Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Cloud-Native Architectures in Financial Services: Enhancing Scalability and Security with AWS and Kubernetes

Ashmitha Nagraj
Senior Full Stack Engineer
Corresponding Author: Ashmitha Nagraj, **E-mail**: nagrajashmitha@gmail.com

| ABSTRACT

The financial industry's service delivery model is constrained by unique requirements; volatile traffic, high fault-tolerance engineering needs, and regulatory compliance require security controls to demonstrate their effectiveness. The purpose of this research is to explore whether cloud native architecture based on AWS primitives and Kubernetes can provide scalable and secure solutions for the financial sector, including compliance for financial sector operational resiliency and regulatory compliance standards (GDPR security obligations, PCI DSS, SOC 2 and other related security obligation). To achieve this objective, authoritative documentation from the Cloud Native Computing Foundation (cloud native computing foundation) and the official documentation for Kubernetes and AWS, along with relevant security documents published by NIST (containerization, micro-services, service mesh, zero trust architecture) and the financial sector's guidelines for managing risk were synthesized. A compliance aware reference architecture was proposed, which defines and enforces system boundaries (network, identity, workload, and data) that employs declarative automation to tie runtime telemetry to audit evidence. Scalability is viewed as a closed-loop feedback mechanism, enabling consideration of load balancing, platform autoscaling, and Kubernetes control loops that include security externalities such as an EDoS attack against autoscaling. The security aspects of identity, least privilege, encryption/key management, segmentation, vulnerability management, tamper evident logging, and incident response preparedness were also examined. The case studies cited within the document represent regulated or finance adjacent production deployment examples that were utilized to support the analysis with measurable outcomes and implementation realities.

| KEYWORDS

Cloud-Native Architectures; Financial Services; Scalability and Security; AWS and Kubernetes

| ARTICLE INFORMATION

ACCEPTED: 01 November 2023 **PUBLISHED:** 20 November 2023 **DOI:** 10.32996/jcsts.2023.5.4.30

Introduction

A major challenge in implementing FS Platforms is the convergence of three factors (1) volatile demand, (2) the adversarial nature of the marketplace, and (3) compliance-based requirements for evidence of intent vs. evidence of action- into a single product offering. As such, payment rails, retail banking, risk engines, fraud detection, and market data analysis are impacted by bursty, correlated load (i.e., campaign-related traffic, market events, holidays, and batch settlement cycles), which tend to occur concurrently with increased fraud activity and operational risk. For example, during the 2022 Black Friday/Cyber Monday event, a global payment platform processed extreme transaction sales at peak demand, demonstrating how quickly modern financial systems must react.

In addition to addressing the need for better scalability, the concentration of clouds and digital transformation have also created regulatory expectations regarding third party risk, resilience, and incident response readiness. The FFIEC's joint statement on cloud security explicitly states that cloud computing creates shared responsibility and requires organizations in the financial sector to develop robust risk management practices related to governance, oversight and operational controls. [2]

The primary hypothesis of this paper is that compliance-aware systems based upon cloud-native architecture can address both scaling and security challenges concurrently. The term "cloud-native" is sometimes misused, however here it refers to the Cloud Native Computing Foundation's definition of cloud-native: cloud-native technologies allow developers to create scalable applications in dynamic environments using containers, service meshes, microservices, immutable infrastructure and declarative APIs, etc. [3]. Kubernetes is one of the most widely accepted open-source frameworks for deploying these types of architectures: it is an open-source platform for automating the deployment, scaling, and management of containerized application workloads through declarative configuration. [4].

AWS was selected as the subject of this paper due to the clarity of its operational model regarding the boundaries of customer and provider responsibilities. The AWS shared responsibility model clearly identifies security and compliance as areas where both the customer and AWS are responsible, with AWS managing the underlying infrastructure layers and the customer being responsible for securing their own configuration and usage of AWS services. [5]. These boundaries matter from a regulatory perspective because financial compliance regulations require the ability to articulate and defend the ownership of individual system components.

Problem statement

There are two types of failure that have been prevalent in recent major failures in finance. First there are legacy scalability issues: Systems were built with static capacity and slow provisioning; they are unable to scale during a burst in demand. Legacy systems are fixed with permanent over-provisioning (which is expensive and does not provide long term reliability). Second, Configuration Driven Security Issues: Cloud Control Plans are becoming increasingly complex, with multiple layers of complexity in their architecture, creating new surfaces for misconfigurations, and if those configurations allow incorrect access, it can result in a massive compromise of an organizations resources regardless of whether a specific vulnerability has been exploited. The peer reviewed study on the 2019 Capital One breach identified the lessons learned of how misconfiguring cloud services along with request routing/SSRF like exploitation paths directly impacted the breach and clearly demonstrated that the boundaries of a system, both architecturally and through correct configuration will determine the level of security in a cloud environment and not simply the use of a hyper scale provider. [6]

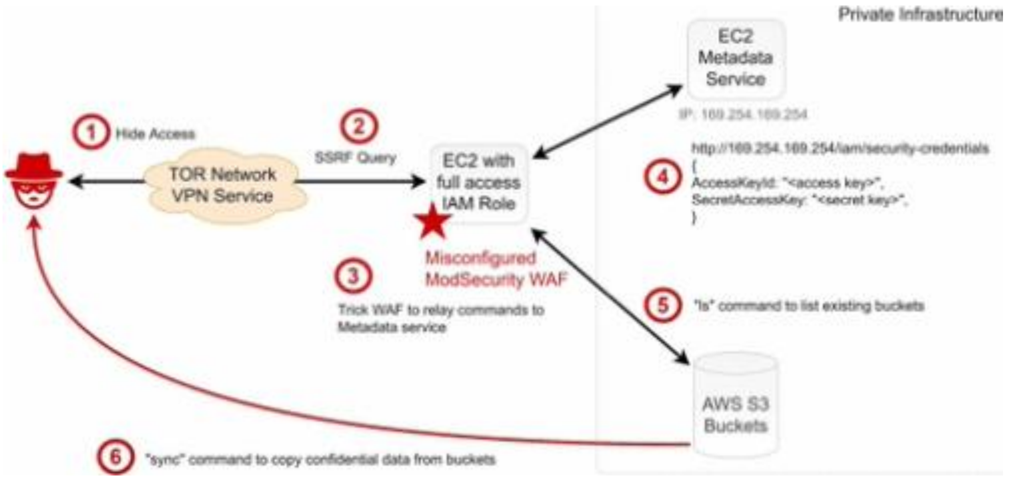

Private Infrastructure
EC2 Metadata Service
IP: 169.254.169.254
1 Hide Access
2 SSRF Query
TOR Network VPN Service
EC2 with full access IAM Role
http://169.254.169.254/iam/security-credentials
4 { AccessKeyId: "<access key>", SecretAccessKey: "<secret key>", }
Misconfigured ModSecurity WAF
3 Trick WAF to relay commands to Metadata service
5 "ls" command to list existing buckets
AWS S3 Buckets
6 "sync" command to copy confidential data from buckets

Figure 1: Capital One cyberattack [6]

Objectives, scope, and contributions

The goal of this paper isn't to say "cloud-native automatically means it's secure" or "Kubernetes will automatically ensure your application is available." This paper is about how AWS and Kubernetes can be used together as an operational model for a scalable, secure and compliant cloud architecture in practice. It provides an approach to:

- **Scalability**: How to scale the infrastructure under burst loads with multi-layered elasticity using load balancers, auto-scaling, and the orchestrator control loop. [7]
- **Security**: How to reduce security risks through Identity management, encryption, segmentation, and logging to support compliance auditing.
- **Compliance**: How to provide continuous compliance by creating audit trail evidence from telemetry data and declarative policies instead of relying on post-hoc reports. [8]

This paper introduces an integration framework based upon synthesis: A structured mapping from financial compliance objectives (e.g., PCI-DSS segmentation and scoping; SOC 2 Trust Services Criteria categories; GDPR's security-of-processing obligations; Financial Sector Resilience Expectations) to specific AWS / Kubernetes Controls and measurable evidence outputs. [9]

Cloud-Native Foundations, Related Work, and Method

Architectural principles relevant to regulated finance

Cloud native is essentially a collection of disciplined constraints; namely: immutable configurations, API driven operations and declarative system definitions. The restrictions imposed by these constraints allow for automation to occur within an environment however they also allow for auditability to occur. To support financial governance objectives, declarative systems produce artifacts that can be versioned, reviewed, scanned and tested as code.

While cloud native does introduce many additional components and additional areas of trust boundaries NIST provides both container and microservice security guidelines. Specifically, the container security guidelines provided by NIST state that container adoption introduces security risks related to all containers/images, registries/orchestration platforms/host operating systems and recommends container aware policies, vulnerability management and runtime hardening. [10] Additionally, NIST's microservice security guidelines state that while microservice architectures create new threats between services, security strategies are required for API gateways, service meshes and other supporting infrastructures. [11]

Similarly, financial systems have a strong alignment with the principles of Zero Trust Architecture (ZTA): moving defensive mechanisms from static perimeter-based to identity based on assets and resources. NIST SP 800-207 provides the definition of ZTA and outlines a strategy for how enterprises may begin transitioning to this type of architecture, particularly in environments where the "perimeter" has become a distributed mesh of services. [12]

Compliance requirements as architectural drivers

The compliance environment can vary in terms of geography and business model however there are several compliance requirements that are common across different geographies and business models:

- PCI DSS version 4.0.1 outlines the specific requirements and testing processes to protect CDE's as well as outlines transition periods for specific requirements with respect to certain date ranges. [13]
- SOC 2 reports assess the controls applicable to security, availability, processing integrity, confidentiality, and privacy, based upon the AICPA Trust Services Criteria. [14]
- GDPR Article 32 outlines the measures to be taken to provide an appropriate level of protection for the security of processing, which shall include (if necessary) encryption, resilience, restoration capabilities, and regular testing and evaluation of the effectiveness of those measures. [15]
- The US GLBA Safeguards Rule (16 CFR Part 314) establishes guidelines for the administrative, technical, and physical safeguards that must be in place to protect customer information. [16]
- NYDFS 23 NYCRR 500 provides significant requirements related to multi-factor authentication (MFA) for covered entities and has revised language to require MFA for individuals who access information systems (except for exceptions and compensating controls). [17]
- EU DORA provides uniform requirements for ICT risk management, incident reporting, operational resilience testing, and ICT third party risk management for all financial institutions. [18]

An important architectural consequence is that compliance is not simply completing paperwork; it is also a set of constraints regarding how identity flows occur, how segmentation occurs, how encryption and key custody occur, how ready one is for incidents to occur, and how evidence is retained.

Method and evaluation approach

The methodology used in this study is a mix-methodology, based on two different methodologies: (1) Synthesis of Primary Sources (CNCF / Kubernetes / AWS Documentation, NIST Publications, Regulatory Documents) and (2) Evidence Triangulation using Case Studies of Finance Relevant Success Stories from AWS and Kubernetes Published Success Stories. [19]

The Empirical Framework for this study will be framed using a Metric Vocabulary that is operationally relevant to practice, as follows:

- **Scalability**: Throughput Capacity, Latency Distribution (P50/P95/P99), Time it takes to Scale Response, Saturation Behavior of Burst Loads.
- **Reliability**: Availability SLO Attainment, Time to Restore Service, Blast Radius of Failures (Service-Level vs Region-Level Impacts).
- **Security**: Coverage by Least Privilege Access, Vulnerability Exposure Over Time, Configuration Drift, Audit Log Completeness.
- **Compliance Evidence**: Ability to Produce Clear Artifacts Linking Controls to Implemented Configurations (Policies, Logs, Scan Outputs, Change History).

Although Controlled Experiments are outside the scope of this Narrative Paper, the Evaluation Stance of this study is Intentionally Engineering-Oriented: Every Recommended Mechanism has a Measurable Outcome, and has a Source Backed Control Objective.

Scalability Engineering with AWS and Kubernetes

Scalability in financial services is not “scale up eventually.” It is about absorbing bursts without crossing risk thresholds: latency SLO violations can translate directly into financial losses, customer harm, or risk model degradation. Cloud-native scalability is best viewed as a multi-layer control system.

Layered elasticity: load balancing, fleet scaling, and control loops

Elastic Load Balancing was created to distribute incoming traffic at the network edge and service entry layer and scale load balancer capacity based on traffic volume [20]. Elastic Load Balancing is an application load balancer, it includes support for request-based routing of requests to targets such as containerized applications or serverless functions. Therefore, the ELB allows you to create routing patterns to target microservices. [21]

At the compute layer, AWS Auto Scaling and EC2 Auto Scaling allow users to define policies and perform health checks and then scale out or in their instance fleets to maintain enough capacity to meet current load conditions. [22]

Kubernetes has two control loops for workload replication and for cluster capacity:

- The Horizontal Pod Autoscaler (HPA) will automatically update a workload resource, such as a deployment or statefulset to scale the number of pods to meet current demand. [23]
- When pods cannot be scheduled onto nodes, Node/Cluster scaling will add capacity by adding nodes to match scheduling needs across node groups. [24]

Amazon EKS also provides custom scaling options for AWS-hosted Kubernetes clusters. AWS Documentation states that Karpenter is a high-performance cluster autoscaler that launches right-sized compute in under one minute to meet pod scheduling demands for a "just-in-time" provisioning capability. [23]

Serverless scaling and predictable burst envelopes

Financial architectures frequently combine Kubernetes-hosted microservices with event-driven serverless components for asynchronous workflows (notifications, streaming transformations, risk scoring bursts, and reconciliation triggers). AWS documents a specific Lambda concurrency scaling rate per function per region (1,000 additional execution environment instances every 10 seconds, or 10,000 requests per second every 10 seconds), which provides a quantifiable scaling envelope for design and quota planning.[25]

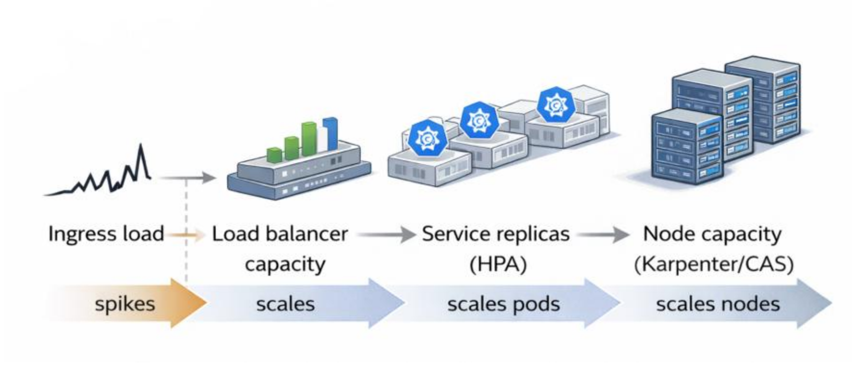


Figure 2. Illustrative scaling envelopes across layers (conceptual)

This layered model implies a practical design pattern: keep the fastest-scaling layers closest to the spike source (request routing and replica scaling), and ensure slower layers (node provisioning, database scaling) are isolated through buffering (queues/streams) and backpressure.

Evidence from finance-relevant production platforms

Case studies provide a proof-of-concept or validation that elasticity solutions can be successfully implemented with real-world constraints, they do not represent a standard benchmark.

For example: Nasdaq reports that it now processes 70 billion records per day and has seen measurable results in its use of AWS-based analytics, including reducing time-to-load market data by five hours and improving Redshift query performance by 32%. In these cases, scalability is shown to have a direct impact on business and not just API-related applications. [26]

In Capital One's Kubernetes case study, the focus is on resiliency and speed for a provisioning platform that supports streaming, big-data decisioning, and machine learning, and they report that one of the applications that utilized container orchestration was able to handle tens of millions of transactions per day and directly tie container orchestration to fraud detection and credit decisioning workload functions. [27]

Scaling is also a security problem: EDoS and cost-safety

The double-edge of autoscaling for the Financial Services industry is that it minimizes the chance of capacity collapse, but it has the potential to expand the economic threat from an attacker who could manipulate the scaling behavior. Peer reviewed research into EDoS against Kubernetes autoscaling presents models that evaluate the scaling thresholds and the potential economic damage to illustrate how autoscaling systems can be manipulated to cause excessive use of consumed resources [28]. This will interact directly with DDoS defenses and cost protection. AWS WAF was created to help prevent common web attacks and bots from degrading availability or consuming resources, and AWS Shield Advanced provides DDoS cost protection features to cap scaling charges related to DDoS usage spikes to protected resources. [29]

Original Insight: Scaling-Safe architecture for financial services, elasticity should be viewed as a constrained optimization problem, not simply throughput maximization. Implement rate limits and bot controls at the edge (WAF, API Gateways), to reduce manipulation of scaling behavior [29]. Instead of using a Single Resource Metric, use Multi-Signal Metrics to couple autoscaling (Queue Depth + Authenticated Request Rate + Error Budgets) and implement explicit "Cost Guardrails so that human visibility is provided before scaling escalates costs. The first and third items are directly grounded in AWS Control Capabilities. The second item is an engineering recommendation motivated by typical SRE Practice and EDoS Research.

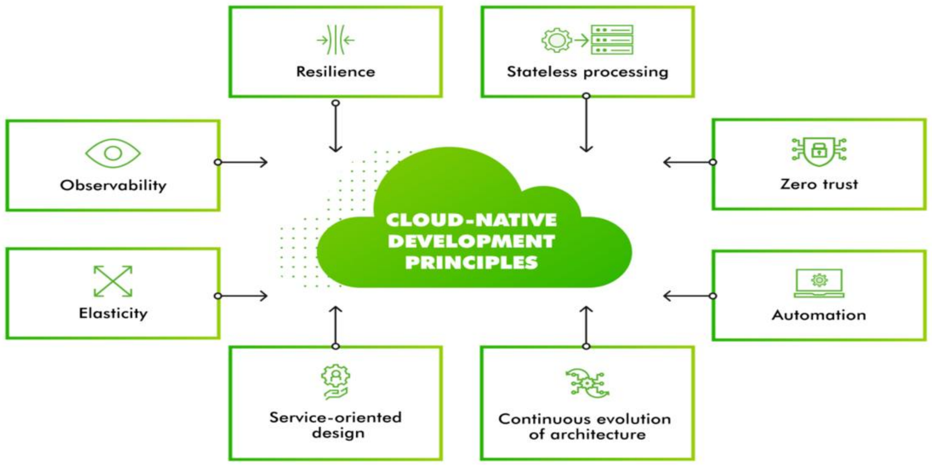


Figure 3: Cloud-Native Development Principles

Security and Compliance by Construction

While security in financial services is about preventing breaches, it's also about proving that controls exist, function properly and are monitored. Cloud-native architectures can help with this, but only if security is implemented across identity, network, workload, data, and operations.

Shared Responsibility and Control Ownership

AWS's shared responsibility model provides an auditable narrative around responsibilities: AWS manages physical facilities, hardware, and virtualization layers, while customers manage configurations, access controls and application layer security. This boundary defines which controls can be inherited from AWS and those that must be implemented as customer controls. [6]

Compliance Perspective

AWS artifact supports this by providing On-Demand Access to all available AWS Security and Compliance Documents, including SOC Reports and PCI Related Artifacts, enabling customers to present evidence of inherited-control in a defensible way. [30]

Identity- Least Privilege for Humans and Workloads: Least Privilege is one simple concept, yet the hardest to implement consistently. AWS IAM guidance defines least privilege as granting only the permissions required for a task and no additional permissions, and AWS IAM best practices emphasize temporary credentials, MFA, and regular review of permissions and access [31]. Kubernetes uses Role-Based Access Control (RBAC) to regulate workload access to resources based on roles, and the Kubernetes documentation presents RBAC as a key security control to ensure users and workloads have only the necessary access [32]. A Platform-Level pattern that bridges these layers is Workload-Specific Identity. For Example, with EKS, IAM Roles for Service Accounts (IRSA) allows associating AWS IAM permissions with Kubernetes Service Accounts, therefore Fine-Grained credentials are obtained without Node-Wide permissions inheriting by the pod. [33]

Financial Relevance: This Mapping enables segregation of duties and reduces Blast Radius in compromise scenarios. It also enhances Auditability through creating clearer "Who/What Accessed Which AWS API" trails when combined with CloudTrail logging.

Cryptography and key custody

Encryption is an organizational compliance requirement, and the real architectural challenge is how to manage and audit the keys. AWS KMS is intended to allow for the creation, use and control of all the keys necessary to encrypt or decrypt data and to sign data, and in its documentation, AWS has provided examples of how to use "envelope encryption," which involves encrypting data using a "data" key, and then encrypting the "data" key with a KMS key.

The PCI Data Security Standard v4.0.1 includes specific requirements and guidelines to protect the security of cryptographic keys as well as limit the number of entities that have access to the keys, further indicating that managing cryptographic keys is not simply a matter of "turning on encryption", but rather includes managing the physical and logical location of keys, access rights to those keys, and monitoring the access to the keys. [34]

Original Insight: Cryptography is a boundary for auditing. As part of designing financial architecture, treat the use of different cryptographic methods to secure the data as explicitly defined variables within the design. The use of such variables will make the scope of a PCI audit easier to determine and create clear evidence narratives on how a CDE was determined: not based on the physical or logical separation of a network, but by the cryptographic method used to encrypt the data.

Segmentation: network boundaries plus Kubernetes policies

PCI DSS scope guidance has historically emphasized segmentation as the reason one can create a compliant scope, rather than having no bounds for the audit scope. AWS provides guidance on PCI scoping and segmentation for PCI DSS 4.0 workloads on AWS, which include methods of defining in-scope and out-of-scope resources and documenting segmentation boundaries. [35]

In Kubernetes, Network Policies provide a method for defining traffic flow rules at layer 3/4 between Pods and from Pods to external sources; however, enforcing these policies requires a network plugin. AWS recommends using a multi-layered approach when creating Network Policies with Kubernetes, utilizing Security Groups for Pods (SGP) and Network Policies to provide least-privilege segmentation and separation of pods and namespace [36]. This presents a significant leverage point for compliance. A defense-in-depth segmentation model may be built using AWS network controls to limit traffic flowing into the VPC, with Kubernetes policy used to limit traffic moving laterally within the cluster.

Workload hardening: Pod Security Standards and admission control

Kubernetes has Pod Security Standards that can range from a permissive model of policy to a completely restrictive model of policy; Kubernetes also provides Pod Security Admission to apply those same standards at admission time [37]. The NIST guidelines for container security recommend implementing restrictions on privileges, reducing an attack's potential target (attack surface), and applying security policies that are aware of containers across all layers of the runtime and orchestration environments [10].

A realistic minimum standard for financial systems would include the following: enforcing default non-privileged container settings, restricting hostPath mount permissions, preventing privileged escalations, and blocking all images that have failed to pass through any previously defined provenance or vulnerability testing gates. The exact level of restriction will depend upon the specific workload being processed; however, the method in which the restrictions are enforced should be standardized across the entire platform rather than each team having its own method.

Logging, monitoring, and audit evidence

Audit-grade logging is where cloud-native platforms can outperform legacy systems, because every control plane action can be logged by design.

- AWS CloudTrail records actions taken by users, roles, or AWS services and is explicitly positioned as enabling governance, compliance, and auditing by recording events across AWS interfaces (console, CLI, SDK).[38]
- Kubernetes auditing provides a chronological record of actions in a cluster, including user actions, application API usage, and control plane activity. [39]
- Logs are directly related to compliance obligations:
    - Change management and enforcement of access controls and incident response timelines (SOC 2 evidence). [14]
    - Timely restoration of systems and routine testing and audit logs and incident response documentation provide evidence of operations (as required by GDPR Art 32). [40]

Compliance automation tools**:** AWS Config will review, audit, and evaluate AWS resource configurations and alert when there are configurations that do not meet compliance requirements in addition to providing historical configurations for auditing [41]. Additionally, AWS has published machine readable compliance bundles indicating an intent to place compliance documents within automated workflows instead of viewing them as static PDFs.

Incident response and resilience as security requirements

As many compliance programs are now treating resilience as an integral part of their overall approach to security, AWS is emphasizing the importance of being prepared for potential threats and having the capability to isolate systems, contain damage, perform forensic analysis, and restore operations back to a "known good" state. AWS has developed a technical guide for incident response that is specifically designed for use in AWS-based environments. The guide provides a structure for organizations to respond to incidents by utilizing cloud-based technologies. [42] DORA includes both incident reporting and operational resilience testing as components of a single, unified compliance program for all financial entities, which means that incident response preparedness will be viewed as a regulatory requirement for all entities, as opposed to simply an internal best practice. [43]

Reference Architecture and Case Evidence

A compliance-aware reference architecture

Below is a reference architecture designed to satisfy the *mechanics* of scalability and the *provability* of security controls.

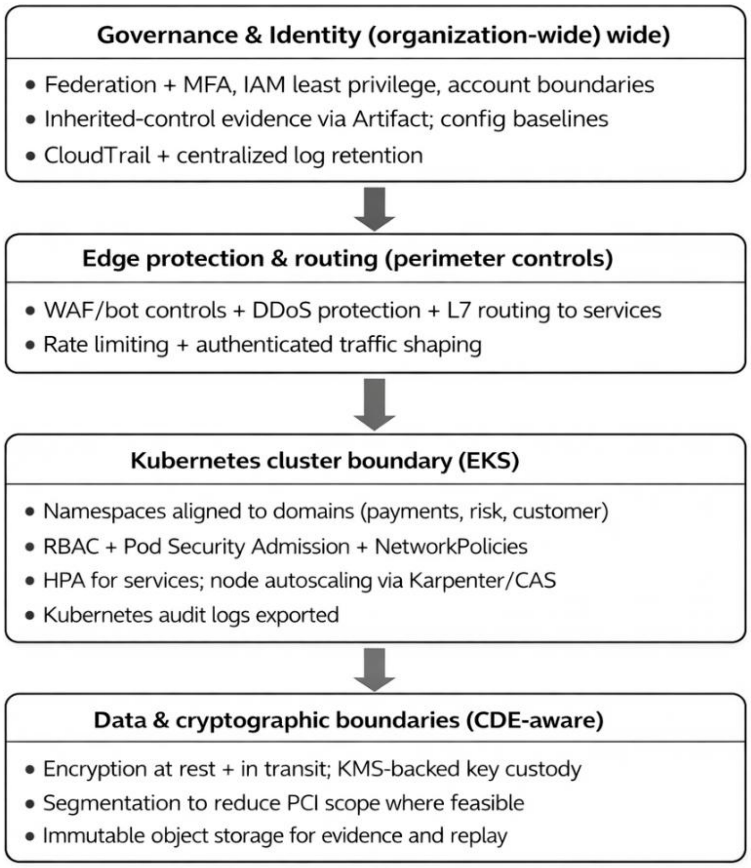

Governance & Identity (organization-wide) wide)
• Federation + MFA, IAM least privilege, account boundaries
• Inherited-control evidence via Artifact; config baselines
• CloudTrail + centralized log retention
Edge protection & routing (perimeter controls)
• WAF/bot controls + DDoS protection + L7 routing to services
• Rate limiting + authenticated traffic shaping
Kubernetes cluster boundary (EKS)
• Namespaces aligned to domains (payments, risk, customer)
• RBAC + Pod Security Admission + NetworkPolicies
• HPA for services; node autoscaling via Karpenter/CAS
• Kubernetes audit logs exported
Data & cryptographic boundaries (CDE-aware)
• Encryption at rest + in transit; KMS-backed key custody
• Segmentation to reduce PCI scope where feasible
• Immutable object storage for evidence and replay

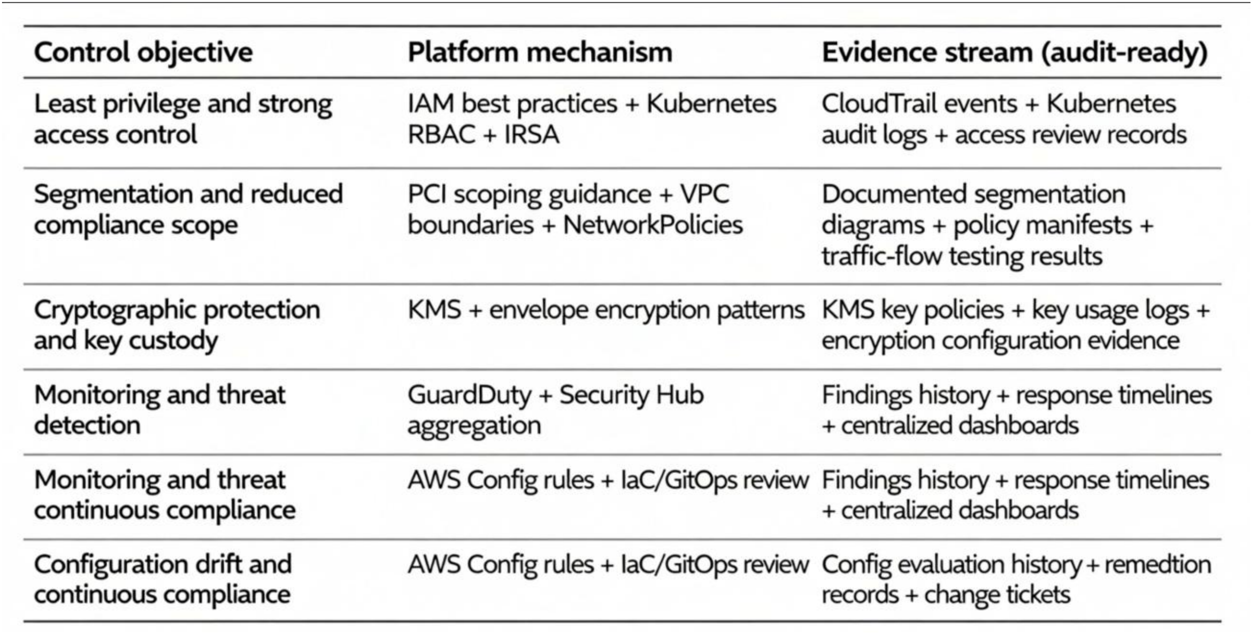
Figure 4. Reference architecture for regulated financial workloads (AWS + Kubernetes)

This design mirrors documented functionality:

- Kubernetes allows for declarative workload and scaling automation.
- The EKS control plane, and the overall architecture of the EKS cluster are both built around multiple Availability Zones (AZ) resiliency. AWS has documented at least two API servers running in different AZ and the ETCD clusters have been deployed in three AZ. [44]
- Karpenter creates just in time compute node deployments to address Pod scheduling issues; Karpenter is also able to deploy compute resources in under one minute. [26]
- Edge defense will be enhanced by utilizing AWS WAF and DDoS protection; WAF has the explicit purpose of helping to defend against bot traffic and common web-based exploits that can impact availability or utilize an organizations resource pool. [45]
- AWS Artifact provides customers with the ability to download compliance documents and reports, which is necessary for demonstrating inherited control.

Control mapping: from requirements to technical evidence

A common failure mode in PCI DSS and SOC 2 programs is that controls exist but are not consistently measurable, or evidence is scattered across teams. The reference architecture addresses this by insisting that each control objective produces a stable evidence stream.



Control objective	Platform mechanism	Evidence stream (audit-ready)
Least privilege and strong access control	IAM best practices + Kubernetes RBAC + IRSA	CloudTrail events + Kubernetes audit logs + access review records
Segmentation and reduced compliance scope	PCI scoping guidance + VPC boundaries + NetworkPolicies	Documented segmentation diagrams + policy manifests + traffic-flow testing results
Cryptographic protection and key custody	KMS + envelope encryption patterns	KMS key policies + key usage logs + encryption configuration evidence
Monitoring and threat detection	GuardDuty + Security Hub aggregation	Findings history + response timelines + centralized dashboards
Monitoring and threat continuous compliance	AWS Config rules + IaC/GitOps review	Findings history + response timelines + centralized dashboards
Configuration drift and continuous compliance	AWS Config rules + IaC/GitOps review	Config evaluation history + remedtion records + change tickets

Table 1. Example control-to-evidence mapping (PCI DSS / SOC 2 / operational resilience)

Case evidence and what it demonstrates

Case studies are most useful when used carefully. They demonstrate feasibility and organizational patterns more than they provide portable benchmarks.

Nasdaq: Large-scale analytics elasticity. Nasdaq company has reported a total of 70 billion records processed daily through its use of large-scale analytics elasticity (Nasdaq) and by using an AWS-based architecture Nasdaq was able to achieve measurable improvements in both the time it took to complete the data load and in query performance. The primary takeaway for this paper's purposes is that cloud native patterns apply to ingestions of financial high-volume data and analytics workflows just as much as they do to micro service API workflows. [28]

Siam Commercial Bank: Hybrid/on-prem constraints meet Kubernetes management. SCB also reported that they had reduced their new workload implementation time by 50% when they utilized EKS. The report explicitly referenced both latency and data residency as requirements that are critical in this context; specifically, for financial services organizations, regulatory or risk constraints frequently necessitate differentiated deployment locations, and the consistent use of a single Kubernetes operational model greatly increases efficiency through reducing friction between different environments. [46]

Capital One: Transaction-scale workloads and resilience engineering. Capital One's case study for Kubernetes shows how their use of the technology has been due to the need to manage fraud detection and credit decisioning workloads which are dealing with millions of transactions per day. The value in the case study is the connection to financial-grade workloads where resiliency is one of the drivers of the adoption of Kubernetes.[30]

Discover: Payment settlement and data platform modernization. The case study in Discover's presentation of using AWS network and object storage patterns to move transactions into Amazon S3 for pricing and settlement workflow purposes demonstrates how scalable, available, secure, and high-performance environments can be created. This provides evidence of financial institutions' ability to utilize cloud service providers for transaction flow because of designing architectures that control data movement within defined security boundary parameters. [47]

Implementation guidance

A financial architecture designed for submission quality should be at an implementation level of detail. The examples below are based on primary sources which provide information on the mechanisms behind the architectures that they describe.

Use of IRSA (Identity and Access Management Roles for Service Accounts) and AWS API Access:
Using IRSA allows pods to call AWS APIs using service account bound IAM roles instead of using a single set of node wide credentials. This limits the damage from a compromised pod and makes it easier to monitor usage. [42]

Workload Hardening: Baseline Workloads:
To limit the ability of malicious users to execute workloads without authorization, use Pod Security Admission Controller to enforce Pod Security Policies and select the policy tier that best fits the workload classification. Then, create cluster-wide policies with namespace-specific exception requests requiring explicit approval. [48]

East/West Segmentation:
Implement least privilege traffic flow within a network by creating Kubernetes Network Policies and combining these with native AWS boundaries, i.e., Security Groups for Pods when feasible.

Creation of Audit Trails:
Enable CloudTrail for AWS API calls and Kubernetes audit logs for all actions performed within the cluster. Store both sets of logs in immutable or append only storage for long term retention and forensic purposes.

Continuous Configuration Compliance Evaluation:
Evaluate AWS resources against rules and store historical configurations as audit trail records using AWS Config. [52]

Controlled Auto Scaling:
Create horizontal pod autoscalers (HPA) for latency sensitive applications and implement either Karpenter or Cluster Autoscaler for node capacity planning. To minimize scaling attacks, add WAF and bot control to the edges of your auto scaling process. [48]

Discussion and Conclusion

Benefits, with a financial-services lens

Cloud-native architectures are often sold on “developer velocity,” but financial outcomes are typically judged on stability, risk reduction, and auditability.

- Tighter elastic scaling timescales. The combination of load balancers, auto-scaling replicas and rapid provision of nodes enable business applications to react to large volumes of transaction bursts with less need for manual intervention; particularly where you know the scaling envelope (for example documented Lambda scale times) [49].
- Smaller 'blast radii' because of segmentation and identity boundary constructs. The use of Kubernetes namespaces, RBAC, network policies and pod security standards provides mechanisms for constructing smaller trust zones; whilst AWS identity and cryptography services provide workload specific permissions and audited key custody [50].
- Empirically supported compliance. Artifact from AWS supports the access of inherited control documentation; whilst CloudTrail + Kubernetes audit logs provide systemic records of change and access that can be used to strengthen the operational resilience and soc2 narratives [51].

Challenges and trade-offs

The same features that make cloud-native architectures powerful can create new failure modes.

- Risk of Configuration Complexity and Error- More distributed environments have a larger number of interfaces where configurations may go wrong. According to the Capital One breach study, misconfigured boundaries and poor boundary settings were likely the most prominent contributors to real world risk. [52]
- Automated Scaling Exploitation- Research on Economic Denial-of-Service (Edos) indicates that malicious exploitation of automated scaling functionality could lead to significant cost increases for a company. Thus, financial organizations should consider defense at the edge and the design of their auto-scaling policies as part of the same set of controls. [53]
- Failure to Properly Scope Compliance Boundaries- If a cloud environment is not properly segmented, the compliance boundaries required by PCI-DSS can grow significantly, increasing costs and audit complexity. This is why AWS has developed PCI-DSS Scoping Guidance. [45]

- Diverse Jurisdictions with Different Requirements- DORA requires companies to maintain operational resiliency and develop standards for managing third party risks. The New York Department of Financial Services (NYDFS) and the Gramm-Leach-Bliley Act (GLBA) Safeguards Rule require companies operating within the United States to implement specific operational security practices such as logging, testing, and incident response, etc. Companies cannot ignore these laws and regulations when designing their architecture. [54]

Future directions and research agenda

Several trends are likely to matter immediately for financial services cloud-native platforms.

- NIST's service mesh deployment guidance also describes how proxy-based service meshes can provide coordinated security and observability services for microservices through their use as a control distribution plane. The main advantage for financial institutions is the ability to apply consistent identity and policy enforcement at the service boundary without each organization having to develop its own bespoke mutual TLS and authorization logic.
- NIST SP 800-207 and subsequent works (e.g. NIST SP 800-207A), describe how to practically implement fine-grained identity-based controls in multi-cloud/hybrid environments; this directly relates to finance's "assume breach" posture. [55]
- The trend toward machine readable compliance artifacts (i.e. AWS PCI compliance packages and report automation) indicates a path toward continuous compliance systems that operate similarly to software systems (versioned, tested, monitored and reproducible). [56]
- EDoS research provides an explicit research agenda, which is that auto-scaling control loops require models of adversaries rather than just performance models. Financial services, due to the potential for rapid economic impact, are likely to be among the first to develop and deploy "cost-safe auto-scaling" patterns. [57]

Conclusion

Financial service application architectures based upon AWS and Kubernetes could provide scalable and secure applications, yet to achieve this they must also be compliance aware. There is sufficient evidence from Nasdaq ingestion and analytics results, and SCB workload implementation time reductions that financial service applications will benefit from increased scalability and operations improvement at a large scale; furthermore, there is evidence of financial service applications using Kubernetes as a platform to meet transaction level requirements and deploy reliable software [58].

The primary recommendation made by this research is architectural; therefore, treat scalability and security as related control systems. To use autoscaling you must first protect it from being triggered by an adversary, the identity used in each workload must be unique to that workload, segmentation must occur at two layers, the cloud layer and the cluster layer, cryptography must be treated as a defined boundary, and all actions taken by your system must produce continuous audit evidence through logging and configuration auditing. In doing so you will be able to satisfy modern regulatory requirements that focus on resiliency and the documentation of controls throughout the lifecycle of the application and not just at the point of audit [58].

References

[1] Ben David, Ronen. (2021). Kubernetes Autoscaling: YoYo Attack Vulnerability and Mitigation. 10.48550/arXiv.2105.00542.
[2] Anderson, Jamelia. (2022). Cloud Outsourcing in the Financial Sector: An Assessment of Internal Governance Strategies on a Cloud Transaction Between a Bank and a Leading Cloud Service Provider. European Business Organization Law Review. 23. 10.1007/s40804-022-00252-4. [3]
[3] Deng, Shuiguang & Zhao, Hailiang & Huang, Binbin & Zhang, Cheng & Chen, Feiyi & Deng, Yinuo & Yin, Jianwei & Dustdar, Schahram & Zomaya, Albert. (2023). Cloud-Native Computing: A Survey from the Perspective of Services. 10.36227/techrxiv.23500383. [4]
[4] Jiao, Qingsong & Xu, Botong & Fan, Yunjie. (2021). Design of Cloud Native Application Architecture Based on Kubernetes. 494-499. 10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00088. [5]
[5] Mendoza, Clarisse & Reyes, Cristina. (2023). EXPLORING THE IMPACT OF SHARED RESPONSIBILITY MODELS ON CLOUD SECURITY POSTURE AND VULNERABILITY MANAGEMENT. Journal of Emerging Technologies.
[6] Khan, Shaharyar & Kabanov, Ilya & Hua, Yunke & Madnick, Stuart. (2022). A Systematic Analysis of the Capital One Data Breach: Critical Lessons Learned. ACM Transactions on Privacy and Security. 26. 10.1145/3546068.
[7] Nguyen Thanh, Tung & Yeom, Yu-Jin & Kim, Taehong & Park, Daeheon & Kim, Sehan. (2020). Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration. Sensors. 20. 4621. 10.3390/s20164621.
[8] Kellogg, Martin & Schäf, Martin & Tasiran, Serdar & Ernst, Michael. (2020). Continuous compliance. 511-523. 10.1145/3324884.3416593.
[9] Theodoropoulos, Theodoros & Rosa, Luis & Benzaid, Chafika & Gray, Peter & Marin, Eduard & Makris, Antonios & Cordeiro, Luis & Diego, Ferran & Sorokin, Pavel & Di Girolamo, Marco & Barone, Paolo & Taleb, Tarik & Tserpes, Konstantinos. (2023). Security in Cloud-Native Services: A Survey. Journal of Cybersecurity and Privacy. 3. 758-793. 10.3390/jcp3040034.

[10] Wong, Ann & Chekole, Eyasu & Ochoa, Martín & Zhou, Jianying. (2023). On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. Computers & Security. 128. 103140. 10.1016/j.cose.2023.103140.
[11] Minna, Francesco & Massacci, Fabio. (2023). SoK: Run-time security for cloud microservices. Are we there yet?. Computers & Security. 127. 103119. 10.1016/j.cose.2023.103119.
[12] Yeoh, William & Liu, Marina & Shore, Malcolm & Jiang, Frank. (2023). Zero Trust Cybersecurity: Critical Success Factors and a Maturity Assessment Framework. Computers & Security. 133. 103412. 10.1016/j.cose.2023.103412.
[13] Rahaman, Sazzadur & Wang, Gang & Yao, Danfeng Daphne. (2019). Security Certification in Payment Card Industry: Testbeds, Measurements, and Recommendations. 481-498. 10.1145/3319535.3363195.
[14] Schoenfeld, Jordan. (2022). Cyber risk and voluntary Service Organization Control (SOC) audits. Review of Accounting Studies. 29. 10.1007/s11142-022-09713-0.
[15] Shastri, Supreeth & Wasserman, Melissa & Chidambaram, Vijay. (2019). GDPR Anti-Patterns: How Design and Operation of Modern Cloud-scale Systems Conflict with GDPR. 10.48550/arXiv.1911.00498.
[16] Mamun, Abdullah & Hassan, M. Kabir & Lai, Van SON. (2004). The Impact of the Gramm-Leach-Bliley Act on the Financial Service Industry. Journal of Economics and Finance. 28. 333-347. 10.1007/BF02751736.
[17] Hossain, Mohammad & Raza, Md Adil. (2023). EXPLORING THE EFFECTIVENESS OF MULTIFACTOR AUTHENTICATION IN PREVENTING UNAUTHORIZED ACCESS TO ONLINE BANKING SYSTEMS. SSRN Electronic Journal. 1. 8-12. 10.2139/ssrn.5207142.
[18] Javaheri, Danial & Fahmideh, Mahdi & Chizari, Hassan & Lalbakhsh, Pooia & Hur, Junbeom. (2023). Cybersecurity threats in FinTech: A systematic review. Expert Systems with Applications. 241. 122697. 10.1016/j.eswa.2023.122697.
[19] Deng, Shuiguang & Zhao, Hailiang & Huang, Binbin & Zhang, Cheng & Chen, Feiyi & Deng, Yinuo & Yin, Jianwei & Dustdar, Schahram & Zomaya, Albert. (2023). Cloud-Native Computing: A Survey from the Perspective of Services. 10.36227/techrxiv.23500383.
[20] Afzal, Shahbaz & Ganesh, Kavitha. (2019). Load balancing in cloud computing -A hierarchical taxonomical classification. Journal of Cloud Computing. 8. 10.1186/s13677-019-0146-7.
[21] Wang, Hao & Wang, Yong & Liang, Guanying & Gao, Yunfan & Gao, Weijian & Zhang, Wenping. (2021). Research on load balancing technology for microservice architecture. MATEC Web of Conferences. 336. 08002. 10.1051/matecconf/202133608002.
[22] Anis Aziz, Wagdy. (2023). Auto Scaling Solutions for Cloud Applications.
[23] Trần, Minh-Ngọc & Vu, Dinh-Dai & Kim, Younghan. (2022). A Survey of Autoscaling in Kubernetes. 263-265. 10.1109/ICUFN55119.2022.9829572.
[24] Taherizadeh, Salman & Grobelnik, Marko. (2019). Key influencing factors of the Kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications. Advances in Engineering Software. 140. 10.1016/j.advengsoft.2019.102734.
[25] Ngo, Kim & Mukherjee, Joydeep & Jiang, Zhen & Litoiu, Marin. (2022). Evaluating the Scalability and Elasticity of Function as a Service Platform. 117-124. 10.1145/3489525.3511682.
[26] Berisha, Blend & Mëziu, Endrit & Shabani, Isak. (2022). Big data analytics in Cloud computing: an overview. Journal of Cloud Computing. 11. 10.1186/s13677-022-00301-w.
[27] Pahl, Claus & Jamshidi, Pooyan. (2016). Microservices: A Systematic Mapping Study. 137-146. 10.5220/0005785501370146.
[28] Bawa, Parminder & Rehman, Shafiq & Manickam, Selvakumar. (2019). Comparative Analysis of State-of-the-Art EDoS Mitigation Techniques in Cloud Computing Environment. 10.48550/arXiv.1905.13447.
[29] Islam, Syed & Kumar, Mohankumar & Jannat, Umma. (2023). Exploring the Effectiveness of Web Application Firewalls Against Diverse Attack Vectors. 1798-1806. 10.1109/ICECA58529.2023.10395379.
[30] Deimling, Franz & Fazzolari, Michela. (2023). AMOE: a Tool to Automatically Extract and Assess Organizational Evidence for Continuous Cloud Audit. 10.48550/arXiv.2307.16541.
[31] Neelakandhan, Maya & Ramprakash, Guruprasad & Gaidhani, Mrudula. (2022). Achieving least privilege at cloud scale with cloud infrastructure entitlements management. Cyber Security: A Peer-Reviewed Journal. 5. 10.69554/NDIC4652.
[32] Kummarapurugu, Charan Shankar. (2019). A Comparative Analysis of OAuth 2.0 and OpenID Connect for Identity Federation in Cloud Environments. World Journal of Advanced Research and Reviews. 1. 054-060. 10.30574/wjarr.2019.1.2.0017.
[33] Kamadi, Sandeep. (2023). Identity-Driven Zero Trust Automation in GitOps: Policy-as-Code Enforcement for Secure code Deployments. International Journal of Scientific Research in Computer Science Engineering and Information Technology. 893. 10.32628/CSEIT235148.
[34] Kuzminykh, Levgeniia & Ghita, B.V. & Shiaeles, Stavros. (2021). Comparative Analysis of Cryptographic Key Management Systems. 10.48550/arXiv.2109.09905.
[35] Basta, Nardine & Ikram, Muhammad & Kaafar, Dali & Walker, Andy. (2021). Towards a Zero-Trust Micro-segmentation Network Security Strategy: An Evaluation Framework. 10.48550/arXiv.2111.10967.
[36] Budigiri, Gerald & Baumann, Christoph & Muhlberg, Jan & Truyen, Eddy & Joosen, Wouter. (2021). Network Policies in Kubernetes: Performance Evaluation and Security Analysis. 407-412. 10.1109/EuCNC/6GSummit51104.2021.9482526.
[37] Shamim, Md Shazibul Islam & Bhuiyan, Farzana Ahamed & Rahman, Akond. (2020). XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices. 58-64. 10.1109/SecDev45635.2020.00025

[38] Jiao, Qingsong & Xu, Botong & Fan, Yunjie. (2021). Design of Cloud Native Application Architecture Based on Kubernetes. 494-499. 10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00088.
[39] Yang, Yutian & Shen, Wenbo & Ruan, Bonan & Liu, Wenmao & Ren, Kui. (2021). Security Challenges in the Container Cloud. 137-145. 10.1109/TPSISA52974.2021.00016.
[40] Tenkrooden, Kyle. (2023). Ensuring GDPR Compliance for a Small-Scale Dynamic Web Application Using AWS Cloud Services: Best Practices and Strategies.
[41] Guduru, Sandhya. (2020). Cloud Security Automation: Enforcing CIS Benchmarks with AWS Config, Azure Policy, and OpenStack Chef Cookbooks. The Journal of Scientific and Engineering Research. 7. 243-248. 10.5281/zenodo.15234591.
[42] Ozer, M. & Varlioglu, Said & Gonen, Bilal & Adewopo, Victor & Elsayed, Nelly & Zengin, Selcuk. (2020). Cloud Incident Response: Challenges and Opportunities. 49-54. 10.1109/CSCI51800.2020.00015.
[43] Scott, Hal. (2021). The E.U.'s Digital Operational Resilience Act: Cloud Services & Financial Companies. SSRN Electronic Journal. 10.2139/ssrn.3904113.
[44] Adeyemi, Ibrahim & Martin, Chloe & Samuel, Adebis. (2021). Kubernetes Control Plane and Node Architecture Explained.
[45] Theodoropoulos, Theodoros & Rosa, Luis & Benzaid, Chafika & Gray, Peter & Marin, Eduard & Makris, Antonios & Cordeiro, Luis & Diego, Ferran & Sorokin, Pavel & Di Girolamo, Marco & Barone, Paolo & Taleb, Tarik & Tserpes, Konstantinos. (2023). Security in Cloud-Native Services: A Survey. Journal of Cybersecurity and Privacy. 3. 758-793. 10.3390/jcp3040034.
[46] Surantha, Nico & Ivan, Felix & Chandra, Ritchie. (2023). A case analysis for Kubernetes network security of financial service industry in Indonesia using zero trust model. Bulletin of Electrical Engineering and Informatics. 12. 3134-3141. 10.11591/eei.v12i5.4240.
[47] Salmon, Maraehau & Foroughi, Tabesh & Parmar, Akash. (2022). Cloud Computing - Providers & Users (April 2022). 10.13140/RG.2.2.29905.40809.
[48] Mondal, Subrota & Wu, Xiaohai & Kabir, H M Dipu & Dai, Hong-Ning & Ni, Kan & Yuan, Honggang & Wang, Ting. (2023). Toward Optimal Load Prediction and Customizable Autoscaling Scheme for Kubernetes. Mathematics. 11. 2675. 10.3390/math11122675.
[49] Anis Aziz, Wagdy. (2023). Auto Scaling Solutions for Cloud Applications. International Journal of Knowledge and Systems Science. 24.
[50] Surantha, Nico & Ivan, Felix. (2020). Secure Kubernetes Networking Design Based on Zero Trust Model: A Case Study of Financial Service Enterprise in Indonesia. 10.1007/978-3-030-22263-5_34.
[51] Piper, Ben & Clinton, David. (2019). CloudTrail, CloudWatch, and AWS Config. 10.1002/9781119560395.ch7.
[52] Khan, Shaharyar & Kabanov, Ilya & Hua, Yunke & Madnick, Stuart. (2022). A Systematic Analysis of the Capital One Data Breach: Critical Lessons Learned. ACM Transactions on Privacy and Security. 26. 10.1145/3546068.
[53] Ben David, Ronen. (2021). Kubernetes Autoscaling: YoYo Attack Vulnerability and Mitigation. 10.48550/arXiv.2105.00542.
[54] Vijayabaskar, Santhosh & Rao, Pattabi & Kanchi, Pavan & Jain, Shalu & Agarwal, Raghav. (2023). Integrating Cloud-Native Solutions in Financial Services for Enhanced Operational Efficiency. Universal Research Reports. 10. 402-419. 10.36676/urr.v10.i4.1355.
[55] Surantha, Nico & Ivan, Felix & Chandra, Ritchie. (2023). A case analysis for Kubernetes network security of financial service industry in Indonesia using zero trust model. Bulletin of Electrical Engineering and Informatics. 12. 3134-3141. 10.11591/eei.v12i5.4240.
[56] Chowdhury, Fahad & Kiah, Laiha & Ahsan, M.. (2017). Economic denial of sustainability (EDoS) mitigation approaches in cloud: Analysis and open challenges. 206-211. 10.1109/ICECOS.2017.8167135.
[57] Surantha, Nico & Ivan, Felix & Chandra, Ritchie. (2023). A case analysis for Kubernetes network security of financial service industry in Indonesia using zero trust model. Bulletin of Electrical Engineering and Informatics. 12. 3134-3141. 10.11591/eei.v12i5.4240.
[58] Neumannova, Anita & Bernroider, Edward & Elshuber, Christoph. (2023). The Digital Operational Resilience Act for Financial Services: A Comparative Gap Analysis and Literature Review. 10.1007/978-3-031-30694-5_40