Custom | 74.217.76.101/32 | Set to my CIDR (UI access) |
| Custom ... | TCP | 9443 | Custom | 52.36.110.208/32, 52.40.16... | Set to CDP CIDR |
| Custom ... | TCP | 0-65535 | Custom | 10.10.0.0/16 | Open to internal commun... |
| Custom ... | UDP | 0-65535 | Custom | 10.10.0.0/16 | Open to internal commun... |

**Figure 6:** Example of Security Group rules


Network ACL: acl-2827394e

Inbound rules

| Rule # | Type | Protocol | Port Range / ICMP Type | Source | Allow / Deny |
|---|---|---|---|---|---|
| 100 | ALL Traffic | ALL | ALL | 0.0.0.0/0 | ALLOW |
| * | ALL Traffic | ALL | ALL | 0.0.0.0/0 | DENY |

Outbound rules

| Rule # | Type | Protocol | Port Range / ICMP Type | Source | Allow / Deny |
|---|---|---|---|---|---|
| 100 | ALL Traffic | ALL | ALL | 0.0.0.0/0 | ALLOW |
| * | ALL Traffic | ALL | ALL | 0.0.0.0/0 | DENY |

**Figure 7:** Example of Network ACL rules

This allows isolating incoming traffic (from the Internet) to our resources, thus making them inaccessible from the outside (Fig. 8). The logical isolation of resources is a very good practice for ensuring a high level of infrastructure protection.

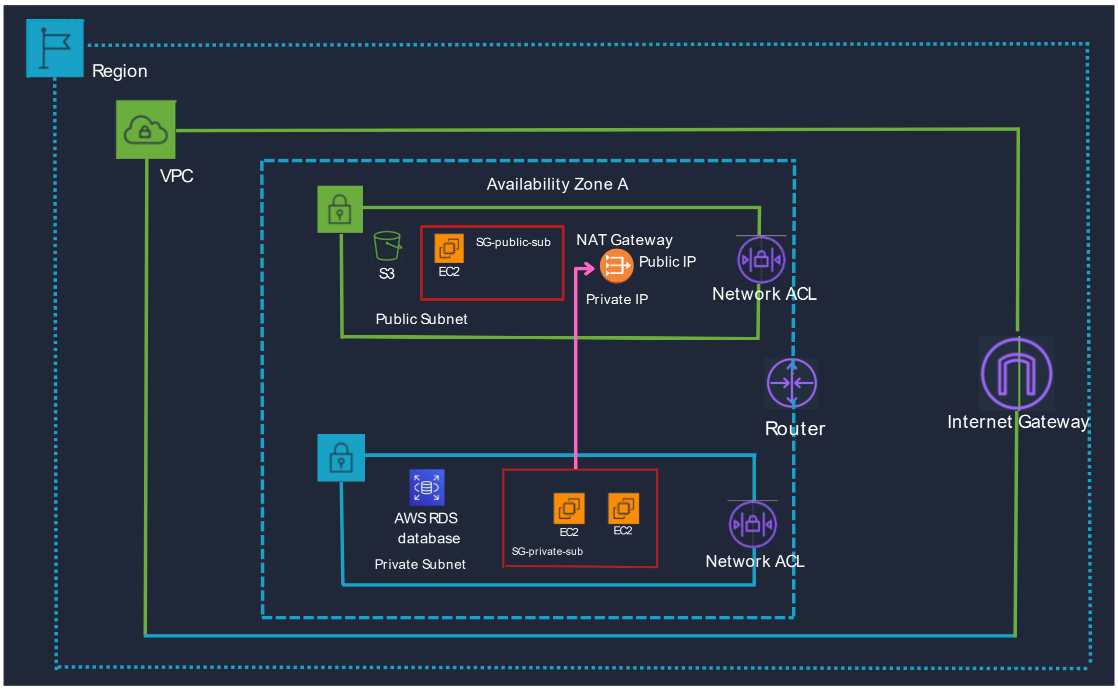


**Figure 8:** Logical isolation of resources in AWS

## 8. Logging and Event Monitoring

The security measures, which the authors conditionally attribute to the "Monitoring and Incident Response" principle, must continuously monitor the system for threats, anomalies, and security incidents and respond to them, i.e., address both monitoring issues and the automation of responses to anomalous or dangerous events.

Logging involves recording events occurring in the system such as authentication processes, requests for various actions with services, anomalous events, etc. This process should also be expanded with another process—recording log data in a file, which should be encrypted and stored in a secure place, such as S3 storage.

In turn, monitoring is a tool that continuously observes logs to detect anomalous or suspicious events, allowing for prompt response to potential threats and ensuring immediate reaction.

Summarizing the information above, the authors of the article concluded that logging and event monitoring are necessary components for ensuring an effective response to dangerous events, diagnosing problems, and understanding the performance of protection measures.

Among the tools offered by AWS, the functions of logging and event monitoring are performed by CloudWatch and CloudTrail services.

CloudWatch, as a comprehensive service, includes four functionally different components: CloudWatch Logs—a service for collecting and monitoring activities in the cloud and On-Premise environments; CloudWatch Metrics—numerical data about various resources and services (for example, CPU usage metrics); CloudWatch Events—a service for automating the management of rules to respond to certain real-time events; CloudWatch Alarms—a notification management service that can be configured to detect unusual or dangerous activity in metrics and trigger automatic actions. CloudWatch as a whole allows users to effectively track and manage resources in the cloud environment and respond to changes in real time.

CloudTrail is a service that automatically intercepts all actions with API endpoints and records them in S3 object storage. The integration of CloudTrail as a data source with the CloudWatch suite of services is a true implementation of automation for responding to predictable, template threats (Fig. 9).

## 9. Vulnerability Assessment of System Applications

Device posture assessment is designed to evaluate the threat a device poses to an organization and its systems. For instance, an assessment can confirm whether a device has the latest software and security updates installed, as well as if an endpoint security solution is in place and operational, which is crucial for the operation of critical infrastructure.

As a cloud provider, AWS offers a solution—Amazon Inspector—a service that can be used for network assessments and vulnerability assessments of computing resources, information leaks, and automation of regular checks. Amazon Inspector automatically identifies workloads such as Amazon EC2 instances, containers, and Lambda functions, and scans them for software vulnerabilities and unintended network exposures.

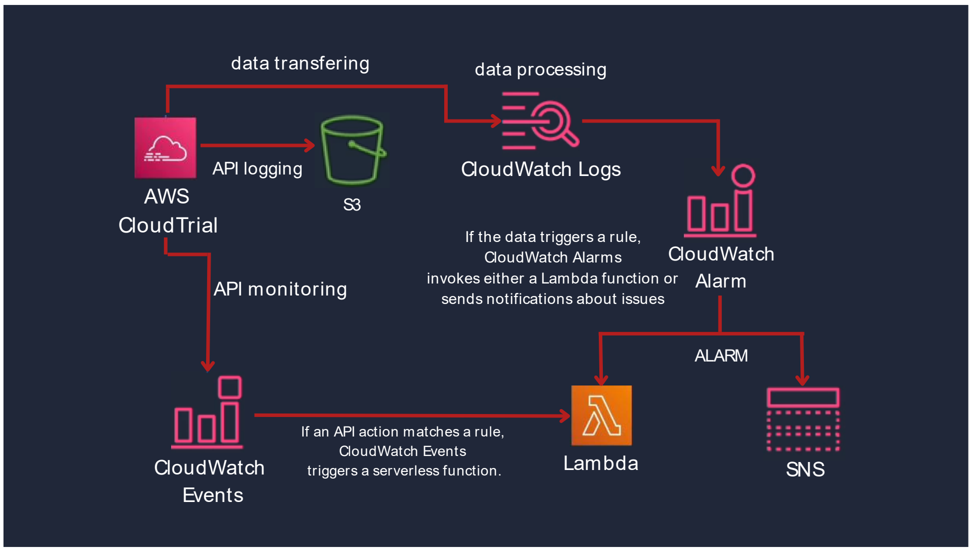

data transfering
data processing
API logging
AWS CloudTrail
S3
CloudWatch Logs
If the data triggers a rule, CloudWatch Alarms invokes either a Lambda function or sends notifications about issues
CloudWatch Alarm
API monitoring
ALARM
If an API action matches a rule, CloudWatch Events triggers a serverless function.
CloudWatch Events
Lambda
SNS

**Figure 9:** Automation scheme for Monitoring and Logging services

## 10. Confidential Information Verification

A leak of confidential information can be a critical factor for an enterprise and can disrupt established processes. Ensuring a secure place for processing and storing confidential information is one of the most important goals of any enterprise, startup, and especially an international company.

In the AWS environment, the authors selected Amazon Macie, a service that uses machine learning and pattern matching to monitor the publication of confidential data, provides visibility into data security risks, and allows for the automation of protection against these risks. Macie also identifies various types of data, such as PII (Personally Identifiable Information), PHI (Protected Health Information), regulatory documents, API keys, and secret keys. This ensures continuous monitoring and detection of confidential data that has been published either personally by the user or programmatically.

## 11. Data Encryption Tools

Based on the description of the encryption mechanism and its importance, it can be concluded that data encryption is a critically important mechanism for ensuring the confidentiality, integrity, and immutability of data, and thus an additional and critical level of security without which critical infrastructure would not be a complete system.

AWS KMS provides the ability to encrypt data in S3 object storage and databases, as well as encrypt traffic using TLS (Transport Layer Security). KMS supports external key stores for encryption keys (External Key Store) and custom key stores for encryption keys (Custom Key Store), offering greater flexibility in resource configuration.

The AWS CloudHSM service ensures a high level of security for encryption keys in the cloud environment, utilizing physical hardware security modules under our control.

It's worth mentioning the Amazon Certificate Manager service, which allows for easy issuance, management, and deployment of SSL/TLS certificates for web applications and AWS resources. ACM automatically generates and renews certificates within AWS, ensuring the security and encryption of HTTPS (Hypertext Transfer Protocol Secure) connections.

## 12. Filtering and Controlling Network Traffic

A firewall performs the functions of filtering and controlling network traffic to protect against unauthorized access and network attacks, including DDoS (Distributed Denial of Service) attacks, port scanning (for openness), packet interception and traffic analysis, SQL injections, cross-site scripting (XSS), information leakage, and more.

The cloud service provider AWS offers the WAF (Amazon Web Application Firewall) service, which is a comprehensive solution that includes the functionality of many types of firewalls, but its main focus is on protecting web applications from web attacks. WAF allows managing traffic based on web access control lists (Web ACLs), i.e., defining rules based on IP addresses, HTTP headers and bodies, and custom URIs. The primary purpose of WAF is to provide high protection against attacks such as SQL injections, cross-site scripting (XSS), information leakage, and others.

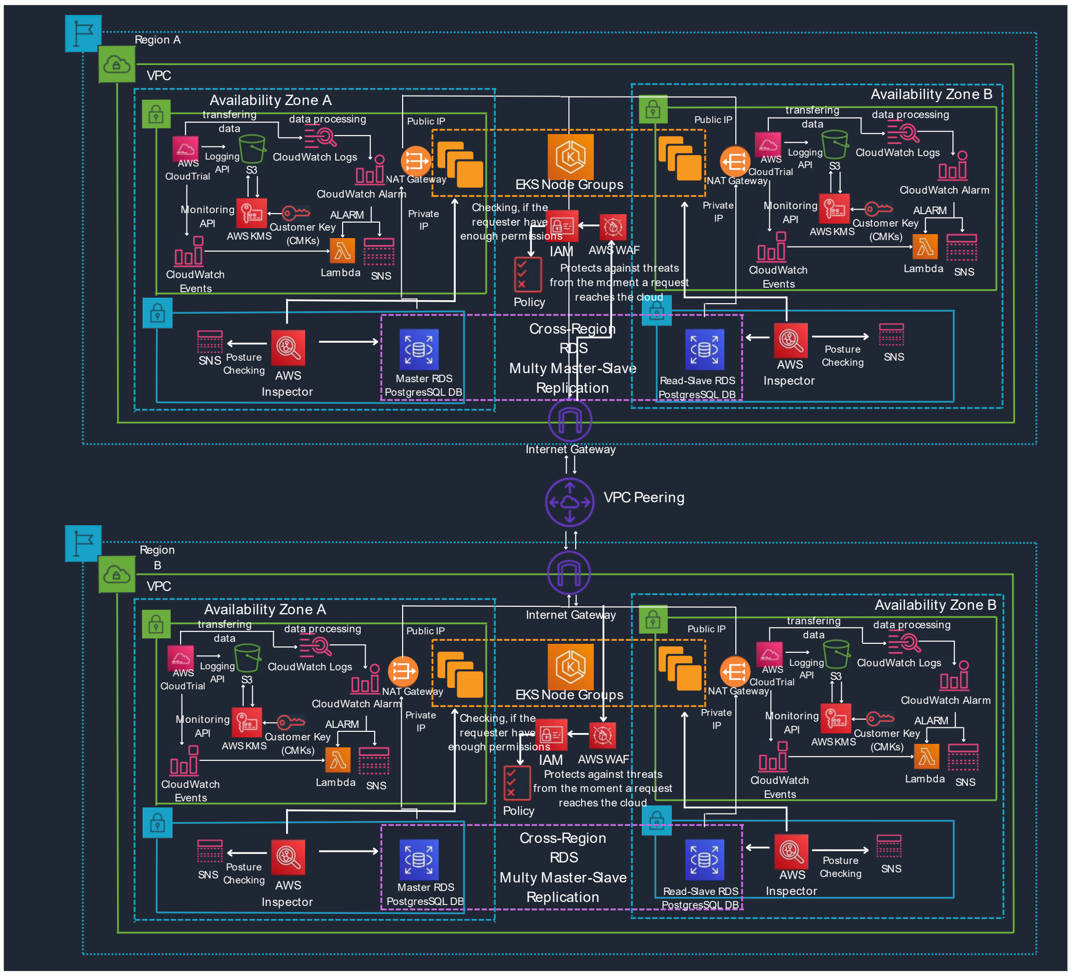


**Figure 10:** Multilevel Security System Built on the Principles of "Security as Code"

In Fig. 10, we can see a schematic representation of the constructed solution, i.e., a system for ensuring multilevel security based on the principles of "Security as Code", which includes aspects of the methodology, expands its functionality, and is also an extension of the Defense-in-Depth solution.

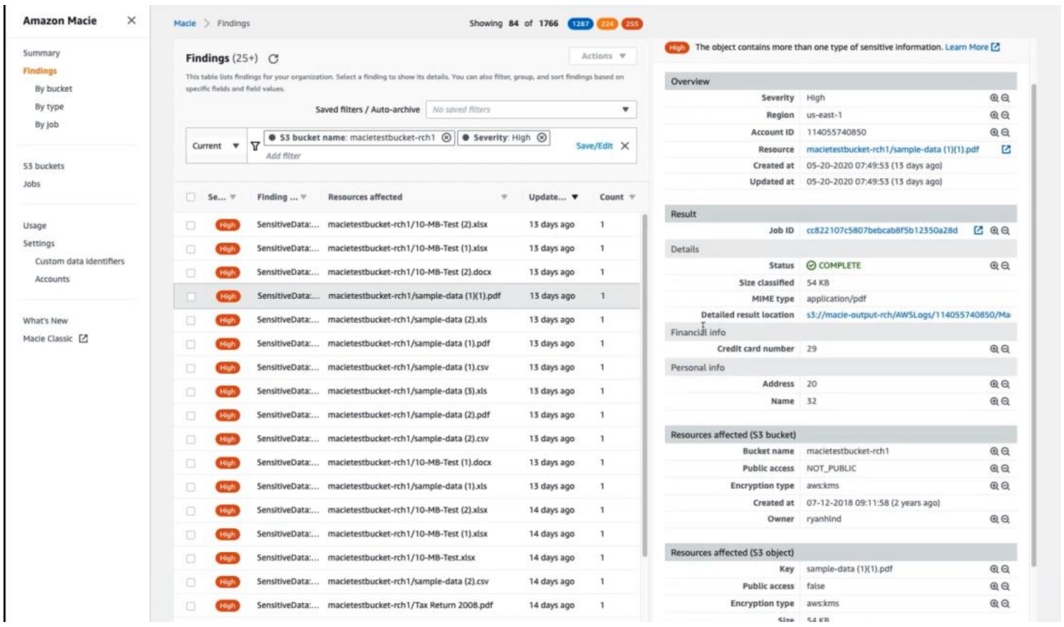
The developed solution fully implements security at multiple levels, preventing attacks such as DDoS (Distributed Denial of Service), port scanning (for openness), packet interception and traffic analysis, SQL injections, Cross-Site Scripting (XSS), information leakage, and more. It also addresses crucial issues: effective access and privilege management—IAM; logical isolation of network resources and their configuration —VPC, continuous monitoring, event logging, and immediate response to anomalies and dangers—a combination of AWS CloudTrail (API activity logging), CloudWatch (monitoring and logging, event response), Lambda (serverless function) and user notification mechanism—SNS; system application vulnerability checking (Posture Checking)—AWS Inspector; confidential information verification—Macie; data encryption tools—KMS; network traffic protection tools—Amazon Certificate Manager; network traffic filtering and control—AWS WAF.

## 13. Testing of the Constructed Solution

Testing of the constructed solution will be conducted using the following common checks and attacks: Posture Checking; data confidentiality verification; SQL injection; and Bot Stop.

Such tests will help understand the level of protection of the constructed solution and, in general, prepare for defense against known threats.

Here are the testing results:



Showing 84 of 1766 1287 224 255

Findings (25+)

This table lists findings for your organization. Select a finding to show its details. You can also filter, group, and sort findings based on specific fields and field values.

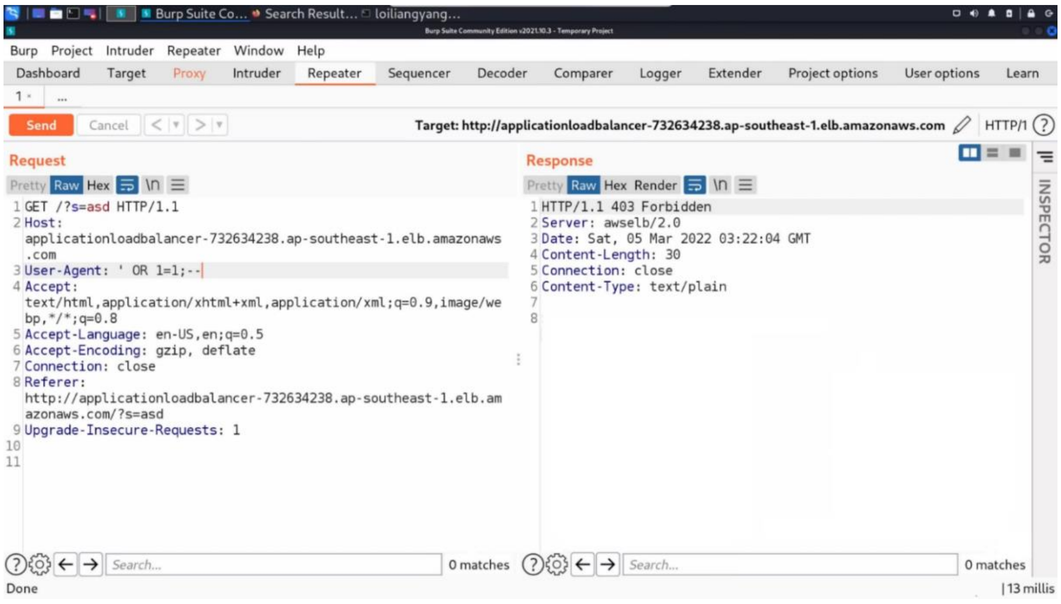
S3 bucket name: macietestbucket-rch1 | Severity: High

| Se... | Finding ... | Resources affected | Update... | Count |
|---|---|---|---|---|
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (2).xlsx | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (1).xlsx | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (2).docx | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (1)(1).pdf | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (2).xls | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (1).pdf | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (1).csv | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (3).xls | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (2).pdf | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (2).csv | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (1).docx | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (1).xls | 13 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (2).xlsx | 14 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test (1).xlsx | 14 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/10-MB-Test.xlsx | 14 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/sample-data (2).csv | 14 days ago | 1 |
| High | SensitiveData:... | macietestbucket-rch1/Tax Return 2008.pdf | 14 days ago | 1 |

High The object contains more than one type of sensitive information. Learn More

| Overview | |
|---|---|
| Severity | High |
| Region | us-east-1 |
| Account ID | 114055740850 |
| Resource | macietestbucket-rch1/sample-data (1)(1).pdf |
| Created at | 05-20-2020 07:49:53 (13 days ago) |
| Updated at | 05-20-2020 07:49:53 (13 days ago) |
| **Result** | |
| Job ID | cc822107c5807bebcab8f5b12350a28d |
| **Details** | |
| Status | COMPLETE |
| Size classified | 54 KB |
| MIME type | application/pdf |
| Detailed result location | s3://macie-output-rch/AWSLogs/114055740850/Ma |
| **Financial info** | |
| Credit card number | 29 |
| **Personal info** | |
| Address | 20 |
| Name | 32 |
| **Resources affected (S3 bucket)** | |
| Bucket name | macietestbucket-rch1 |
| Public access | NOT_PUBLIC |
| Encryption type | aws:kms |
| Created at | 07-12-2018 09:11:58 (2 years ago) |
| Owner | ryanhind |
| **Resources affected (S3 object)** | |
| Key | sample-data (1)(1).pdf |
| Public access | false |
| Encryption type | aws:kms |
| Size | 54 KB |

**Figure 11:** Detection of Potentially Compromised Information



Target: http://applicationloadbalancer-732634238.ap-southeast-1.elb.amazonaws.com HTTP/1

Request

```
GET /?s=asd HTTP/1.1
Host:
applicationloadbalancer-732634238.ap-southeast-1.elb.amazonaws
.com
User-Agent: ' OR 1=1;--
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/we
bp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer:
http://applicationloadbalancer-732634238.ap-southeast-1.elb.am
azonaws.com/?s=asd
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 403 Forbidden
Server: awselb/2.0
Date: Sat, 05 Mar 2022 03:22:04 GMT
Content-Length: 30
Connection: close
Content-Type: text/plain
```

**Figure 12:** SQL Injection Request Blocked by AWS WAF

The conducted testing proves the objectivity and appropriateness of the implemented solution for infrastructure protection.

## 14. Conclusions

The construction of infrastructure solutions in cloud platforms has garnered significant interest in the last five to ten years. Physical servers, in comparison to cloud solutions, are falling out of trend, becoming more expensive to acquire and maintain, have limited functionality, and are more complex to configure. With the mass migration of IT companies to the cloud, there arises a problem of complexity in the approach to developing cloud infrastructure and mechanisms for its protection.

The flexibility in configuring infrastructure solutions allows the use of new techniques for product deployment. The simplicity, templating, and modularity of IaC code enable rapid development of existing infrastructure solutions and the introduction of best practices. Control over the infrastructure lifecycle, from requesting specific virtual machine resources, network and subnet configurations, and DNS, to the automation of software product deployment, opens up vast opportunities for current business needs.

“Security as Code” represents the integration of security with the Infrastructure as Code approach. Correct implementation and choice of tools for security implementation are key aspects of this approach. Security as Code must ensure a high degree of product and infrastructure security, so this approach should be considered a multilayered comprehensive structure, where each layer of security is critically important not only at the level of its responsibility and functionality but also for all components of the system’s life cycle, whether the product or the infrastructure as a whole.

The authors of the article focused on the practical side of building a comprehensive solution in developing infrastructure and ensuring its security, illustrated by an enterprise example. The work considered many different aspects regarding the choice of the right platform, architecture design, and security provision, and, as a result, built and implemented a practical solution using the “Infrastructure as Code” and “Security as Code” approaches.

## References

[1] J. Wu, et al., Cloud Storage as the Infrastructure of Cloud Computing, International Conference on Intelligent Computing and Cognitive Informatics (2010) 380–383. doi: 10.1109/ICICCI. 2010.119.

[2] M. TajDini, V. Sokolov, V. Buriachok, Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Define Radio, in: 8th International Conference on “Mathematics. Information Technologies. Education:” Modern Machine Learning Technologies and Data Science, vol. 2386 (2019) 287–296.

[3] M. TajDini, V. Sokolov, P. Skladannyi, Performing Sniffing and Spoofing Attack Against ADS-B and Mode S using Software Define Radio, in: IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (2021) 7–11. doi: 10.1109/UkrMiCo52950.2021.9716665.

[4] V. Sokolov, P. Skladannyi, N. Korshun, ZigBee Network Resistance to Jamming Attacks, in: IEEE 6th International Conference on Information and Telecommunication Technologies and Radio Electronics (2023) 161–165. doi: 10.1109/UkrMiCo61577.2023.10380360.

[5] V Sokolov, P. Skladannyi, V. Astapenya, Bluetooth Low-Energy Beacon Resistance to Jamming Attack, in: IEEE 13th International Conference on Electronics and Information Technologies (2023) 270–274. doi: 10.1109/ ELIT61488.2023.10310815.

[6] V. Sokolov, P. Skladannyi, A. Platonenko, Jump-Stay Jamming Attack on Wi-Fi Systems, in: IEEE 18th International Conference on Computer Science and Information Technologies (2023) 1–5. doi: 10.1109/CSIT61576.2023.10324031.

[7] Amazon, What is ETL (Extract Transform Load)? URL: https://aws.amazon.com /what-is/etl/#:~:text=Extract%2C%20 transform%2C%20and%20load%20(,a nd%20machine%20learning%20(ML)

[8] S. Shylesh, A Study of Software Development Life Cycle Process Models,

SSRN (2017). doi: 10.2139/SSRN. 2988291.

[9] R. Longbottom, Cray 1 Supercomputer Perfomance Comparisons with Home Computers, Phone and Tables. URL: http://www.roylongbottom.org.uk/Cray%201%20Supercomputer%20Performance%20Comparisons%20With%20Home%20Computers%20Phones%20and%20Tablets.htm

[10] AWS Web Services Inc. or its affiliates, Shared Responsibility Model. URL: https://aws.amazon.com/compliance/shared-responsibility-model/

[11] Recro, Top 5 Cloud Service providers: A Comparison. URL: https://recro.io/blog/top-5-cloud-service-providers/

[12] A. Hiles, Business Continuity: Best Practices: World-class Business Continuity Management, 2nd Edition, (2004).

[13] G. Raje, Security and Microservice Architecture on AWS, O'Relly Media, Inc, (2021).

[14] M. Holmgren, Multi-Master Database Replication and e-Learning, KTH School of Information and Communication Technology (ICT) (2015).

[15] P. Kirvan, Techtarget, RPO vs. RTO: Key Differences Explained with Examples, Tips (2021).

[16] J. Chijioke-Uche, Infrastructure as Code Strategies and Benefits in Cloud Computing, Walden University ProQuest Dissertations Publishing (2022).

[17] O. Vakhula, I. Opirskyy, O. Mykhaylova, Research on Security Challenges in Cloud Environments and Solutions based on the "Security-as-Code" Approach, in: Cybersecurity Providing in Information and Telecommunication Systems II Vol. 3550 (2023) pp. 55–69.

[18] A. Pessa, Comparative Study of Infrastructure as Code Tools for Amazon Web Services, M. Sc. Thesis, Faculty of Information Technology and Communication Sciences (2023). URL: https://trepo.tuni.fi/bitstream/handle/10024/149567/PessaAntti.pdf?sequence=2

[19] HashiCorp, AWS Provider Documentation. URL: https://registry.terraform.io/providers/hashicorp/aws/latest

[20] BK Sarthak Das, Virginia Chu, Security as Code, O`Reilly Media, Inc. Book (2023).

[21] M. Silic, A. Back, Shadow IT—A View from Behind the Curtain, Comp. Secur. 45 (2014) 274–283. doi: 10.1016/j.cose. 2014.06.007.