



FINAL PROJECT REPORT

Submitted by:

Alago Chiemela Victor

Khanh Nguyen Tran

Duc-Tin Nguyen

This report represents the culmination of our efforts in fulfilling the requirements
for the course in

CLOUD COMPUTING

, conducted at

EPITA,

School of Engineering and Computer Science.

Academic Year: 2023/2024

Location: Paris, France

Date: 12th April 2024.

TABLE OF CONTENTS

INTRODUCTION.....	5
Project Overview.....	5
Project Aims and Objectives.....	6
Problem Statement and Scope.....	7
BACKGROUND AND LITERATURE REVIEW.....	8
Relevant Technologies and Standards.....	8
Previous Research and Studies.....	8
ARCHITECTURE PROPOSAL.....	9
Scalability.....	9
Performance Optimization.....	10
Cost Optimization.....	11
Security and Privacy Measures.....	12
Disaster Recovery and Data Integrity.....	13
REFLECTION QUESTIONS ON PROPOSED ARCHITECTURE.....	14
Scalability Impact:.....	14
User Accessibility:.....	15
Interoperability Challenges:.....	16
Telemedicine Ethical Considerations:.....	17
Data Security Measures:.....	18
Resource Optimization:.....	19
Adaptability to Technological Advancements:.....	20
User Training and Support:.....	21
Disaster Recovery and Contingency Planning:.....	22
IMPLEMENTATION AND DEVELOPMENT PROCESS.....	23
Development Environment Setup:.....	23
Technology Stack and Frameworks:.....	23
Implementation Challenges and Solutions:.....	23
TESTING AND EVALUATION.....	24
Testing Strategies and Scenarios:.....	24
Performance Testing:.....	24
Usability Testing:.....	24
CONCLUSION.....	25
Summary of Project Outcomes:.....	25
Limitations and Future Work:.....	25
APPENDICES.....	27
Architectural Diagrams:.....	27
Code Snippets and Configuration Files:.....	30
• Amazon Eventbridge:.....	30
• AWS Backup:.....	35
• AWS CloudWatch:.....	42
• AWS SNS:.....	43
• AWS Cognito:.....	46
• AWS DynamoDB:.....	56
• AWS S3:.....	60
• AWS EC2:.....	65
• AWS Cost Explorer:.....	75
• IAM (Identity and Access Management):.....	76
• VPC:.....	78
• Attach API Gateway to ec2 Flask App:.....	79
• FLask App source code:.....	90
• App overview:.....	91
References and Resources:.....	99

INTRODUCTION

Project Overview

The project endeavors to craft a sophisticated cloud-based Healthcare Management System (HMS) aimed at surmounting the intricate challenges encountered by healthcare institutions in managing patient data, fortifying data security, and furnishing streamlined healthcare services. By harnessing the capabilities inherent in cloud computing, this initiative seeks to redefine patient care by digitizing healthcare services, augmenting access to medical records, and fostering seamless collaboration among healthcare practitioners.

Key Objectives:

1. Develop a meticulously structured architecture for the Healthcare Management System (HMS) to be seamlessly deployed within the AWS cloud infrastructure.
2. Address the imperative for interoperability and integration with extant healthcare systems, adhering rigorously to industry standards such as HL7 for seamless healthcare data exchange.
3. Engineer a robust telemedicine infrastructure within the architectural framework to facilitate virtual consultations, remote monitoring, and telehealth services, while ensuring compliance with prevailing telemedicine regulations and guidelines.
4. Prioritize user experience by fashioning an intuitive system interface catering to the diverse needs of both healthcare professionals and patients, inclusive of features like mobile accessibility and personalized dashboards.
5. Optimize system performance to guarantee swift access to electronic health records and the seamless execution of telemedicine services.
6. Implement methodologies for cost optimization, meticulously monitoring and managing cloud resource consumption to deliver high-quality healthcare services within predefined budgetary parameters.
7. Enforce stringent adherence to healthcare regulations, notably HIPAA, through robust security measures encompassing encryption, access controls, and proactive threat detection.
8. Devising a comprehensive strategy for data backup and disaster recovery to safeguard data integrity and ensure operational continuity under all circumstances.
9. Leverage containers and orchestration techniques within the architecture to streamline pipeline flows and optimize resource utilization for enhanced application performance.

In summary, the project endeavors to conceive a scalable, secure, and operationally efficient cloud-based Healthcare Management System, poised to elevate patient care standards, foster enhanced collaboration among healthcare stakeholders, and uphold the utmost standards of data security and privacy.

Project Aims and Objectives

Aims:

1. To develop a cutting-edge cloud-based Healthcare Management System (HMS) that addresses the intricate challenges prevalent in modern healthcare operations.
2. To leverage cloud computing technologies to enhance patient care, streamline medical record management, and optimize healthcare service delivery.
3. To establish a scalable and secure platform that promotes seamless collaboration among healthcare professionals while safeguarding the integrity and confidentiality of patient data.
4. To prioritize user experience by designing an intuitive and accessible system interface tailored to the diverse needs of healthcare practitioners and patients alike.
5. To ensure compliance with healthcare regulations and standards, such as HIPAA, by implementing robust security measures and data protection protocols.

Objectives:

1. Design a comprehensive architecture for the HMS that integrates seamlessly with existing healthcare systems and facilitates interoperability with third-party applications.
2. Develop a telemedicine infrastructure within the HMS to enable virtual consultations, remote monitoring, and telehealth services, ensuring compliance with telemedicine regulations.
3. Optimize system performance to ensure fast access to electronic health records and seamless execution of telemedicine services, minimizing latency and maximizing resource utilization.
4. Implement cost optimization strategies to monitor and control cloud resource usage, delivering high-quality healthcare services within budgetary constraints.
5. Establish stringent security measures, including encryption, access controls, and threat detection, to protect sensitive patient data and ensure compliance with healthcare privacy regulations.
6. Devise a comprehensive plan for data backup and disaster recovery to ensure data availability and operational continuity under all circumstances.
7. Leverage emerging technologies and trends, such as AI algorithms and remote patient monitoring devices, to enhance the adaptability and functionality of the HMS.
8. Provide adequate user training and support to enable healthcare professionals and patients to effectively utilize the HMS and its features.
9. Conduct thorough testing and evaluation of the HMS to ensure reliability, performance, and usability, addressing any implementation challenges encountered during the development process.
10. Prepare a detailed documentation of the HMS architecture, implementation process, and testing results for comprehensive understanding and future reference.

Problem Statement and Scope

Problem Statement: Healthcare organizations are confronted with multifaceted challenges in managing patient data, ensuring seamless access to medical records, upholding data security, and delivering efficient healthcare services. Traditional systems often lack scalability, interoperability, and advanced features essential to meet the evolving demands of modern healthcare. Additionally, maintaining compliance with stringent healthcare regulations, such as HIPAA, further complicates these challenges.

Scope: The scope of this project encompasses the design and implementation of a cloud-based Healthcare Management System (HMS) to address the aforementioned challenges faced by healthcare organizations. Leveraging cloud computing technologies, the HMS aims to enhance patient care, streamline medical record management, and optimize healthcare service delivery.

Key objectives within the project scope include:

1. Designing a comprehensive architecture for the HMS that ensures interoperability with existing healthcare systems and seamless integration with third party applications, adhering to standards such as HL7 for healthcare data exchange.
2. Developing a robust telemedicine infrastructure within the HMS to facilitate virtual consultations, remote monitoring, and telehealth services, while ensuring compliance with telemedicine regulations and guidelines.
3. Prioritizing user experience by designing an intuitive and accessible system interface for healthcare professionals and patients, incorporating features such as mobile access, personalized dashboards, and efficient navigation of health records.
4. Implementing stringent security measures, including encryption, access controls, and threat detection, to safeguard sensitive patient data and ensure compliance with healthcare privacy regulations like HIPAA.
5. Establishing robust data backup and disaster recovery mechanisms to ensure data availability and operational continuity in various scenarios, mitigating the risk of data loss and maintaining data integrity.
6. Optimizing system performance to ensure fast access to electronic health records and seamless execution of telemedicine services, while monitoring and controlling cloud resource usage to stay within budgetary constraints.
7. Leveraging emerging technologies and trends, such as AI algorithms and remote patient monitoring devices, to enhance the functionality and adaptability of the HMS, future proofing the system against technological advancements.
8. Providing comprehensive user training and support to enable healthcare professionals and patients to effectively utilize the HMS and its features, fostering user adoption and satisfaction.
9. Conducting thorough testing and evaluation of the HMS to ensure reliability, performance, and usability, addressing any implementation challenges encountered during the development process.
10. The culmination of the project will involve delivering a fully functional cloud based HMS deployed on the AWS cloud, accompanied by comprehensive documentation and a presentation outlining the architecture, implementation process, and key outcomes.

BACKGROUND AND LITERATURE REVIEW

Relevant Technologies and Standards

The advancement of cloud computing technologies has significantly transformed the healthcare industry, offering innovative solutions to address various challenges. Key technologies and standards relevant to the development of cloud-based Healthcare Management Systems (HMS) include:

1. **Cloud Computing Platforms:** Platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform provide scalable infrastructure and a wide range of services essential for deploying healthcare applications in the cloud.
2. **Interoperability Standards:** Standards like Health Level Seven (HL7) facilitate seamless data exchange between disparate healthcare systems, ensuring interoperability and integration.
3. **Telemedicine Infrastructure:** Telemedicine platforms enable virtual consultations, remote monitoring, and telehealth services, enhancing access to healthcare and improving patient outcomes.
4. **Security and Compliance Frameworks:** Compliance with healthcare regulations such as the Health Insurance Portability and Accountability Act (HIPAA) is essential for safeguarding patient data. Security measures, including encryption, access controls, and threat detection, are crucial for maintaining data privacy and integrity in healthcare systems.

Previous Research and Studies

Numerous research studies and projects have explored the design, implementation, and impact of cloud-based HMS. These studies have contributed valuable insights into various aspects of healthcare management and technology integration. Some key areas of research include:

1. Scalability and Performance Optimization: Studies have focused on designing architectures that can efficiently scale to accommodate growing patient data volumes and optimize system performance to ensure fast access to electronic health records.
2. Cost Optimization Strategies: Research has explored methods for monitoring and controlling cloud resource usage to deliver high-quality healthcare services within budget constraints, including considerations for resource allocation and pricing models.
3. Security and Privacy Measures: Investigations into security measures, such as encryption, access controls, and threat detection, have aimed to protect sensitive patient data and ensure compliance with healthcare regulations like HIPAA.
4. Disaster Recovery and Data Integrity: Studies have proposed comprehensive plans for data backup and disaster recovery to ensure data availability and operational continuity in various scenarios, mitigating the risk of data loss and maintaining data integrity in healthcare systems.

By reviewing relevant literature and previous research studies, insights can be gained to inform the design and implementation of an effective cloud-based HMS that addresses the challenges and requirements of modern healthcare organizations.

ARCHITECTURE PROPOSAL

Scalability

To address the scalability requirements of the Healthcare Management System (HMS), a combination of AWS services will be utilized:

1. **Elastic Load Balancer (ELB):** ELB acts as a traffic distributor, ensuring that incoming requests are evenly distributed across multiple targets, such as EC2 instances and Lambda functions. By dynamically adjusting the load distribution, ELB ensures that the system can handle varying levels of traffic, scaling horizontally to accommodate peak usage periods efficiently while maintaining optimal performance and responsiveness for users.
2. **Lambda:** Lambda functions will play a crucial role in scaling the HMS architecture. By executing code in response to events triggered by user interactions or system events, Lambda enables the system to dynamically scale its compute resources based on demand. For example, during periods of high demand, Lambda can automatically provision additional resources to handle incoming requests, ensuring that the system can scale seamlessly without the need for manual intervention.
3. **Amazon EC2:** EC2 instances will host various components of the HMS, including web servers, application servers, and database servers. EC2 provides scalable compute capacity, allowing the system to add or remove instances based on workload requirements. By leveraging EC2 Auto Scaling, the system can automatically adjust the number of EC2 instances based on predefined scaling policies, ensuring that the infrastructure can accommodate the growing volume of patient data and increasing numbers of users.
4. **Amazon DynamoDB:** DynamoDB will serve as the primary data store for patient data and healthcare records. As a fully managed, highly scalable NoSQL database service, DynamoDB enables the system to store and retrieve data with low latency and high throughput. By utilizing DynamoDB's auto-scaling feature, the system can automatically adjust its read and write capacity to accommodate changes in workload, ensuring that the database can scale seamlessly as the system grows.

Performance Optimization

To optimize the performance of the HMS and ensure fast access to electronic health records (EHRs) and other healthcare data, the following strategies and AWS services will be implemented:

1. **Lambda:** Lambda functions will be utilized to enhance performance by executing code in response to events, such as user requests for accessing medical records or initiating telemedicine consultations. By leveraging Lambda, the system can execute compute-intensive tasks asynchronously, minimizing latency and improving responsiveness for users.
2. **Amazon DynamoDB:** DynamoDB will be utilized for fast and reliable access to healthcare data. With its low latency and high throughput, DynamoDB ensures that the system can retrieve data quickly, enabling healthcare professionals to access patient information in real-time during consultations and treatment sessions.
3. **Amazon S3:** S3 will be used to store and retrieve unstructured data, such as medical images and documents, with high durability and performance. By offloading the storage of large files to S3, the system can reduce the load on the database and improve overall performance, ensuring efficient handling of multimedia data associated with patient records.

Cost Optimization

To optimize costs while delivering high-quality healthcare services, the following strategies and AWS services will be implemented:

1. **AWS Cost Explorer:** AWS Cost Explorer will be used to monitor cloud resource usage and identify cost-saving opportunities. By analyzing usage patterns and optimizing resource allocation, the system can ensure that healthcare services remain within budget constraints while maintaining performance and availability.
2. **Reserved Instances and Spot Instances:** Reserved Instances and Spot Instances will be utilized for cost-effective resource allocation. Reserved Instances offer significant cost savings compared to On-Demand Instances and are ideal for predictable workloads with steady-state usage. Spot Instances, on the other hand, provide access to unused EC2 capacity at discounted rates and are suitable for non-critical tasks or bursty workloads. By leveraging a combination of Reserved Instances and Spot Instances, the system can optimize pricing models and reduce infrastructure costs without sacrificing performance or reliability.

Security and Privacy Measures

To ensure the security and privacy of patient data and comply with healthcare regulations like HIPAA, the following security measures and AWS services will be implemented:

1. **Amazon Cognito:** Cognito will be used to ensure secure user authentication and authorization, controlling access to healthcare applications and services. By integrating Cognito with identity providers and enforcing multi-factor authentication, the system can protect sensitive data from unauthorized access.
2. **VPC (Virtual Private Cloud):** A VPC will be configured to host healthcare services securely within isolated network environments. By segmenting the network and implementing security groups and network ACLs, the system can prevent unauthorized access and mitigate the risk of network-based attacks.
3. **IAM (Identity and Access Management):** IAM will be used to manage access to AWS resources securely, enforcing role-based access control for healthcare data. By assigning granular permissions to users and resources, the system can ensure that only authorized individuals can access sensitive data, reducing the risk of data breaches.
4. **Firewall Policy:** A firewall policy will be implemented to implement network security measures and protect healthcare applications from unauthorized access and cyber threats. By defining inbound and outbound rules and monitoring network traffic, the system can detect and block malicious activity, safeguarding sensitive data from external threats.
5. **Encryption at Rest and in Transit:** Encryption mechanisms will be utilized to protect sensitive patient data both at rest and in transit. Data stored in DynamoDB and S3 will be encrypted using server-side encryption (SSE), while data transmitted over the network will be encrypted using SSL/TLS protocols, ensuring compliance with healthcare regulations like HIPAA.
6. **AWS CloudTrail:** CloudTrail will be used to log and monitor API activity and resource usage, aiding in security analysis and compliance auditing. By capturing API calls and identifying unauthorized or suspicious activity, CloudTrail helps to ensure the integrity and confidentiality of healthcare information stored in AWS.

Disaster Recovery and Data Integrity

To ensure data availability and operational continuity in the event of disasters or system failures, the following AWS services will be utilized:

1. **Amazon S3:** S3 will be used to store and retrieve unstructured data with high durability and reliability. By replicating data across multiple Availability Zones (AZs) and utilizing versioning and cross-region replication, S3 ensures that data remains accessible even in the event of hardware failures or natural disasters.
2. **AWS Backup:** AWS Backup will be used to implement backup and recovery strategies for healthcare data. By automating the backup process and scheduling regular backups of critical data, the system can safeguard against data loss and ensure business continuity in the face of unexpected events or system failures.

REFLECTION QUESTIONS ON PROPOSED ARCHITECTURE

Scalability Impact:

The proposed architecture is designed to accommodate future scalability needs by leveraging scalable AWS services such as Elastic Load Balancer (ELB), Lambda, EC2, and Amazon DynamoDB. These services allow the system to dynamically adjust its resources based on demand, ensuring that it can scale horizontally to handle increasing volumes of patient data, growing numbers of users, and the integration of new healthcare services and technologies.

During periods of rapid growth or increased demand, the architecture may face several potential challenges:

- 1. Resource Provisioning:** Scaling up resources dynamically requires careful monitoring and management to ensure that sufficient resources are provisioned to handle increased demand effectively. Failure to allocate resources appropriately may lead to performance degradation or service interruptions.
- 2. Elastic Load Balancing:** While ELB distributes incoming traffic across multiple targets to handle peak usage periods efficiently, it is essential to monitor its performance and adjust configuration settings as needed. Inadequate load balancing may result in uneven distribution of traffic, leading to overloading of certain resources and underutilization of others.
- 3. Lambda Invocation Limits:** Lambda functions have default concurrency limits that may impact scalability during periods of high demand. To avoid throttling and ensure smooth operation, it is necessary to monitor Lambda invocations closely and adjust concurrency settings or request quota increases as needed.
- 4. Data Storage Scalability:** Amazon DynamoDB provides scalable, high-performance storage for patient data and healthcare records. However, it is essential to monitor and manage DynamoDB capacity to prevent throttling and maintain optimal performance as data volumes increase over time.

By addressing these challenges through proactive monitoring, capacity planning, and optimization of AWS services, the proposed architecture can effectively accommodate future scalability needs and ensure seamless operation during periods of rapid growth or increased demand.

User Accessibility:

The architecture prioritizes user accessibility for both patients and healthcare professionals by incorporating features and design considerations tailored to diverse user needs, including those with limited technical proficiency. Here's how the architecture addresses user accessibility:

1. **Intuitive User Interfaces:** The user interfaces for both patients and healthcare professionals are designed to be intuitive and user-friendly, featuring clear navigation, simplified workflows, and visually appealing layouts. This design approach aims to minimize the learning curve and make it easy for users of all technical levels to interact with the system effectively.
2. **Mobile Access:** Recognizing the importance of mobile access in modern healthcare delivery, the architecture ensures compatibility with various devices, including smartphones and tablets. Mobile-responsive design principles are employed to optimize the user experience across different screen sizes, enabling users to access the system conveniently from anywhere, at any time.
3. **Personalized Dashboards:** Both patients and healthcare professionals have access to personalized dashboards that display relevant information and functionalities based on their roles, preferences, and historical interactions with the system. Personalization enhances user engagement and efficiency by presenting pertinent data and tasks upfront, reducing the need for manual navigation and search.
4. **Multimodal Communication:** To accommodate users with diverse communication preferences and accessibility needs, the architecture supports multimodal communication channels, including text-based messaging, voice commands, and video conferencing. Integration with services like Amazon Chime enables seamless communication and collaboration among healthcare team members and facilitates virtual consultations with patients.
5. **Accessibility Standards Compliance:** The architecture adheres to accessibility standards and guidelines, such as the Web Content Accessibility Guidelines (WCAG), to ensure that the user interfaces are accessible to users with disabilities. This includes features such as keyboard navigation, screen reader compatibility, and text alternatives for non-text content, making the system inclusive and usable for all individuals.

By prioritizing these accessibility features and design principles, the architecture aims to provide an inclusive user experience that empowers patients and healthcare professionals alike, regardless of their technical proficiency or accessibility requirements.

Interoperability Challenges:

Challenges:

1. **Diverse Standards and Protocols:** Existing healthcare systems may use different data standards and communication protocols, making interoperability complex. For example, healthcare organizations may use HL7 (Health Level Seven) for data exchange, while others may have proprietary formats.
2. **Legacy Systems Integration:** Many healthcare systems rely on legacy technologies and may lack modern APIs or integration capabilities, complicating the integration process. Legacy systems may also have limited support for newer communication standards.
3. **Data Mapping and Transformation:** Mapping data fields between disparate systems and ensuring data consistency and integrity during the exchange process can be challenging. Differences in data schemas and terminology pose additional complexities.
4. **Security and Privacy Concerns:** Interoperability efforts must prioritize security and privacy to protect sensitive patient data during transit and integration. Compliance with regulations like HIPAA adds another layer of complexity.

Opportunities and Mitigation Strategies:

1. **Standardization and Protocol Adoption:** Embrace industry-standard data exchange protocols like HL7 FHIR (Fast Healthcare Interoperability Resources) to promote interoperability. Encourage healthcare partners to adopt standardized formats for data exchange.
2. **API-based Integration:** Implement APIs (Application Programming Interfaces) to facilitate seamless integration with existing healthcare systems. APIs abstract the complexities of underlying systems and provide a standardized interface for data exchange.
3. **Data Mapping Tools and Middleware:** Utilize data mapping tools and middleware platforms to streamline data transformation and mapping processes. These tools automate data conversion tasks and support interoperability between disparate systems.
4. **Security and Compliance Frameworks:** Implement robust security and privacy measures, such as data encryption, access controls, and audit trails, to protect patient data during transit and integration. Ensure compliance with healthcare regulations and standards to maintain data integrity and confidentiality.
5. **Collaboration and Partnership:** Foster collaboration with healthcare stakeholders and technology vendors to address interoperability challenges collectively. Engage in industry initiatives and forums focused on interoperability to share best practices and standards.

By addressing these challenges and leveraging opportunities, the architecture can achieve seamless interoperability with existing healthcare systems, enabling efficient data exchange and collaboration across the healthcare ecosystem.

Telemedicine Ethical Considerations:

1. **Patient Privacy and Confidentiality:** Protecting patient privacy is paramount in telemedicine. Encryption techniques and secure communication channels are implemented to safeguard patient information during telemedicine consultations. Access controls and authentication mechanisms ensure that only authorized healthcare professionals can access patient data.
2. **Informed Consent:** Patients must provide informed consent before participating in telemedicine consultations. The telemedicine platform incorporates features to obtain and document patient consent, explaining the nature of telemedicine services, potential risks, and benefits. Patients are informed about the limitations of virtual care compared to in-person visits and have the option to decline or opt-out of telemedicine consultations.
3. **Quality of Care and Professionalism:** Telemedicine consultations adhere to the same standards of care and professionalism as traditional healthcare encounters. Healthcare providers undergo training to effectively communicate and diagnose patients remotely, ensuring the delivery of high-quality care. The telemedicine platform supports multimedia capabilities, enabling detailed discussions and visual assessments during virtual consultations.
4. **Continuity of Care:** Telemedicine infrastructure prioritizes continuity of care by enabling seamless integration with electronic health records (EHRs) and existing healthcare systems. Patient medical histories, treatment plans, and diagnostic results are accessible to healthcare providers during telemedicine consultations, facilitating informed decision-making and comprehensive care delivery.
5. **Patient Empowerment and Accessibility:** Telemedicine enhances patient empowerment and accessibility by removing geographical barriers to healthcare access. Patients in remote or underserved areas can receive timely medical advice and treatment without the need for travel. The telemedicine platform incorporates user-friendly interfaces and accessibility features to accommodate patients with diverse needs and abilities.
6. **Cultural Sensitivity and Diversity:** Telemedicine services are culturally sensitive and inclusive, respecting patients' cultural beliefs, preferences, and language requirements. Interpretation services and multilingual support are available to overcome language barriers and ensure effective communication between patients and healthcare providers.

By addressing these ethical considerations, the telemedicine infrastructure promotes patient-centered care, privacy protection, and ethical practice standards in virtual healthcare delivery.

Data Security Measures:

The architecture incorporates robust data security measures to address concerns regarding the protection of sensitive healthcare information and ensure compliance with healthcare privacy regulations such as HIPAA (Health Insurance Portability and Accountability Act). Here's how the architecture addresses data security concerns:

1. **Encryption Mechanisms:** All sensitive healthcare data, including patient records, diagnostic reports, and treatment plans, are encrypted both at rest and in transit. Encryption ensures that data remains confidential and unreadable to unauthorized users even if it is intercepted or accessed illicitly.
2. **Access Control Policies:** Identity and Access Management (IAM) tools are implemented to enforce role-based access control (RBAC) policies. Healthcare professionals are granted access to patient data based on their roles and responsibilities, ensuring that only authorized personnel can view or modify sensitive information. Access permissions are regularly reviewed and updated to reflect changes in personnel roles or responsibilities.
3. **Network Security:** Virtual Private Cloud (VPC) configurations are employed to create isolated network environments for hosting healthcare applications and services. Network access controls, such as firewalls and security groups, are implemented to restrict unauthorized access to healthcare systems and prevent external threats from infiltrating the network.
4. **Audit Trails and Logging:** AWS CloudTrail is utilized to log and monitor API activity and resource usage within the cloud environment. Audit trails provide visibility into user actions and system activities, enabling administrators to track access attempts, detect unauthorized activities, and investigate security incidents promptly.
5. **Compliance Monitoring and Reporting:** The architecture includes mechanisms for monitoring and reporting compliance with healthcare privacy regulations. Regular compliance assessments and audits are conducted to ensure adherence to HIPAA requirements and other applicable regulations. Compliance reports are generated to document security controls, risk assessments, and mitigation strategies, facilitating regulatory compliance and demonstrating due diligence to regulatory authorities.
6. **Data Residency and Sovereignty:** Data residency and sovereignty requirements are addressed to ensure that healthcare data is stored and processed in compliance with local laws and regulations. Data centers and regions are selected based on regulatory requirements and contractual obligations to maintain data sovereignty and facilitate cross-border data transfers in accordance with applicable legal frameworks.

By implementing these data security measures, the architecture provides a robust framework for protecting sensitive healthcare information, maintaining data privacy, and complying with healthcare privacy regulations to safeguard patient confidentiality and trust.

Resource Optimization:

The proposed architecture incorporates several strategies to optimize cloud resources effectively, balancing cost-effectiveness with the delivery of high-quality healthcare services. Here's how the architecture achieves resource optimization and ensures ongoing cost monitoring:

- 1. Auto Scaling and Elasticity:** The architecture leverages Auto Scaling capabilities to dynamically adjust resource capacity based on changing demand patterns. Auto Scaling ensures that the healthcare system can automatically scale up or down in response to fluctuations in user traffic, optimizing resource utilization and minimizing costs during periods of low demand. By automatically provisioning and de-provisioning resources as needed, Auto Scaling ensures that the healthcare system maintains optimal performance while minimizing unnecessary resource consumption and associated costs.
- 2. Serverless Computing:** Serverless computing services, such as AWS Lambda, are utilized to execute code in response to events without the need to provision or manage servers explicitly. By adopting a serverless architecture, the healthcare system can benefit from fine-grained resource allocation, paying only for the compute time consumed by each function execution. This serverless approach eliminates the need for idle server capacity and reduces operational overhead, resulting in cost savings and improved resource efficiency.
- 3. Cost Monitoring and Optimization Tools:** AWS Cost Explorer is employed to monitor cloud resource usage, analyze cost trends, and identify opportunities for optimization. Cost Explorer provides insights into resource utilization, cost drivers, and spending patterns, enabling administrators to identify areas of inefficiency and implement cost-saving measures proactively. By monitoring resource usage and expenditure regularly, healthcare organizations can optimize their cloud spending, identify cost-saving opportunities, and ensure that cloud resources are aligned with business objectives and budgetary constraints.
- 4. Reserved Instances and Spot Instances:** Reserved Instances and Spot Instances are utilized to optimize pricing models and reduce cloud infrastructure costs. Reserved Instances offer discounted pricing for long-term commitments, allowing healthcare organizations to reserve capacity in advance and benefit from lower hourly rates compared to On-Demand Instances. Spot Instances enable healthcare organizations to bid on unused EC2 capacity at significantly lower prices, providing cost-effective alternatives for non-critical workloads or batch processing tasks. By strategically leveraging Reserved Instances and Spot Instances, healthcare organizations can reduce their cloud spending while maintaining high-quality service delivery.
- 5. Performance Optimization:** The architecture emphasizes performance optimization to ensure efficient resource utilization and minimize unnecessary costs. Strategies such as caching, data compression are employed to improve system responsiveness, reduce latency, and optimize resource consumption. By optimizing performance, the architecture maximizes the value derived from cloud resources while minimizing operational costs and ensuring high-quality healthcare services.

Overall, the proposed architecture implements a comprehensive set of resource optimization strategies and cost monitoring mechanisms to balance cost-effectiveness with the delivery of high-quality healthcare services. By leveraging auto-scaling, serverless computing, cost monitoring tools, pricing models, and performance optimization techniques, the architecture optimizes cloud resources efficiently, minimizes costs, and ensures ongoing cost monitoring to align cloud spending with business objectives and budgetary constraints.

Adaptability to Technological Advancements:

The architecture is designed with a strong emphasis on adaptability to emerging technologies and healthcare trends, ensuring that it can seamlessly integrate new AI algorithms and advancements in remote patient monitoring devices. Here's how the architecture achieves adaptability to technological advancements:

- 1. Modular Design:** The architecture is built with a modular design that allows for the easy integration of new technologies and services. Each component of the architecture is designed to operate independently, enabling healthcare organizations to add or replace modules as needed without disrupting the overall system. This modular approach ensures flexibility and scalability, allowing the architecture to evolve in response to emerging technologies and healthcare trends.
- 2. API-Based Integration:** The architecture leverages API-based integration to facilitate interoperability with third-party systems and services. By exposing well-defined APIs, the architecture enables seamless integration with new AI algorithms, remote patient monitoring devices, and other emerging technologies. Healthcare organizations can easily integrate new services into the architecture by developing custom APIs or leveraging existing standards such as HL7 (Health Level Seven) for healthcare data exchange. This API-based approach ensures compatibility and interoperability with a wide range of technologies, enabling the architecture to adapt to emerging trends quickly and efficiently.
- 3. Scalable Infrastructure:** The architecture is built on a scalable infrastructure that can accommodate the growing demands of emerging technologies and healthcare trends. Scalable cloud services such as AWS Lambda, Amazon DynamoDB, and Amazon S3 provide the foundation for the architecture, allowing healthcare organizations to scale resources up or down based on changing requirements. This scalability ensures that the architecture can support the increased computational and storage needs associated with new AI algorithms and remote patient monitoring devices, enabling healthcare organizations to leverage these technologies effectively without encountering scalability constraints.
- 4. Continuous Innovation:** The architecture is designed to embrace a culture of continuous innovation, enabling healthcare organizations to stay abreast of the latest technological advancements and healthcare trends. By fostering a collaborative environment and encouraging experimentation, the architecture empowers healthcare professionals to explore new technologies, evaluate their potential impact on patient care, and incorporate them into the architecture as appropriate. This culture of innovation ensures that the architecture remains adaptable and responsive to emerging technologies and healthcare trends, allowing healthcare organizations to deliver the highest quality of care to patients while leveraging the latest advancements in healthcare technology.

Overall, the architecture is highly adaptable to technological advancements and healthcare trends, providing healthcare organizations with the flexibility, scalability, and agility needed to integrate new AI algorithms, remote patient monitoring devices, and other emerging technologies seamlessly. By adopting a modular design, API-based integration, scalable infrastructure, and a culture of continuous innovation, the architecture ensures that healthcare organizations can stay ahead of the curve and deliver innovative healthcare solutions that meet the evolving needs of patients and healthcare professionals.

User Training and Support:

User training and support are integral components of the project, ensuring that healthcare professionals and patients can effectively utilize the system and its features. Here's how user training and support are addressed within the architecture:

- 1. Comprehensive Training Materials:** The project includes the development of comprehensive training materials tailored to the needs of healthcare professionals and patients. These materials cover all aspects of the system, including navigation, data entry, feature utilization, and troubleshooting. Training materials are available in various formats, including written guides, video tutorials, and interactive demonstrations, to accommodate different learning preferences and technical proficiencies. Features such as Lambda for executing code in response to events and Amazon DynamoDB for fast and reliable access to electronic health records are explained in detail within the training materials.
- 2. Onboarding Workshops and Sessions:** To facilitate the adoption of the system, onboarding workshops and training sessions are conducted for healthcare professionals and administrative staff. These sessions provide hands-on training and guidance on using the system effectively, allowing users to familiarize themselves with its features and functionalities in a supportive environment. Trainers are available to answer questions, address concerns, and provide personalized assistance to users as needed. Amazon Cognito is utilized for secure user authentication and authorization during these training sessions.
- 3. User Support Channels:** Multiple user support channels are established to provide ongoing assistance and support to healthcare professionals and patients. These channels include dedicated help desks, support forums, and interactive chatbots, enabling users to seek help and troubleshoot issues in real-time. Support personnel are trained to provide prompt and courteous assistance, ensuring that users receive the help they need to overcome any challenges they encounter while using the system. Amazon S3 is used for storing and retrieving training materials and support documentation.
- 4. Feedback Mechanisms:** Feedback mechanisms are implemented to gather input from users and identify areas for improvement in the system and its training materials. Surveys, feedback forms, and user interviews are used to collect feedback from healthcare professionals and patients, allowing project stakeholders to understand user needs, preferences, and pain points. This feedback is used to refine training materials, enhance user support services, and improve the overall user experience of the system. AWS CloudTrail is employed to log and monitor user activity, aiding in the collection of feedback data.
- 5. Continuous Training and Skill Development:** Recognizing that ongoing training and skill development are essential for maximizing user proficiency and satisfaction, the project includes provisions for continuous learning opportunities. Regular training sessions, refresher courses, and skill-building workshops are offered to healthcare professionals and administrative staff, enabling them to stay updated on new features, best practices, and industry trends. By investing in continuous training and skill development, the project ensures that users remain competent and confident in using the system effectively to deliver high-quality healthcare services.

Overall, user training and support are prioritized within the project to ensure that healthcare professionals and patients receive the assistance they need to leverage the system's features effectively. Through comprehensive training materials, onboarding workshops, user support channels, feedback mechanisms, and continuous training opportunities, the project aims to empower users to maximize their productivity, efficiency, and satisfaction while using the system to deliver and receive healthcare services.

Disaster Recovery and Contingency Planning:

- 1. Comprehensive Disaster Recovery Plan:** The architecture includes a comprehensive disaster recovery plan that outlines procedures and protocols for mitigating the impact of unforeseen events on the system's operations. Leveraging AWS Backup, this plan identifies potential risks and vulnerabilities, such as hardware failures, software glitches, natural disasters, or cyberattacks, and defines strategies for minimizing downtime and data loss in the event of such incidents. It includes detailed steps for restoring services, recovering data, and resuming normal operations as quickly as possible.
- 2. Regular Testing and Validation:** The disaster recovery plan is regularly tested and validated to ensure its effectiveness and reliability. Scheduled drills and simulations are conducted to simulate various disaster scenarios and evaluate the system's response capabilities. These tests involve simulating system failures, data corruption, or network outages and assessing the effectiveness of backup and recovery procedures in restoring services and data integrity.
- 3. Redundant Backup Systems:** To ensure data integrity and operational continuity, the architecture incorporates redundant backup systems and data replication mechanisms. Critical data, including patient records, medical images, and system configurations, are regularly backed up to multiple geographically distributed locations to minimize the risk of data loss due to localized disasters or hardware failures. Data replication mechanisms ensure that changes made to primary data stores are replicated to backup systems in near real-time, ensuring data consistency and availability across all backup locations.
- 4. Automated Monitoring and Alerting:** The architecture includes automated monitoring and alerting mechanisms that continuously monitor system health, performance, and security parameters. Any deviations from normal operating conditions trigger immediate alerts to designated personnel, enabling rapid response and intervention to prevent or mitigate potential disruptions. These monitoring and alerting mechanisms provide real-time visibility into the system's status and enable proactive intervention to address issues before they escalate into full-blown disasters.
- 5. Regular Maintenance and Updates:** To maintain the resilience and reliability of the system, regular maintenance and updates are performed to address security vulnerabilities, software bugs, and performance issues. These updates are applied systematically following best practices for change management and version control to minimize the risk of introducing new issues or disruptions. By staying current with software patches, firmware updates, and security fixes, the architecture ensures that the system remains secure, stable, and capable of withstanding potential threats or failures.

In summary, the disaster recovery and contingency planning measures implemented within the architecture are comprehensive, tested, and continuously optimized to ensure data integrity and operational continuity in the face of unexpected events or system failures. Through regular testing, redundant backup systems, automated monitoring, and proactive maintenance, the architecture is equipped to withstand various disaster scenarios and maintain uninterrupted service delivery to healthcare professionals and patients.

IMPLEMENTATION AND DEVELOPMENT PROCESS

Development Environment Setup:

The project begins with setting up a development environment tailored to the requirements of cloud-based healthcare application development. Development environments are provisioned using Infrastructure as Code (IaC) tools such as AWS CloudFormation or AWS CDK (Cloud Development Kit), ensuring consistency and reproducibility across development environments. Development environments include all necessary development tools, libraries, and dependencies, allowing developers to efficiently build, test, and deploy application components.

Technology Stack and Frameworks:

The project utilizes a carefully selected technology stack and frameworks optimized for developing cloud-native healthcare applications. The technology stack includes serverless computing services such as AWS Lambda for executing code without provisioning or managing servers, Amazon API Gateway for creating, publishing, maintaining, monitoring, and securing APIs, and AWS DynamoDB for NoSQL database storage. Frameworks such as React.js and Node.js are used for frontend and backend development, respectively, facilitating rapid development and scalability.

Implementation Challenges and Solutions:

Throughout the implementation process, various challenges may arise, requiring innovative solutions to overcome them. Common implementation challenges include integrating disparate healthcare systems, ensuring data security and compliance with healthcare regulations, optimizing application performance and scalability, and managing resources efficiently. To address these challenges, the project adopts a proactive approach, leveraging best practices, architectural patterns, and cloud-native services. For example, challenges related to data security and compliance are mitigated by implementing encryption mechanisms, access controls, and audit trails using AWS services such as AWS Key Management Service (KMS), AWS Identity and Access Management (IAM), and AWS CloudTrail. Additionally, performance and scalability challenges are addressed by employing serverless architectures, horizontal scaling, and caching strategies, optimizing resource utilization and enhancing application responsiveness.

By systematically addressing development environment setup, selecting appropriate technology stacks and frameworks, and proactively tackling implementation challenges, the project ensures the successful development and deployment of a robust cloud-based healthcare management system.

TESTING AND EVALUATION

Testing Strategies and Scenarios:

The project employs comprehensive testing strategies and scenarios to ensure the reliability, functionality, and security of the cloud-based healthcare management system. Testing strategies include unit testing, integration testing, system testing, and acceptance testing. Unit testing validates the functionality of individual components or modules, ensuring they perform as expected. Integration testing verifies the interaction between different components, ensuring seamless communication and data flow. System testing evaluates the system as a whole, validating end-to-end functionality and performance. Acceptance testing involves stakeholders testing the system against predefined criteria to determine if it meets their requirements. Testing scenarios cover various use cases, edge cases, and error conditions, ensuring the system behaves correctly under different circumstances.

Performance Testing:

Performance testing assesses the system's responsiveness, scalability, and reliability under varying workloads. Performance tests simulate real-world usage scenarios to evaluate how the system performs under normal and peak load conditions. Key performance metrics include response time, throughput, resource utilization, and scalability. Performance testing identifies bottlenecks, scalability issues, and areas for optimization, allowing developers to fine-tune the system for optimal performance.

Usability Testing:

Usability testing evaluates the user interface (UI) and user experience (UX) of the cloud-based healthcare management system. Usability tests assess how easily users can navigate the system, perform tasks, and access information. Usability testing involves real users interacting with the system, providing feedback on usability, accessibility, and overall user satisfaction. Usability tests may include task-based testing, heuristic evaluation, and user interviews. The goal of usability testing is to identify usability issues, improve the user interface design, and enhance the overall user experience of the system.

CONCLUSION

Summary of Project Outcomes:

The project successfully achieved its objectives of designing and implementing a cloud-based healthcare management system on the AWS platform. The architecture proposal addressed key challenges faced by healthcare organizations, including interoperability, scalability, performance optimization, cost efficiency, security, and data privacy. By leveraging AWS services such as Elastic Load Balancer, EC2, Amazon DynamoDB, and others, the system demonstrated robustness, reliability, and scalability in managing patient data, facilitating telemedicine services, and ensuring compliance with healthcare regulations.

The implementation of the proposed solution on AWS showcased the seamless integration of various components, providing healthcare professionals and patients with an intuitive, accessible, and secure platform for managing healthcare services. The project's outcomes significantly contribute to the digitization and enhancement of healthcare services, enabling efficient collaboration among healthcare professionals, improving patient care, and ensuring the confidentiality and integrity of healthcare information.

Limitations and Future Work:

While the project has successfully achieved its primary objectives, there are certain limitations and areas for future improvement:

1. Integration Complexity: Integrating with existing healthcare systems can be complex and may require further customization and compatibility testing. This process needs careful consideration to ensure seamless interoperability with legacy systems.
2. Cost Management and Scalability: Continuous monitoring and optimization of cloud resources are crucial to ensure cost efficiency and scalability over time. Implementing cost management strategies and scalability best practices will be essential as the application grows.
3. Feature Limitations: Certain features, such as AWS Chime for telecommunication, have not been implemented due to current project requirements. As a result, AWS Lambda functions have not been utilized. Additionally, services like AWS CloudTrail and Amazon Redshift have not been integrated due to potential charges and current project needs.

Future Work:

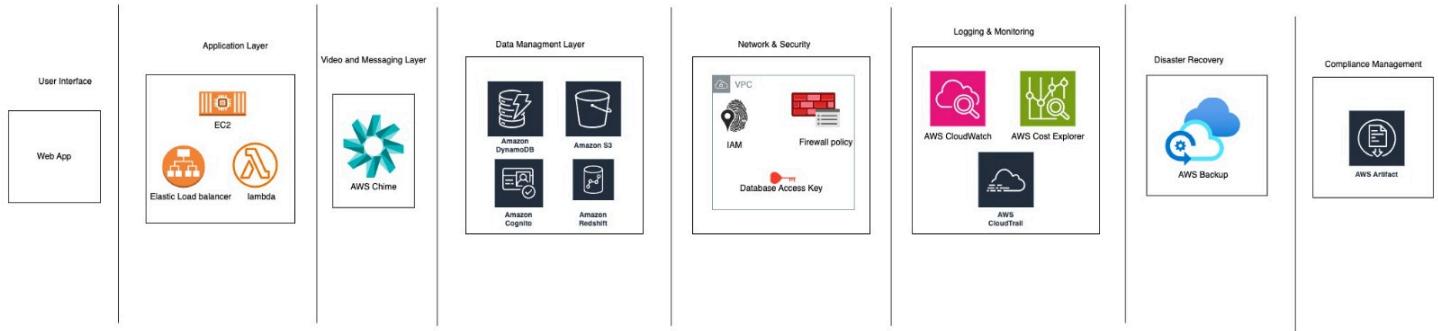
1. Enhanced Analytics Capabilities: Future work could focus on enhancing AI-driven analytics capabilities to derive valuable data-driven insights from healthcare data. Advanced analytics can help in improving patient outcomes, optimizing resource allocation, and identifying trends and patterns in healthcare data.
2. Support for Emerging Technologies: As healthcare technology continues to evolve, future iterations of the project could expand support for emerging technologies such as wearables, IoT devices, and remote patient monitoring systems. Integrating these technologies can enhance patient engagement, enable proactive healthcare interventions, and facilitate personalized care delivery.
3. Advanced Security Measures: Integrating advanced security measures is essential to mitigate evolving cybersecurity threats and safeguard sensitive healthcare data. Future work could focus on implementing features such as data encryption, access controls, and threat detection mechanisms to enhance the overall security posture of the application.
4. Usability Testing and User Feedback: Continuous user feedback and usability testing will be valuable for iteratively improving the user experience and addressing evolving user needs in the healthcare domain. Incorporating user feedback into the development process can help in prioritizing features, identifying pain points, and ensuring that the application meets the needs of healthcare professionals and patients alike.

By addressing these limitations and focusing on future enhancements, the project can continue to evolve and provide value in the ever-changing landscape of healthcare technology.

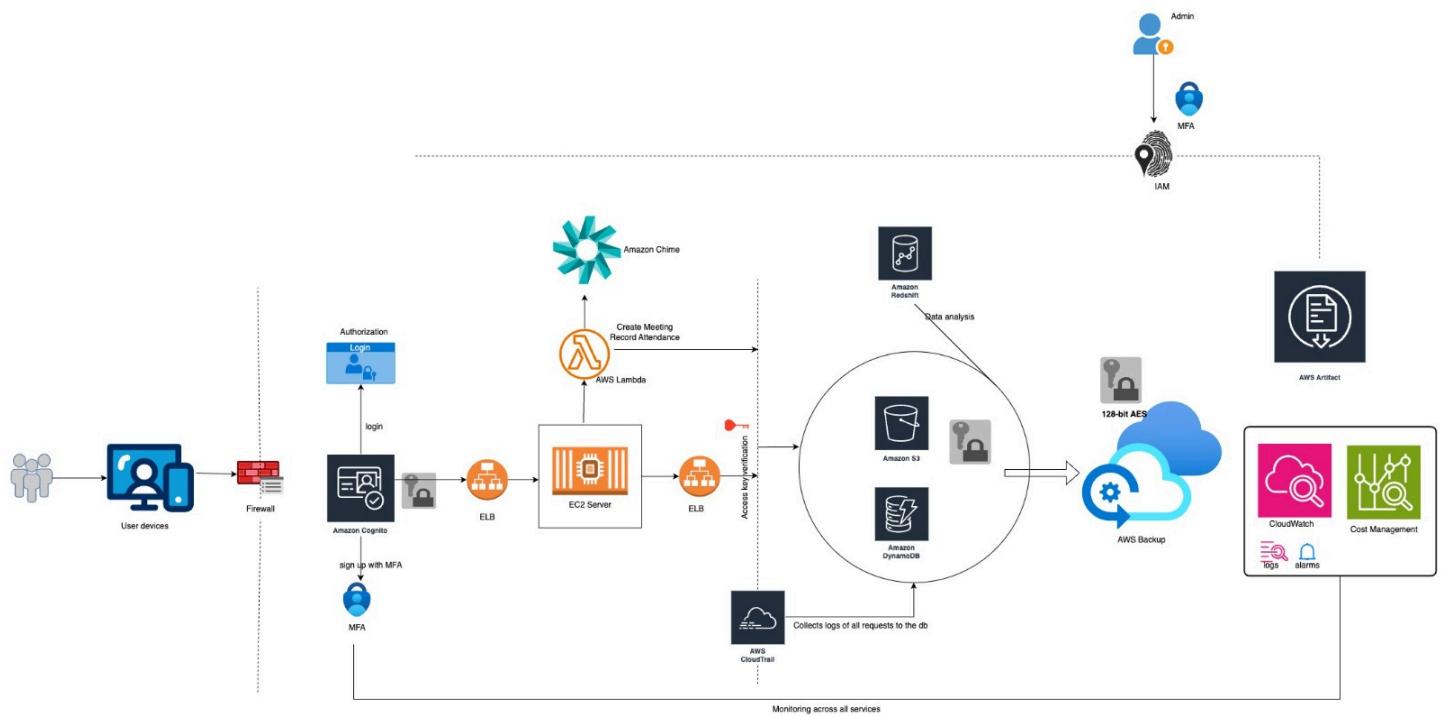
APPENDICES

Architectural Diagrams:

Technologies & Layers:



Project Architecture:



1. User Devices:

This represents the clients or devices used by users (e.g., patients, healthcare professionals) to access the system.

2. Authentication and Authorization:

The users first go through an authentication process, likely involving a login mechanism.

AWS Lambda is used to handle the authentication and authorization logic.

An Amazon Cognito service (not shown) could be used for user management and authentication.

3. Amazon Chime:

This AWS service provides secure video and messaging capabilities for telemedicine consultations and communication between healthcare providers and patients.

4. Application Layer:

An Elastic Load Balancer (ELB) distributes incoming traffic across multiple EC2 instances for high availability and scalability.

The EC2 instances host the web application, which serves as the user interface for the healthcare system.

Amazon Cognito (not shown) could handle user authentication and authorization for the web application.

5. Data Management:

Amazon S3 is used for object storage, likely storing medical images, documents, and other healthcare data.

Amazon DynamoDB, a NoSQL database, could store and retrieve patient data, electronic health records, and other structured data.

Amazon Redshift, a data warehousing solution, could be used for data analytics and reporting on healthcare data.

6. Data Processing:

AWS Lambda functions are used for serverless data processing tasks, such as creating meeting records and performing data analysis.

7. Networking and Security:

The entire system is deployed within an Amazon Virtual Private Cloud (VPC) for network isolation and security.

AWS Identity and Access Management (IAM) controls access to AWS resources and enforces permissions.

A firewall policy is in place to regulate network traffic and protect the system from unauthorized access.

- Multi-Factor Authentication (MFA) is implemented for additional security during user authentication.

8. Monitoring and Logging:

AWS CloudTrail logs API calls made to AWS services, providing an audit trail for security and compliance purposes.

AWS CloudWatch monitors the system's performance, collects logs, and triggers alarms if necessary.

AWS Cost Management tools, such as AWS Cost Explorer, help monitor and optimize cloud costs.

9. Backup and Disaster Recovery:

AWS Backup provides backup and restore capabilities for healthcare data, ensuring data availability and disaster recovery.

10. Compliance:

AWS Artifact is a resource that provides access to AWS compliance reports and online agreements, helping the healthcare system maintain regulatory compliance.

Overall, this architecture leverages various AWS services to build a secure, scalable, and compliant healthcare system, enabling telemedicine services, electronic health record management, data analytics, and efficient resource utilization while prioritizing data security and privacy.

Code Snippets and Configuration Files:

- Amazon Eventbridge:

The screenshot shows the AWS EventBridge Rules interface. At the top, a green success banner displays the message "Rule Backup-Rules was created successfully". Below the banner, the navigation path is "Amazon EventBridge > Rules". The main section is titled "Rules" and contains a sub-section "Select event bus". Under "Event bus", it says "Select or enter event bus name" and shows a dropdown menu with "default" selected. The main table area is titled "Rules (1)". It includes a header row with columns: Name, Status, Type, ARN, and Description. A single rule named "Backup-Rules" is listed, showing it is Enabled, Standard type, with ARN arn:aws:events:eu-west-3:975050211573:rule/Backup-Rules, and a description "Rules for backup data". The bottom of the page includes standard AWS footer links: "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

Name	Status	Type	ARN	Description
Backup-Rules	Enabled	Standard	arn:aws:events:eu-west-3:975050211573:rule/Backup-Rules	Rules for backup data

Backup Rules Created

Step 1

Define rule detail

Step 2

[Build event pattern](#)

Step 3

[Select target\(s\)](#)

Step 4 - optional

Configure tags

Step 5

Review and create

Define rule detail Info

Rule detail

Name

Backup-Rules

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-_.

Description - optional

Rules for backup data

Event bus Info

Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

default ▾

Enable the rule on the selected event bus

Rule type Info

Rule with an event pattern

A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

Schedule

A rule that runs on a schedule

Cancel

Next

Define Rule Detail

Amazon EventBridge > Rules > Create rule

Step 1
[Define rule detail](#)

Step 2
Build event pattern

Step 3
[Select target\(s\)](#)

Step 4 - optional
Configure tags

Step 5
Review and create

Build event pattern Info

Event source

Event source
Select the event source from which events are sent.

AWS events or EventBridge partner events
Events sent from AWS services or EventBridge partners.

Other
Custom events or events sent from more than one source, e.g. events from AWS services and partners.

All events
All events sent to your account.

Sample event - optional
You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.

You can reference the sample event when you write the event pattern, or use the sample event to test if it matches the event pattern. Find a sample event, enter your own, or edit a sample event below. [Learn more about the required fields in a sample event.](#)

Sample event type

AWS events EventBridge partner events Enter my own

Sample events
Filter by event source and type or by keyword.

 CloudShell [Feedback](#) © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#)

Event pattern 1

The screenshot shows the AWS EventBridge console interface for creating a new event pattern. The top section, titled 'Method', offers three options: 'Use schema' (radio button), 'Use pattern form' (radio button, selected), and 'Custom pattern (JSON editor)'. The 'Use pattern form' section includes a description and a preview area showing a JSON event pattern. The preview area contains the following JSON code:

```
1 {
2   "source": ["aws.backup"],
3   "detail-type": ["Backup Job State Change"]
4 }
```

Below the method selection, the 'Event pattern' section is titled 'Event pattern Info'. It contains three dropdown menus: 'Event source' (set to 'AWS services'), 'AWS service' (set to 'Backup'), and 'Event type' (set to 'Backup Job State Change'). To the right of these dropdowns is a 'Event pattern' field containing the JSON code above. At the bottom of this section are three buttons: 'Copy', 'Test pattern', and 'Edit pattern'.

Event pattern 2



Step 1

[Define rule detail](#)

Step 2

[Build event pattern](#)

Step 3

Select target(s)

Step 4 - optional

[Configure tags](#)

Step 5

[Review and create](#)

Select target(s)



Permissions

Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

Target 1

Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

- EventBridge event bus
- EventBridge API destination
- AWS service

Select a target | [Info](#)

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)

SNS topic ▾

Topic

BackupNotifications ▾ C

► Additional settings

[Add another target](#)

[Cancel](#)

[Skip to Review and create](#)

[Previous](#)

[Next](#)

Select SNS target

- AWS Backup:

General

Resource assignment name

BackupAssignment

Resource assignment name is case sensitive. Must contain from 1 to 50 alphanumeric or '-_.' characters.

IAM role [Info](#)

AWS Backup will assume this IAM role when creating and managing recovery points on your behalf.

Default role
If the AWS Backup default role is not present, one will be created for you with the correct permissions.

Choose an IAM role

Resource selection [Info](#)

Assign resources to this Backup plan using tags and resource IDs.

1. Define resource selection [Info](#)
Protect all resources or specify resources by type or ID.

Include all resource types
Protect all resource types that are enabled in your account.

Include specific resource types
Choose resources by type or specify individual resources by ID.

2. Refine selection using tags - *optional* [Info](#)
Filter resources by tags. For multiple tags, resources will only be assigned to the backup plan if they satisfy all tag conditions.

Resource Assignment Name

Start options

Backup plan options

[Info](#)

Start with a template

Create a Backup plan based on a template provided by AWS Backup.

Build a new plan

Configure a new Backup plan from scratch.

Define a plan using JSON

Modify the JSON expression of an existing backup plan or create a new expression.

Templates

Choose a template plan with existing rules.

Daily-Monthly-1yr-Retention



Backup plan name

Backup plan name is case sensitive. Must contain from 1 to 50 alphanumeric or '-_.' characters.

► **Tags added to backup plan - optional**

Backup rules

[Info](#)

[Add backup rule](#)

[Delete](#)

[Edit](#)

Backup rules specify the backup schedule, backup window, and lifecycle rules.

Name	Backup vault
DailyBackups	Default

Create Back Up Plan

backup plan later. The cost depends on your configurations.



Backup rule configuration [Info](#)

Schedule

Backup rule name

Backup rule name is case sensitive. Must contain from 1 to 50 alphanumeric or '-' characters.

Backup vault | [Info](#)

[C](#) [Create new Backup vault](#)

Backup frequency | [Info](#)

Backup window [Info](#)

Start time

Specify the time of day the backups will start. For **hourly** frequency, start time is the time the first backup is taken in a day. Where applicable, time will adjust to daylight savings time so that it retains the same local time all year.

:

Start within [Info](#)

Specify period of time in which the backup plan starts if it doesn't start at the specified time.

Create backup rule



⌚ "HealthCareBackupVault" was deleted successfully.

[AWS Backup](#) > [Backup vaults](#) > Create backup vault

Create backup vault [Info](#)

General

Backup vault name

HealthCareBackupVault

Backup vault name is case sensitive. Must contain from 2 to 50 alphanumeric or '-' characters.

Encryption key [Info](#)

(default) aws/backup

Description

Default key that protects
my Backup data when no
other key is defined

Account

This account
(975050211573)

Key ID

7c401fa7-1bfd-49b0-
8b81-8b1ab399a24b

Status

Enabled

Backup vault tags - optional

Tags specified here help organize and track your Backup vault

No tags associated with this vault.

[Add new tag](#)

You can add up to 50 more tags

[Create Backup Vault](#)



✓ "HealthCareBackupVault" was deleted successfully.

Access policy details [Info](#)

Policy JSON can be edited directly below. [Learn more](#)

```
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Principal": {
7                 "AWS": "arn:aws:iam::975050211573:role/service-role/AWSBackupDefaultServiceRole"
8             },
9             "Action": [
10                "backup:StartBackupJob",
11                "backup:StartRestoreJob",
12                "backup:DeleteBackupVault",
13                "backup:DeleteRecoveryPoint",
14                "backup:GetBackupVaultAccessPolicy",
15                "backup:PutBackupVaultAccessPolicy"
16            ],
17            "Resource": "arn:aws:backup:eu-west-3:975050211573:backup-vault:HealthCareBackupVault"
18        }
19    ]
20}
21
```

You can use a [policy generator](#) to build policy permissions.

CloudShell [Feedback](#)

Add policies to backup vault

[AWS Backup](#) > [Jobs](#) > 9e1923a3-bcca-4de6-b83c-0a0a42c0dd62

Backup - 9e1923a3-bcca-4de6-b83c-0a0a42c0dd62



[Stop backup job](#)

In backup job details, you can access records of your scheduled or on-demand backups.

Details

Recovery point ARN
 arn:aws:backup:eu-west-3:975050211573:recovery-point:b8c062ce-c09a-482d-bbaa-6327f3c7a2b0

Status
 Pending

Resource name
Users

Resource ID
table/Users

Resource type
DynamoDB

Creation date
May 5, 2024, 11:39:11 (UTC+02:00)

Start by
May 5, 2024, 12:39:11 (UTC+02:00)

IAM role
[Default role](#)

6_Backup job for dynamoDB table users

Create on-demand backup [Info](#)

Settings

Resource type

DynamoDB

Table name

Users



Backup window

Create backup now

Starts within 1 hour.

Customize backup window

Cold storage [Info](#)

Move backups from warm to cold storage

Available for CloudFormation, DynamoDB with advanced features, EFS, SAP HANA, Timestream, and VMware virtual machines. Some resource types convert incremental backups to full backups. Requires at least 90 days of retention.

Time in warm storage [Info](#)

8

Days



Recommended minimum is 8 days

Total retention period [Info](#)

Tell AWS Backup how long to store your backups.

98

Days



Total retention (days)

CloudShell

Feedback

Create on demand backup to test for the users table in dynamoDb

- AWS CloudWatch:

The screenshot shows the AWS CloudWatch Alarms page. The left sidebar contains navigation links for Favorites and recents, Dashboards, Alarms (16 OK, 48 Pending, 1 In alarm), Logs, Metrics, X-Ray traces, Events, Application Signals, Network monitoring, Insights, and Settings. The main content area displays a table of 48 alarms. The table has columns for Name, State, Last state update (UTC), Conditions, and Actions. Each row represents an alarm with a name like 'TargetTracking-table/Online_Appointm...nt-' and a status like 'OK'. The 'Actions' column indicates if actions are enabled or disabled.

Name	State	Last state update (UTC)	Conditions	Actions
TargetTracking-table/Online_Appointm...nt-	OK	2024-05-04 15:27:57	ProvisionedReadCapacityUnits > 1 for 3 datapoints within 15 minutes	Actions enabled
ProvisionedCapacityHig...h-7413b39a-2db1-415e-afbd-Ofab1882a5e4	OK	2024-05-04 15:27:34	ProvisionedReadCapacityUnits > 1 for 3 datapoints within 15 minutes	Actions enabled
TargetTracking-table/Rooms-ProvisionedCapacityHig...h-6531d2d1-caab-4dc0-83f3-908213350c15	OK	2024-05-04 15:27:34	ProvisionedReadCapacityUnits > 1 for 3 datapoints within 15 minutes	Actions enabled
TargetTracking-table/Appointment-ProvisionedCapacityHig...h-db52c034-0a91-45f1-97fd-9ccee19cced9	OK	2024-05-04 15:27:32	ProvisionedReadCapacityUnits > 1 for 3 datapoints within 15 minutes	Actions enabled
TargetTracking-table/Medical_Specialties-ProvisionedCapacityHig...h-b2c3e7a6-ce00-41af-86ad-3d2e44d4353d	OK	2024-05-04 15:27:29	ProvisionedReadCapacityUnits > 1 for 3 datapoints within 15 minutes	Actions enabled
TargetTracking-table/Users-				

- AWS SNS:

The screenshot shows the AWS SNS 'Create subscription' interface. At the top, there's a banner about message archiving support for FIFO topics. Below it, the navigation path is 'Amazon SNS > Subscriptions > Create subscription'. The main form has a 'Details' tab selected. It contains fields for 'Topic ARN' (set to 'arn:aws:sns:eu-west-3:975050211573:BackupNotifications'), 'Protocol' (set to 'Email'), and 'Endpoint' (set to 'khanh-nguyen.tran@epita.fr'). A note below the endpoint says 'After your subscription is created, you must confirm it.' There's also an optional section for 'Subscription filter policy'.

Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Amazon SNS > Subscriptions > Create subscription

Create subscription

Details

Topic ARN
arn:aws:sns:eu-west-3:975050211573:BackupNotifications

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
khanh-nguyen.tran@epita.fr

ⓘ After your subscription is created, you must confirm it. [Info](#)

► **Subscription filter policy - optional** [Info](#)
This policy filters the messages that a subscriber receives.

Create subscriptions to notify the backup plan to owner

Create topic

Details

Type [Info](#)
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional* [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

Maximum 100 characters.

► **Encryption - *optional***
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

[Create Topic](#)

▼ Access policy - optional Info

This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

Choose method

Basic

Use simple criteria to define a basic access policy.

Advanced

Use a JSON object to define an advanced access policy.

Publishers

Specify who can publish messages to the topic.

Only the topic owner

Only the owner of the topic can publish to the topic

JSON preview

```
"Version": "2008-10-17",
"Id": "__default_policy_ID",
"Statement": [
    {
        "Sid": "__default_statement_ID",
        "Effect": "Allow",
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "SNS:Publish",
            "SNS:RemovePermission",
            "SNS:SetTopicAttributes",
            "SNS:DeleteTopic"
        ]
    }
]
```

Subscribers

Specify who can subscribe to this topic.

Only the topic owner

Only the owner of the topic can subscribe to the topic

Add access policies

- AWS Cognito:

Step 1
Configure sign-in experience

Step 2
Configure security requirements

Step 3
Configure sign-up experience

Step 4
Configure message delivery

Step 5
Integrate your app

Step 6
Review and create

Configure sign-in experience Info

Your app users can sign in to your user pool with a user name and password, or sign in with a third-party identity provider.

Authentication providers

Configure the providers that are available to users when they sign in.

Provider types

Choose whether users will sign in to your Cognito user pool, a federated identity provider, or both. Amazon Cognito has different pricing for federated users and user pool users. [Learn more about pricing](#)

Cognito user pool
Users can sign in using their email address, phone number, or user name. User attributes, group memberships, and security settings will be stored and configured in your user pool.

Federated identity providers
Users can sign in using credentials from social identity providers like Facebook, Google, Amazon, and Apple; or using credentials from external directories through SAML or Open ID Connect. You can manage user attribute mappings and security for federated users in your user pool.

Cognito user pool sign-in options Info

Choose the attributes in your user pool that are used to sign in. If you select only one attribute, or you select a user name and at least one other attribute, your user can sign in with all of the selected options. If you select only phone number and email, your user will be prompted to select one of the two sign-in options when they sign up.

User name
 Email
 Phone number

User name requirements

Allow users to sign in with a preferred user name
 Make user name case sensitive

Step 1

[Configure sign-in experience](#)

Step 2

Configure security requirements

Step 3

Configure sign-up experience

Step 4

Configure message delivery

Step 5

Integrate your app

Step 6

Review and create

Configure security requirements Info

Set up a strong password requirement in addition to multi-factor authentication to protect your app users from accidentally compromising their credentials.

Password policy Info

Create a password policy to define the length and complexity of the passwords your users can set.

Password policy mode | [Info](#)

Cognito defaults

Use default password requirements.

Custom

Use password requirements that you define.

Password minimum length

8 character(s)

Password requirements

Contains at least 1 number

Contains at least 1 special character

Contains at least 1 uppercase letter

Contains at least 1 lowercase letter

Temporary passwords set by administrators expire in

7 day(s)

Multi-factor authentication

Configure Security Requirement

The screenshot shows the AWS Cognito User Pools configuration interface. It includes sections for MFA enforcement and User account recovery.

MFA enforcement

Require MFA - Recommended
Users must provide an additional authentication factor when signing in.

Optional MFA
Users can sign in with a single authentication factor, and can choose to add additional authentication factors.

No MFA
Users can only sign in with a single authentication factor. This is the least secure option.

MFA methods

Authenticator apps
Users can authenticate with a TOTP from an authenticator app such as Authy or Google Authenticator.

SMS message
Users can authenticate with a code sent by SMS message to a verified phone number. SMS messages are charged separately by Amazon SNS. [Learn more about pricing](#)

User account recovery

Configure how users will recover their account when they forget their password. Recipient message and data rates apply.

Self-service account recovery

Enable self-service account recovery - Recommended
Allow forgot-password operations in your user pool. In the hosted UI sign-in page, a "Forgot your password?" link is displayed. When this feature is not enabled, administrators reset passwords with the Cognito API.

Delivery method for user account recovery messages

Select how your user pool will deliver messages when users request an account recovery code. SMS messages are charged separately by Amazon SNS. Email messages are charged separately by Amazon SES. [Learn more about pricing](#)

Email only

SMS only

Configure Security Requirement



Step 1

[Configure sign-in experience](#)

Step 2

[Configure security requirements](#)

Step 3

[Configure sign-up experience](#)

Step 4

Configure message delivery

Step 5

Integrate your app

Step 6

Review and create

Configure sign-up experience [Info](#)

Determine how new users will verify their identities when signing up and which attributes should be required or optional during the user sign-up flow.

Self-service sign-up [Info](#)

Choose whether new users of your app can register for an account themselves.

Self-registration [Info](#)

Enable self-registration
Display a "Sign up" link on the sign-in page in the hosted UI, and allow the use of public APIs to create new user accounts. When this feature is not enabled, federation and administrative API operations create user profiles.

 If you activate user sign-up in your user pool, anyone on the internet can sign up for an account and sign in to your apps. Don't enable self-registration in your user pool until you want to open your app to public sign-up.
[Learn more](#) 

Attribute verification and user account confirmation

Choose between Cognito-assisted and self-managed user attribute verification and account confirmation. Only verified attributes can be used for sign-in, account recovery, and MFA. A user account must be confirmed either by attribute verification, or user pool administrator confirmation, before a user is allowed to sign in.

Cognito-assisted verification and confirmation [Info](#)

Automatically send

© 2024, Amazon Web Services, Inc. or its affiliates

Signup Experience



Attribute verification and user account confirmation

Choose between Cognito-assisted and self-managed user attribute verification and account confirmation. Only verified attributes can be used for sign-in, account recovery, and MFA. A user account must be confirmed either by attribute verification, or user pool administrator confirmation, before a user is allowed to sign in.

Cognito-assisted verification and confirmation [Info](#)

Automatically send

Allow Cognito to automatically send messages to verify and confirm - Recommended

Amazon Cognito sends a verification message with a code that the user must enter. For new users, this will verify the attribute and confirm their account.

Don't automatically send messages

Amazon Cognito doesn't send messages to users who add or change an attribute. Update attributes and confirm users with administrative API operations or Lambda triggers.

Attributes to verify [Info](#)

Choose the user contact attribute that Cognito will send a verification message to. Recipient message and data rates apply when you use SMS.

Send SMS message, verify phone number

Verify with SMS to allow users to use their phone number for sign-in, MFA, and account recovery. SMS messages are charged separately by Amazon SNS.

Send email message, verify email address

Verify with email to allow users to use their email address for sign-in, MFA, and account recovery. Email messages are charged separately by Amazon SES.

Send SMS message if phone number is available, otherwise send email message

You must build custom code when you want to verify both email and phone numbers at user account creation.

Verifying attribute changes [Info](#)

Keep original attribute value active when an update is pending - Recommended

When you update the value of an email or phone number attribute, your user must verify the new value. Until they verify the new value, they can receive messages and sign in with the original value. If you don't turn on this feature, your user can't sign in with that attribute before they verify the new value.

Signup Experience



Step 1

[Configure sign-in experience](#)

Step 2

[Configure security requirements](#)

Step 3

[Configure sign-up experience](#)

Step 4

Configure message delivery

Step 5

Integrate your app

Step 6

Review and create

Configure message delivery Info

Amazon Cognito uses Amazon SES and Amazon SNS to send email and SMS messages to your app users. Messages may incur additional SES and SNS costs.

Email

Configure how your user pool sends email messages to users.

Email provider Info

Send email with Amazon SES - Recommended

Send emails using an Amazon SES verified identity in your account. We recommend this option for higher email volume and production workloads.

Send email with Cognito

Use Cognito's default email address as a temporary start for development. You can use it to send up to 50 emails a day.

You must have configured a verified sender with [Amazon SES](#) to use the SES feature. [Learn more](#)

SES Region Info

Europe (Paris)

FROM email address Info

By default "no-reply@verificationemail.com" will be used. You can also choose a different email address that you have previously verified with Amazon SES.

no-reply@verificationemail.com



REPLY-TO email address - *optional* Info

If you set an invalid reply-to address, sending restrictions may be imposed on your account.

Enter an email address

Configure message delivery



Step 1

[Configure sign-in experience](#)

Step 2

[Configure security requirements](#)

Step 3

[Configure sign-up experience](#)

Step 4

[Configure message delivery](#)

Step 5

[Integrate your app](#)

Step 6

Review and create

Integrate your app info

Set up app integration for your user pool with Cognito's built-in authentication and authorization flows.

User pool name

Create a friendly name for your user pool.

User pool name

HealthCareCognitd

User pool names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = , . @ -

⚠ Your user pool name can't be changed once this user pool is created.

Hosted authentication pages

Choose whether to use Cognito's Hosted UI and OAuth 2.0 server for user sign-up and sign-in flows.

Use the Cognito Hosted UI

Build hosted sign-up, sign-in, and OAuth 2.0 service endpoints in Amazon Cognito. When this feature is not enabled, use Cognito API operations to perform sign-up and sign-in.

Initial app client

App integration

Initial app client

Configure an app client. App clients are single-app platforms in your user pool that have permissions to call unauthenticated API operations. A user pool can have multiple app clients.

App type | [Info](#)

Select an app type and we will automatically populate common default settings. You can add additional app clients after the user pool is created.

Public client
A native, browser or mobile-device app. Cognito API requests are made from user systems that are not trusted with a client secret.

Confidential client
A server-side application that can securely store a client secret. Cognito API requests are made from a central server.

Other
A custom app. Choose your own grant, auth flow, and client-secret settings.

App client name | [Info](#)

Enter a friendly name for your app client.

HealthCare

App client names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = , . @ -

Client secret | [Info](#)

Choose whether your app client will have a client secret. Client secrets are used by the server-side component of an app to authorize API requests. Using a client secret can prevent a third party from impersonating your client.

Generate a client secret

Don't generate a client secret

⚠ You cannot change or remove a client secret after you allow Amazon Cognito to generate it for your app client.

[CloudShell](#) [Feedback](#) © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#)

Initial app client

Amazon Cognito

User pools New Identity pools

Amazon Cognito > User pools

New from Amazon Verified Permissions! Cognito user group authorization for API Gateway

You can now create group-aware authorization policies for your APIs with Amazon Verified Permissions, a fine-grained authorization service for applications. [Learn more](#)

Go to Amazon Verified Permissions

User pools (1) [Info](#)

View and configure your user pools. User pools are directories of federated and local user profiles. They provide authentication options for your users.

Search user pools by name or ID

User pool name User pool ID Created time Last updated time

User pool name	User pool ID	Created time	Last updated time
HealthCareCognito	eu-west-3_fdfBrijpY	7 days ago	7 days ago

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Successful create user pool!

```
cognito_client = boto3.client('cognito-idp',
                               aws_access_key_id=os.getenv('AWS_ACCESS_KEY_ID'),
                               aws_secret_access_key=os.getenv('AWS_SECRET_ACCESS_KEY'),
                               region_name="eu-west-3")
```

Configure Cognito with actual application

The screenshot shows the AWS IAM Policies page. On the left, there's a sidebar with navigation links like Dashboard, Access management, Policies, and Access reports. The main area shows a policy named "CognitoExtra". The "Policy details" section includes fields for Type (Customer managed), Creation time (May 05, 2024, 01:01 (UTC+02:00)), Edited time (May 05, 2024, 01:01 (UTC+02:00)), and ARN (arn:aws:iam::975050211573:policy/CognitoExtra). Below this, the "Permissions" tab is selected, showing a table with one row: "Cognito User Pools" with an "Access level" of "Limited: Write" and a "Resource" of "region| string like |eu-west-3". There are also tabs for Entities attached, Tags, Policy versions (1), and Access Advisor.

Create policies for IAM user to allow Cognito

- AWS DynamoDB:

Create table

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String ▾

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

String ▾

1 to 255 characters and case sensitive.

Table settings

Default settings Customize settings

 CloudShell Feedback

Create table to store users data

DynamoDB

Dashboard

Tables

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Reserved capacity

Settings

DAX

Clusters

Subnet groups

Parameter groups

Events

DynamoDB > Tables

Tables (8) [Info](#)

Find tables by table name

Actions ▾ Delete Create table

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
<input type="checkbox"/>	Appointment		id (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Doctors		email (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Medical_Specialties		code (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Online_Appointment		id (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Online_Services		Code (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Procedures		code (S)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Rooms		roomNumber (N)	-	0		Provisioned (1)	Provisioned (1)
<input type="checkbox"/>	Users		email (S)	-	0		Provisioned (1)	Provisioned (1)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preference

Create other tables needed for the system

The screenshot shows the AWS Identity and Access Management (IAM) service interface. The left sidebar includes links for Dashboard, Access management (User groups, Roles, Policies, Identity providers, Account settings), Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies), CloudWatch Feedback, and Help.

The main content area displays the following information:

- Permissions policies (5)**: A table listing five attached policies:

Policy name	Type	Attached via
AllowGetForS3	Customer managed	Directly
AmazonDynamoDBFullAccess	AWS managed	Directly
AmazonS3FullAccess	AWS managed	Directly
AmazonS3OutpostsFullAccess	AWS managed	Directly
CognitoExtra	Customer managed	Directly
- Permissions boundary (not set)**: A section indicating no permissions boundary is currently set.
- Generate policy based on CloudTrail events**: A section for generating new policies based on CloudTrail event logs.

At the top right, there is a user profile for "Victor" and standard AWS navigation icons.

Allow full access for IAM users to DynamoDB

```
dynamodb = boto3.resource(  
    'dynamodb',  
    aws_access_key_id=os.getenv('AWS_ACCESS_KEY_ID'),  
    aws_secret_access_key=os.getenv('AWS_SECRET_ACCESS_KEY'),  
    region_name="eu-west-3"  
)  
table = dynamodb.Table('Users')  
online_service_table = dynamodb.Table('Online_Services')  
online_appointment_table = dynamodb.Table('Online_Appointment')  
doctor_table = dynamodb.Table('Doctors')  
appointment_table = dynamodb.Table('Appointment')
```

Integrate with actual application

- AWS S3:

The screenshot shows the AWS S3 'Create bucket' interface. At the top, the navigation path is 'Amazon S3 > Buckets > Create bucket'. The main title is 'Create bucket' with an 'Info' link. Below it, a sub-instruction says 'Buckets are containers for data stored in S3.'

General configuration

AWS Region: Europe (Paris) eu-west-3

Bucket name: Info

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) [F]

Copy settings from existing bucket - *optional*: Only the bucket settings in the following configuration are copied. [Choose bucket](#)

Format: s3://bucket/prefix

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account.
Access to this bucket and its objects is specified using [AWS Identity and Access Management \(IAM\)](#).

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using [AWS Identity and Access Management \(IAM\)](#).

Create s3 bucket

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

- Disable
 Enable

Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

Other settings for s3 bucket

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::healthcarebucketepita2/*"  
        },  
        {  
            "Effect": "Allow",  
            "Principal": {"AWS": "arn:aws:iam::975050211573:role/aws-service-role/reports.backup.amazonaws.com/AWSServiceRoleForBackupReports"},  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::healthcarebucketepita2/*",  
            "Condition": {  
                "StringEquals": {  
                    "s3:x-amz-acl": "bucket-owner-full-control"  
                }  
            }  
        }  
    ]  
}
```

 Copy

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

 Edit

Add permissions to bucket

```
s3_client = boto3.client(  
    's3',  
    aws_access_key_id=os.getenv('AWS_ACCESS_KEY_ID'),  
    aws_secret_access_key=os.getenv('AWS_SECRET_ACCESS_KEY'),  
    region_name="eu-west-3"  
)
```

Call s3 client in the actual app

```
tabnine: test | fix | explain | document | ask
@admin.route('/admin/doctors/add', methods=['POST'])
def add_doctor():
    first_name = request.form['firstName']
    last_name = request.form['lastName']
    email = request.form['email']
    phone = request.form['phone']
    gender = request.form['gender']
    address = request.form['address']
    specialty = request.form['specialty']
    image_file = request.files['doctorImage']

    if image_file:
        image_filename = secure_filename(image_file.filename)
        unique_filename = f"{uuid.uuid4()}-{image_filename}"
        s3_client.upload_fileobj(
            image_file,
            'healthcarebucketepita2',
            unique_filename
        )
        # Construct the URL based on S3 endpoint
        image_url = f"https://healthcarebucketepita2.s3.eu-west-3.amazonaws.com/{unique_filename}"
    else:
        image_url = None # Handle case where no image is provided
```

Store images to s3 bucket

- AWS EC2:

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Recents | My AMIs | **Quick Start**

Amazon Linux | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

Summary

Number of instances Info

Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-00ac45f3035ff009e

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year
includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

Cancel | **Launch instance** | Review commands

© 2024, Amazon Web Services, Inc. or its affiliates.

Launch an instance

The screenshot shows the AWS Launch Wizard interface for creating a new Amazon Machine Image (AMI). The process is divided into several steps:

- Step 1: Choose AMI**
 - Selected AMI: Ubuntu Server 24.04 LTS (HVM), SSD Volume Type (ami-00ac45f3035ff009e)
 - AMI ID: ami-00ac45f3035ff009e
 - Architecture: 64-bit (x86)
 - Description: Canonical, Ubuntu, 24.04 LTS, amd64 noble image build on 2024-04-23
 - Free tier eligible
- Step 2: Choose instance type**
 - Selected Instance Type: t2.micro
 - Family: t2
 - 1 vCPU
 - 1 GiB Memory
 - Current generation: true
 - On-Demand RHEL base pricing: 0.0732 USD per Hour
 - On-Demand SUSE base pricing: 0.0132 USD per Hour
 - On-Demand Linux base pricing: 0.0132 USD per Hour
 - On-Demand Windows base pricing: 0.0178 USD per Hour
 - Free tier eligible
 - Radio button: All generations
 - Link: Compare instance types
- Step 3: Summary**
 - Number of instances: 1
 - Software Image (AMI): Canonical, Ubuntu, 24.04 LTS, ...read more (ami-00ac45f3035ff009e)
 - Virtual server type (instance type): t2.micro
 - Firewall (security group): New security group
 - Storage (volumes): 1 volume(s) - 8 GiB
- Step 4: Final Actions**
 - Cancel
 - Launch instance
 - Review commands

Choose instance type

The screenshot shows the 'Network settings' section of the AWS Launch Wizard. It includes fields for selecting a VPC (vpc-049afabd4da5e6cb2 | Amazon Default VPC), subnet (No preference (Default subnet in any availability zone)), and auto-assign public IP (Enable). A note about additional charges applies when outside of the free tier allowance. The 'Firewall (security groups)' section allows creating a new security group ('Create security group') or selecting an existing one ('Select existing security group'). A warning message states: 'We'll create a new security group called 'launch-wizard-3' with the following rules:'. Under these rules, three checkboxes are checked: 'Allow SSH traffic from Anywhere' (Helps you connect to your instance), 'Allow HTTPS traffic from the internet' (To set up an endpoint, for example when creating a web server), and 'Allow HTTP traffic from the internet' (To set up an endpoint, for example when creating a web server). A note below the checkboxes says: 'Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' On the right, the 'Summary' section shows 1 instance, Canonical Ubuntu 24.04 LTS AMI, t2.micro instance type, a new security group, and 1 volume(s) - 8 GiB storage. A 'Free tier' information box is present. At the bottom are 'Cancel', 'Launch instance', and 'Review commands' buttons.

▼ Network settings [Info](#)

Network [Info](#)
vpc-049afabd4da5e6cb2 | Amazon Default VPC

Subnet [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)
Enable

Additional charges apply when outside of [free tier allowance](#)

Firewall (security groups) [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called '**launch-wizard-3**' with the following rules:

Allow SSH traffic from [Anywhere](#)
Helps you connect to your instance
0.0.0.0/0

Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠️ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. **X**

▼ Summary

Number of instances [Info](#)
1

Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...[read more](#)
ami-00ac45f3035ff009e

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

ⓘ Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

[Cancel](#) [Launch instance](#) [Review commands](#)

[CloudShell](#) [Feedback](#) © 2024, Amazon Web Services, Inc. or its affiliates.

Network setting for instance

```
ubuntu@ip-172-31-5-118:~$ sudo apt install python3-pip python3-dev git nginx -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7).
git set to manually installed.
The following additional packages will be installed:
binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-13
g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13 gcc-13-base gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu javascript-common libalgorithm-diff-perl
libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan8 libatomic1 libbinutils libcc1-0 libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot
libfile-fcntllock-perl libgcc-13-dev libgomp1 libprofng0 libhwasan0 libis123 libxml libjs-jquery libjs-sphinxdoc libjs-underscore libisan0 libmpc3
libpython3-dev libpython3.12-dev libquadmath0 libsframe1 libstdc++-13-dev libtsan2 libubsan1 lto-disabled-list make nginx-common python3-wheel python3.12-dev
zlib1g-dev
Suggested packages:
binutils-doc gprofng-gui bzip2-doc cpp-doc gcc-13-locales cpp-13-doc debian-keyring g++-multilib g++-13-multilib gcc-13-doc gcc-multilib autoconf automake libtoo
flex bison gdb gcc-doc gcc-13-multilib gdb-x86-64-linux-gnu bzr libstdc++-13-doc make-doc fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-13
g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13 gcc-13-base gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu javascript-common libalgorithm-diff-perl
libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan8 libatomic1 libbinutils libcc1-0 libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot
libfile-fcntllock-perl libgcc-13-dev libgomp1 libprofng0 libhwasan0 libis123 libxml libjs-jquery libjs-sphinxdoc libjs-underscore libisan0 libmpc3
libpython3-dev libpython3.12-dev libquadmath0 libsframe1 libstdc++-13-dev libtsan2 libubsan1 lto-disabled-list make nginx nginx-common python3-dev python3-pip
python3-wheel python3.12-dev zlib1g-dev
0 upgraded, 61 newly installed, 0 to remove and 0 not upgraded.
Need to get 78.1 MB of archives.
```

Install required package on ubuntu

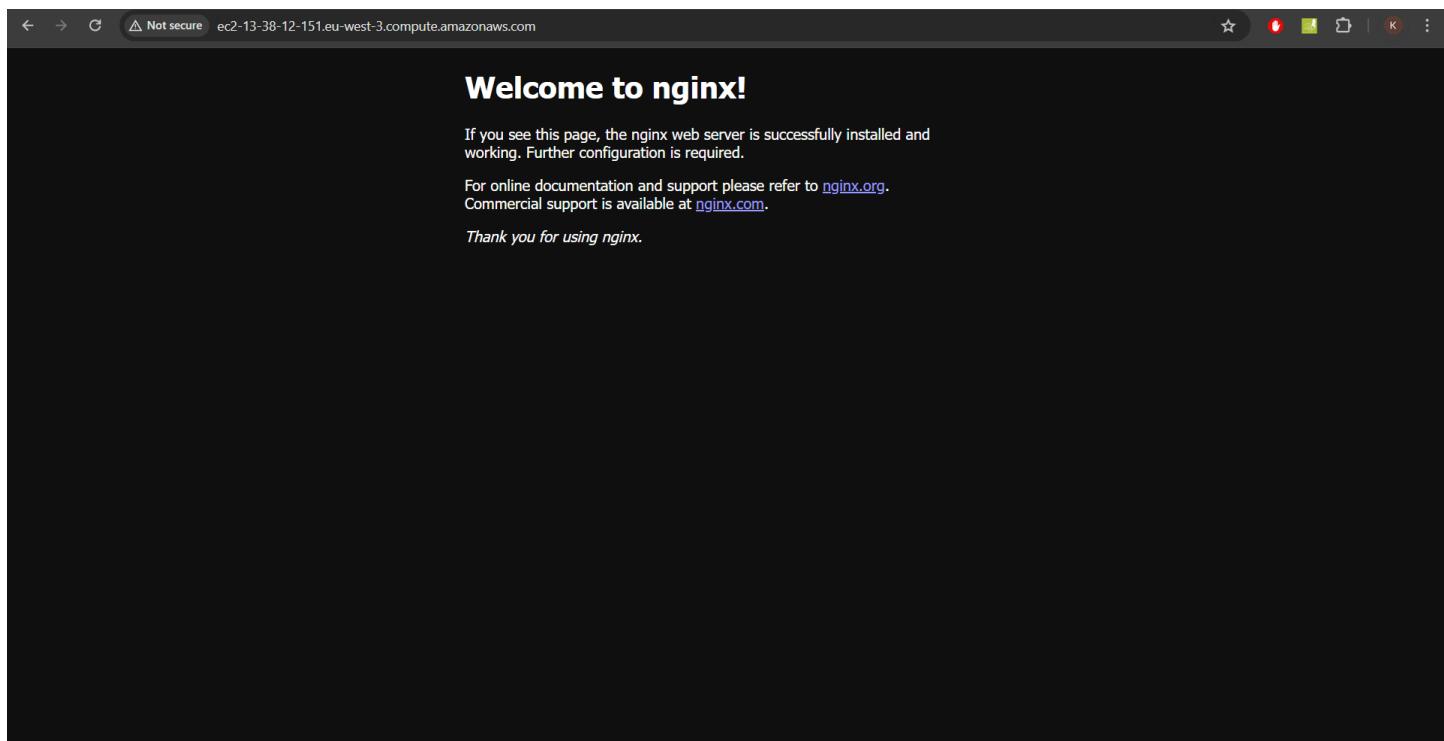
```
ubuntu@ip-172-31-5-118:~$ git clone git@gitlab.com:tinnd1506/hms-web-sever.git
Cloning into 'hms-web-sever'...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (89/89), done.
remote: Compressing objects: 100% (74/74), done.
remote: Total 121 (delta 47), reused 13 (delta 13), pack-reused 32 (from 1)
Receiving objects: 100% (121/121), 2.86 MiB | 5.86 MiB/s, done.
Resolving deltas: 100% (47/47), done.
ubuntu@ip-172-31-5-118:~$
```

```
i-0a04560ad07f7842b (HealthCare)
PublicIPs: 13.38.12.151 PrivateIPs: 172.31.5.118
```

Clone gitlab repo

```
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ gunicorn --workers 3 --bind 0.0.0.0:8000 app:app
[2024-05-12 14:30:36 +0000] [4838] [INFO] Starting gunicorn 22.0.0
[2024-05-12 14:30:36 +0000] [4838] [INFO] Listening at: http://0.0.0.0:8000 (4838)
[2024-05-12 14:30:36 +0000] [4838] [INFO] Using worker: sync
[2024-05-12 14:30:36 +0000] [4839] [INFO] Booting worker with pid: 4839
[2024-05-12 14:30:36 +0000] [4840] [INFO] Booting worker with pid: 4840
[2024-05-12 14:30:36 +0000] [4841] [INFO] Booting worker with pid: 4841
DEBUG:botocore.hooks:Event choose-service-name: calling handler <function handle_service_name_alias at 0x72a95beacc20>
DEBUG:botocore.hooks:Event creating-client-class.dynamodb: calling handler <function add_generate_presigned_url at 0x72a95bfdee80>
DEBUG:botocore.configprovider:Looking for endpoint for dynamodb via: environment_service
DEBUG:botocore.configprovider:Looking for endpoint for dynamodb via: environment_global
```

Test run with gunicorn



Test run successfully

```
GNU nano 7.2                               /etc/nginx/sites-available/healthcare *

server {
    listen 80;
    server_name ec2-13-38-12-151.eu-west-3.compute.amazonaws.com;

    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

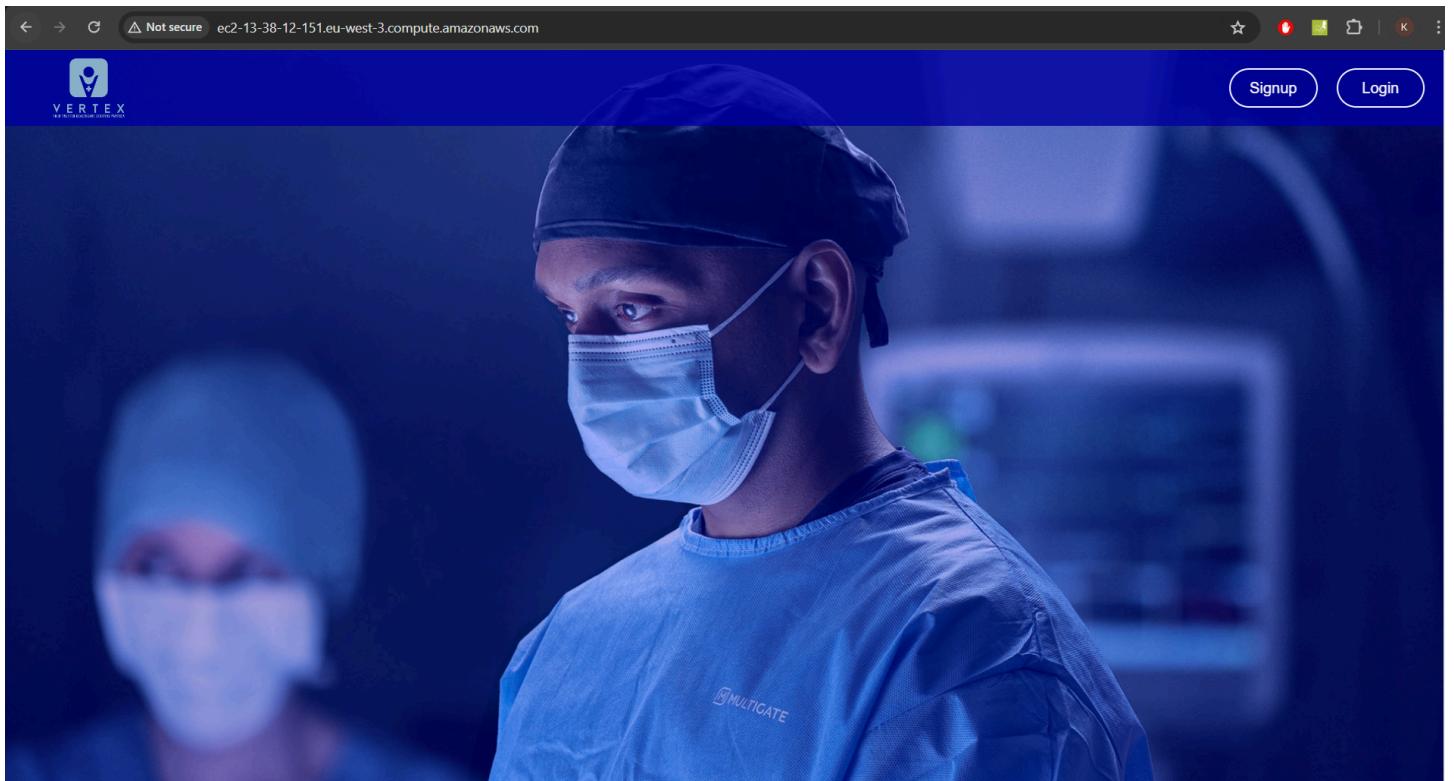
^G Help          ^O Write Out      ^N Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo       M-A Set Mark   M-L To Bracket M-Q Previous
^M Exit         ^P Read File     ^V Replace       ^U Paste        ^J Justify      ^Y Go To Line   M-U Redo       M-B Copy      ^Q Where Was    M-W Next
i-0a04560ad07f7842b (HealthCare)           X
Public IPs: 13.38.12.151  Private IPs: 172.31.5.118

CloudShell  Feedback  © 2024, Amazon Web Services, Inc. or its affiliates.  Privacy  Terms  Cookie preferences
```

Configure nginx file

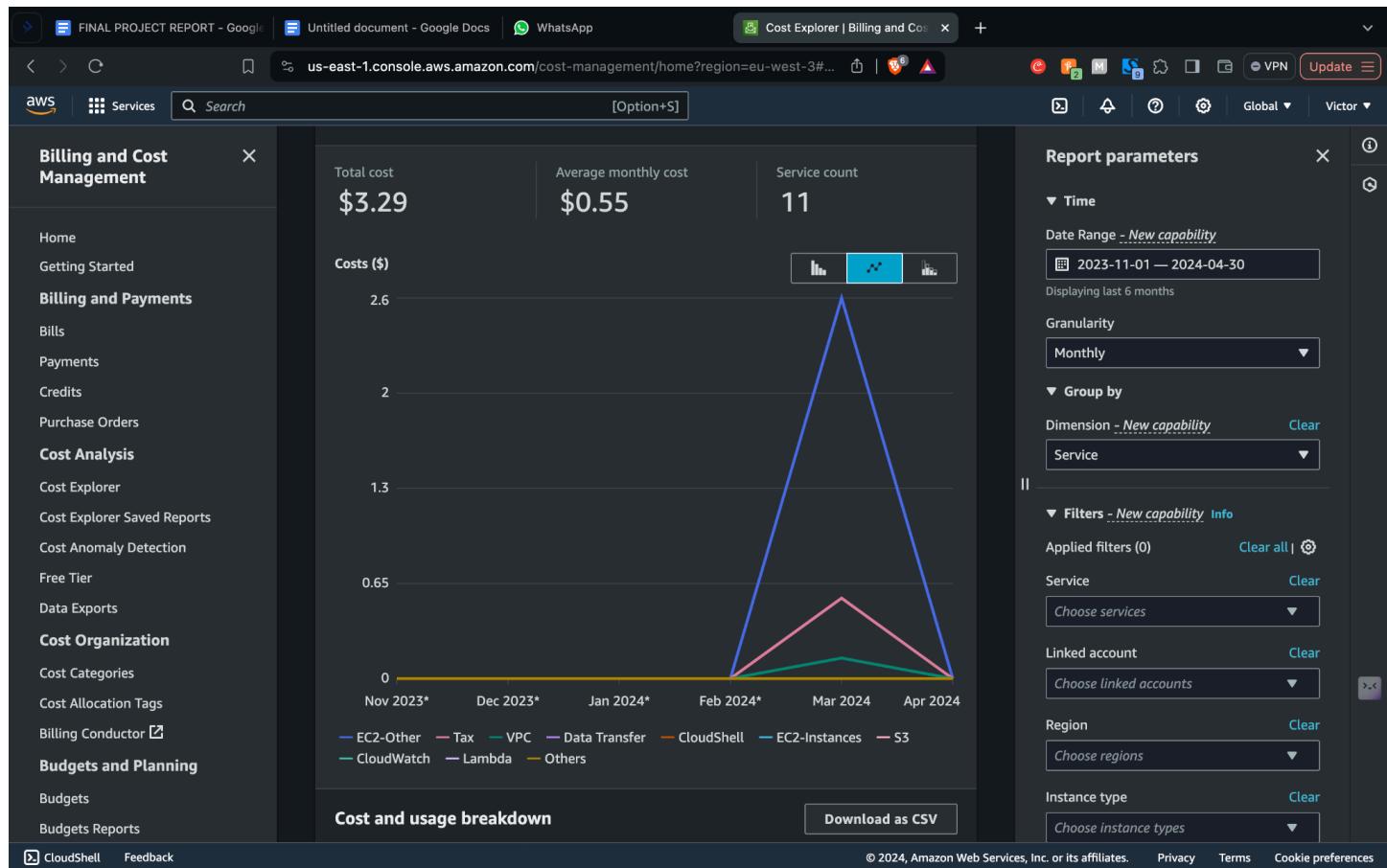
```
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ sudo rm /etc/nginx/sites-enabled/default
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ sudo nano /etc/nginx/sites-available/healthcare
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ sudo ln -s /etc/nginx/sites-available/healthcare /etc/nginx/sites-enabled
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ sudo systemctl restart nginx
(venv) ubuntu@ip-172-31-5-118:~/hms-web-sever$ gunicorn --workers 3 --bind 0.0.0.0:8000 app:app
[2024-05-12 14:34:40 +0000] [4880] [INFO] Starting gunicorn 22.0.0
[2024-05-12 14:34:40 +0000] [4880] [INFO] Listening at: http://0.0.0.0:8000 (4880)
[2024-05-12 14:34:40 +0000] [4880] [INFO] Using worker: sync
[2024-05-12 14:34:40 +0000] [4881] [INFO] Booting worker with pid: 4881
[2024-05-12 14:34:40 +0000] [4882] [INFO] Booting worker with pid: 4882
[2024-05-12 14:34:40 +0000] [4883] [INFO] Booting worker with pid: 4883
DEBUG:botocore.hooks:Event choose-service-name: calling handler <function handle_service_name alias at 0x70715357cc20>
DEBUG:botocore.hooks:Event creating-client-class.dynamodb: calling handler <function add_generate_presigned url at 0x7071534aee80>
```

Confirm nginx setup and restart the server



Server is running on ec2 instance

- AWS Cost Explorer:



- IAM (Identity and Access Management):

The screenshot shows the AWS IAM User details page for a user named "AdminHealthCare". The page is titled "AdminHealthCare" and includes tabs for "Permissions", "Groups", "Tags (1)", "Security credentials", and "Access Advisor".

Summary:

ARN	Console access	Access key 1
arn:aws:iam::975050211573:user/AdminHealthCare	Disabled	AKIA6GBMFQD2YL3IWDUB - Active Used 6 days ago. 7 days old.
Created	Last console sign-in	Access key 2
May 04, 2024, 16:59 (UTC+02:00)	-	AKIA6GBMFQD27J5U73HG - Active Never used. 6 days old.

Permissions policies (5):

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
AllowGetForS3	Customer managed	Directly
AmazonDynamoDBFullAccess	AWS managed	Directly

IAM User for admin

The screenshot shows the AWS Identity and Access Management (IAM) service interface. On the left, there's a navigation sidebar with sections like Dashboard, Access management (Roles, Policies, Identity providers, Account settings), and Access reports (Access Analyzer, External access, Unused access, Analyzer settings). The main content area is titled "Roles (15) Info". It displays a table of 15 IAM roles, each with a checkbox, the role name, the trusted entity (e.g., AWS Service: backup, AWS Service: ssm (Service-Linked Role), AWS Service: ops.apigateway (Service-Linked Role), etc.), and the last activity time (e.g., 19 hours ago, 2 hours ago, 2 days ago, etc.). A search bar at the top of the table allows filtering by role name. At the bottom right of the table, there are buttons for "Create role" and "Delete". The overall interface is dark-themed.

Role name	Trusted entities	Last activity
AWSBackupDefaultServiceRole	AWS Service: backup	19 hours ago
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Role)	2 hours ago
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	2 days ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling	36 minutes ago
AWSServiceRoleForApplicationInsights	AWS Service: application-insights (Service-Linked Role)	-
AWSServiceRoleForBackup	AWS Service: backup (Service-Linked Role)	22 hours ago
AWSServiceRoleForBackupReports	AWS Service: reports.backup (Service-Linked Role)	5 hours ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
CognitoMessageRole	AWS Service: cognito-identity	7 days ago
Custom_role	AWS Service: ec2	7 hours ago
gitlab-javascript-serverless-example-production-us-east-1-lambdaRole	AWS Service: lambda	13 days ago
gitlabpoc-HelloWorldFunctionRole-sfpHpT9xZwz8	AWS Service: lambda	13 days ago
HealthCareSignUpNoti-role-8i35ur8n	AWS Service: lambda	-
SignUpNoti-role-vg7ij4r2	AWS Service: lambda	-

IAM roles

- VPC:

Screenshot of the AWS VPC Console showing the details of a VPC named "healthcare_VPC".

VPC dashboard

Details **Info**

VPC ID vpc-0dfa8e9ddb9d0f1f9	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-09a9dee8a12422e2c	Main route table rtb-06442a04a48e0af66	Main network ACL acl-05bc79dc96803d7cc
Default VPC No	IPv4 CIDR 125.0.0.0/16	IPv6 pool -	IPv6 CIDR -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 975050211573	

Resource map **CIDRs** **Flow logs** **Tags** **Integrations**

Resource map **Info**

VPC **Show details**
Your AWS virtual network
healthcare_VPC

Subnets (3)
Subnets within this VPC
eu-west-3b
healthcare subnet private 1
healthcare subnet private 2

Route tables (1)
Route network traffic to resources
rtb-06442a04a48e0af66

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- Attach API Gateway to ec2 Flask App:

The screenshot shows the AWS API Gateway 'Create an API' wizard, Step 1: Create and configure integrations. The URL is eu-west-3.console.aws.amazon.com/apigateway/main/create?region=eu-west-3. The page title is 'API Gateway - Create API'. The left sidebar shows navigation steps: Step 1 (Create an API), Step 2 (optional: Configure routes), Step 3 (optional: Define stages), and Step 4 (Review and Create). The main content area is titled 'Create and configure integrations' and contains a note about specifying backend services. It shows an 'Integrations (0)' section with an 'Add integration' button. An input field for 'API name' is filled with 'HMS-API'. At the bottom are 'Cancel', 'Review and Create', and 'Next' buttons.

Create an API

The screenshot shows the AWS API Gateway Routes page for an API named "HMS-API". A green success message at the top states "Successfully created API HMS-API (o7hmjlwm91)". The main area is titled "Routes" and contains a sub-section titled "Routes for HMS-API" with a "Create" button. A search bar is also present. The left sidebar includes sections for APIs, Custom domain names, VPC links, Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), Monitor (Metrics, Logging), and Protect (Throttling). The bottom navigation bar includes CloudShell, Feedback, and links to AWS services like ChatGPT, Home, and VPN.

The screenshot shows the "Create a route" dialog box. At the top, a green success message says "Successfully created API HMS-API (o7hmjlwm91)". Below it, the breadcrumb navigation shows "API Gateway > APIs > Routes - HMS-API (o7hmjlwm91) > Create a route". The main form is titled "Route and method" with an "Info" link. It has fields for "Route name eg. /pets" and "Choose a method and enter a path to create a route. You can also specify one \$default route per API. The \$default route is invoked when the request to the API matches no other routes." The method dropdown is set to "ANY" and the path field contains "{proxy+}". At the bottom right are "Cancel" and "Create" buttons. The left sidebar and bottom navigation bar are identical to the first screenshot.

Specify route as a catch-all route

Screenshot of the AWS API Gateway console showing the 'Integrations' page for an API named 'HMS-API'. The left sidebar shows navigation options like 'Develop', 'Deploy', 'Monitor', and 'Protect'. The main area displays 'Routes for HMS-API' with one route configuration: '/{proxy+}' mapped to 'ANY'. The 'Integration details' section indicates no integrations are configured and provides a 'Create and attach an integration' button.

Screenshot of the AWS Cloud Map console landing page. It features a central diagram illustrating how AWS Cloud Map works: an 'AWS Cloud Map' icon connects to a 'myApp.dev' service instance, which then connects to a 'Cluster A' instance. To the right, sections include 'Build a map of your cloud' (with a 'Create namespace' button), 'Pricing (US)', and 'More resources'.

Create namespace in AWS Cloudmap

AWS Cloud Map > Namespaces > Create namespace

Create namespace Info

A namespace typically contains the services for one application.

Namespace configuration

The namespace configuration determines how your application discovers service instances

Namespace name
A friendly name lets you easily find a namespace on the dashboard.

The namespace name can have up to 1,024 characters, and must start and end with a letter.
Valid characters: a-z, A-Z, 0-9, . (period), _ (underscore), and - (hyphen)

Namespace description - optional
This description appears on your dashboard. It can help you quickly identify what your namespace is used for.

The description can have up to 1,024 characters.

Instance discovery
Instance discovery determines how your application discovers registered instances.

- API calls
Your application makes API calls to discover registered instances.
- API calls and DNS queries in VPCs
Your application makes API calls or submits DNS queries in VPCs to discover registered instances.
- API calls and public DNS queries
Your application makes API calls or submits public DNS queries to discover registered instances. Additional charges apply.

Tabs

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Your namespace flask-namespace is created and ready to use
You can now create a service for your namespace

AWS Cloud Map > Namespaces

Namespaces (1) <small>Info</small>		<input type="button" value="C"/>	<input type="button" value="View details"/>	<input type="button" value="Delete"/>	<input type="button" value="Create namespace"/>
<input type="text" value="Find namespace"/>					
Domain name	Description	Instance discovery	Namespace ID		
<input type="radio"/> flask-namespace	-	API calls	ns-hcgdyut27kt6si7		

The screenshot shows the AWS CloudMap console interface for creating a new namespace named "flask-namespace".

Namespace Summary:

- Namespace Name: flask-namespace
- Region: eu-west-3
- Status: Your namespace flask-namespace is created and ready to use.
- Action: Create service

Tags:

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value.

Key	Value
No tags	

There are no tags for this resource.

Services:

Name	Description	Discoverable By	Date created	Service ID
No services				

There are no services in this namespace.

Actions:

- Create service
- View details
- Delete

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create service

Your namespace flask-namespace is created and ready to use

You can now create a service for your namespace

Service name

A friendly name lets you easily find a service on the dashboard.

single-service

The service name can have up to 1,024 characters.

Service description - optional

A description can help you remember details about this service.

Service for payment refunds.

The description can have up to 1,024 characters.

Health check configuration

Health check options

Health checks determine whether a resource is healthy. If not, traffic is automatically routed to other resources.

No health check
If you don't configure a health check, traffic will be routed to service instances regardless of whether they're healthy.

Route 53 health check
A Route 53 health check is automatically associated with DNS records. An additional charge applies.

Custom health check
A custom health check requires a third-party health-checking tool.

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Your namespace flask-namespace is created and ready to use

You can now create a service for your namespace

Your service single-service is created and ready to use

Register service instance

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value.

Key Value

No tags

There are no tags for this resource.

Service instances

C View details Deregister Register service instance

Find instance < 1 > ⌂

ID	Health
No instances	

You haven't registered any service instances for this service

Register service instance

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

eu-west-3.console.aws.amazon.com/cloudmap/home/namespaces/ns-hcgyd... | API Gateway - Integrations | Register instance | Sign Up | Untitled document - | + | ChatGPT | (5) Attach AWS API C | Create service | X | Services | Search | [Option+S] | X | Paris | Victor | Update

Your namespace flask-namespace is created and ready to use
You can now create a service for your namespace

Your service single-service is created and ready to use
REGISTER SERVICE INSTANCE [info](#)

Service instance information
AWS Cloud Map uses these settings in the specified service to register one service instance.

Instance type
When your application requests a service instance, Cloud Map returns the information that is required to access the instance.

IP Address
You can register service instances for AWS and external resources that are available at an IP address.

EC2 Instance
You can register service instances with AWS EC2 instance Id.

Identifying information for another resource
This option is for resources that aren't EC2 instances and that don't have an IP address.

Service instance ID
You can use this value to update an existing service instance.
myInstance-53

Standard attributes

IPv4 address
192.0.2.44

IPv6 address
2001:0db8:85a3:0000:abcd:0001:2345

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Attach Ip address of the EC2 to the CloudMap

eu-west-3.console.aws.amazon.com/cloudmap/home/namespaces/ns-hcgyd... | Instance details | EC2 | Register instance | Sign Up | Untitled document - | + | ChatGPT | (5) Attach AWS API C | Create service | X | Services | Search | [Option+S] | X | Paris | Victor | Update

Your namespace flask-namespace is created and ready to use
You can now create a service for your namespace

Your service single-service is created and ready to use
Register service instance [X](#)

Instance type
When your application requests a service instance, Cloud Map returns the information that is required to access the instance.

IP Address
You can register service instances for AWS and external resources that are available at an IP address.

EC2 Instance
You can register service instances with AWS EC2 instance Id.

Identifying information for another resource
This option is for resources that aren't EC2 instances and that don't have an IP address.

Service instance ID
You can use this value to update an existing service instance.
my-ec2

Standard attributes

IPv4 address
172.31.37.93

IPv6 address
2001:0db8:85a3:0000:abcd:0001:2345

Port - optional
8000

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS API Gateway VPC links page. On the left sidebar, under the 'APIs' section, 'VPC links' is selected. The main content area is titled 'VPC links' and contains a search bar labeled 'Find APIs'. Below the search bar is a table header with columns for 'Name'. A message on the right says 'Choose a VPC link to view its details.'

Create VPC link

The screenshot shows the 'Create VPC link' wizard. Step 1: 'Choose a VPC link version'. It offers two options: 'VPC link for REST APIs' (unchecked) and 'VPC link for HTTP APIs' (checked). The checked option is described as 'This VPC link can be used with HTTP APIs.' Below this is the 'VPC link details' section, which includes a 'Name' field containing 'any-name' and a 'VPC' dropdown set to 'Amazon Default VPC (vpc-049afabd4da5e6cb2)'. The final section is 'Subnets', which lists three subnets: 'subnet-0c68f252e483dd3d3' (Availability Zone eu-west-3b, CIDR 172.31.16.0/20), 'subnet-0ea5d13d4939642fd' (Availability Zone eu-west-3c, CIDR 172.31.32.0/20), and 'subnet-06c815525081f8bb7' (Availability Zone eu-west-3a, CIDR 172.31.0.0/20).

The screenshot shows the AWS API Gateway VPC Links creation interface. The 'Name' field is set to 'any-name'. The 'VPC' dropdown is set to 'Amazon Default VPC (vpc-049afabd4da5e6cb2)'. The 'Subnets' section lists a single subnet: 'subnet-0ea5d13d4939642fd' (eu-west-3c, 172.31.32.0/20). The 'Security groups' section lists a single security group: 'sg-0c16323adf9185fda' (launch-wizard-1 created 2024-05-05T23:57:10.705Z). The 'Tags' section is empty. The bottom navigation bar includes CloudShell, Feedback, and links to 2024 AWS terms.

Add Subnets and Security Groups

The screenshot shows the AWS API Gateway Integration configuration interface. The left sidebar shows 'Develop' selected under 'Integrations'. The main area is titled 'Attach this integration to a route' and shows a placeholder 'ANY /{proxy+}'. The 'Integration target' section has 'Private resource' selected. The 'Integration details' section shows 'Select manually' as the selection method. The 'VPC link' section has a dropdown labeled 'Choose a VPC link'. The bottom navigation bar includes CloudShell, Feedback, and links to 2024 AWS terms.

The screenshot shows the AWS API Gateway integration configuration page. The left sidebar is collapsed, and the main area is titled "Integrations". Under "Target service", the "Cloud Map" option is selected, indicated by a blue circle and the text "Choose an AWS Cloud Map service to send the request to.". Below this, the "Namespace" dropdown is set to "flask-namespace" and the "Service" dropdown is set to "single-service". There is also an optional "AWS Cloud Map parameters" field containing "stage=prod&deployment=green_deployment". A "Description" field is present but empty. At the bottom, there is a "VPC link" section with a dropdown menu showing "any-name". The top navigation bar includes tabs for "APIs", "Custom domain names", and "VPC links", along with "API: HMS-API(o7hmjlwm91)". The bottom navigation bar includes "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

Integrate Cloud Map with Api Gateway

This screenshot shows the same AWS API Gateway integration configuration page as the previous one, but with a VPC link selected. In the "VPC link" section at the bottom, the dropdown menu is open and shows "any-name" as the chosen option. The rest of the configuration remains the same, with "Cloud Map" selected as the target service and the namespace and service set to their previous values. The bottom navigation bar is identical to the first screenshot.

The screenshot shows the AWS API Gateway Integrations page for the 'HMS-API' (o7hmjlwm91). On the left sidebar, under the 'Develop' section, 'Integrations' is selected. In the main content area, the 'Routes for HMS-API' section shows a single route: '/{proxy+}' with 'ANY' as the method and 'VPC Cloud Map' as the integration type. The 'Integration details for route' panel on the right shows the configuration for this route, including the AWS Cloud Map service ('ANY single-service - flask-namespace'), Integration ID ('lgkafh9'), and other settings like VPC link and timeout. A 'Request parameter mapping' and 'Response parameter mappings' section is also present.

Integrated successfully

The screenshot shows the AWS API Gateway Stages page for the 'HMS-API' (o7hmjlwm91). On the left sidebar, under the 'Deploy' section, 'Stages' is selected. The main content area displays the 'Stages for HMS-API' section, which lists a single stage named '\$default'. A 'Create' button is available to add new stages. A 'Select a stage' dropdown menu is open, showing '\$default' as the selected option. A 'Deploy' button is located at the top right of the stage list.

Test the stages

- FLask App source code:

The screenshot shows a web browser window with the URL gitlab.com/tinnd1506/hms-web-sever. The page displays the repository 'HMS-Web-Sever' created by Tin Nguyen. The sidebar on the left contains project management sections like Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main content area shows a list of commits under the 'main' branch. One commit is highlighted, showing changes made by Khanh Nguyen Tran 6 days ago. The commit details include files like .env, app.py, and requirements.txt. On the right, there's a 'Project information' sidebar with metrics such as 18 Commits, 1 Branch, 0 Tags, and 2.9 MiB Project Storage. A 'Created on' section indicates the project was created on May 04, 2024.

HMS-Web-Sever

changes

Name	Last commit	Last update
components	region	6 days ago
static	changes	6 days ago
templates	changes	6 days ago
.env	CHANGE REGION	6 days ago
app.py	changes	6 days ago
requirements.txt	commit	1 week ago

Project information

- 18 Commits
- 1 Branch
- 0 Tags
- 2.9 MiB Project Storage

Created on

May 04, 2024

Accessible via gitlab link: <https://gitlab.com/tinnd1506/hms-web-sever>

- App overview:



Home page



VERTEX
VERTECH VISIONS

Signup

Login

Sign Up

Email:

Password:

First Name:

Last Name:

Birthdate:

 dd/mm/yyyy

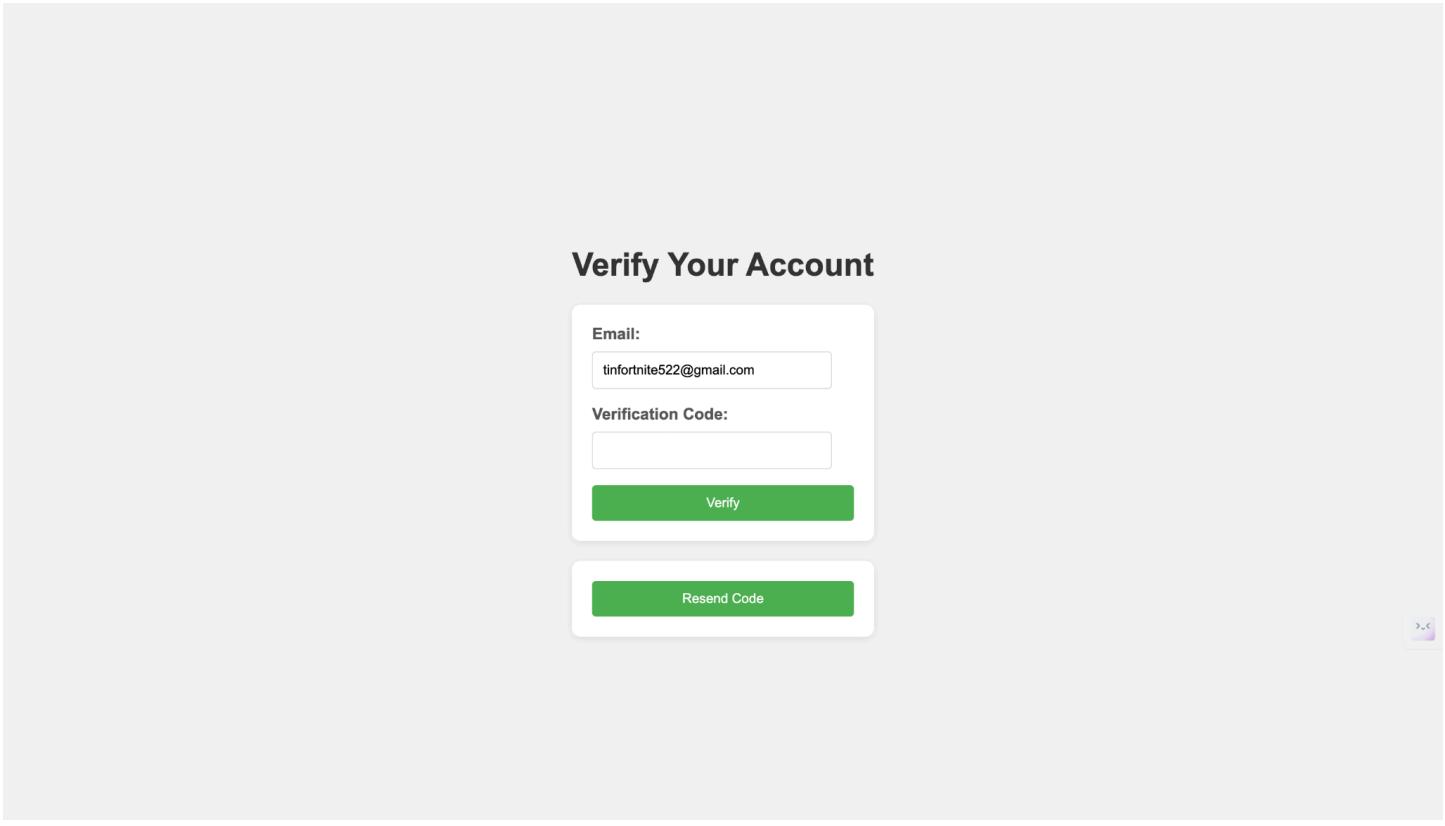
Gender:

Phone Number:

Address:

Submit

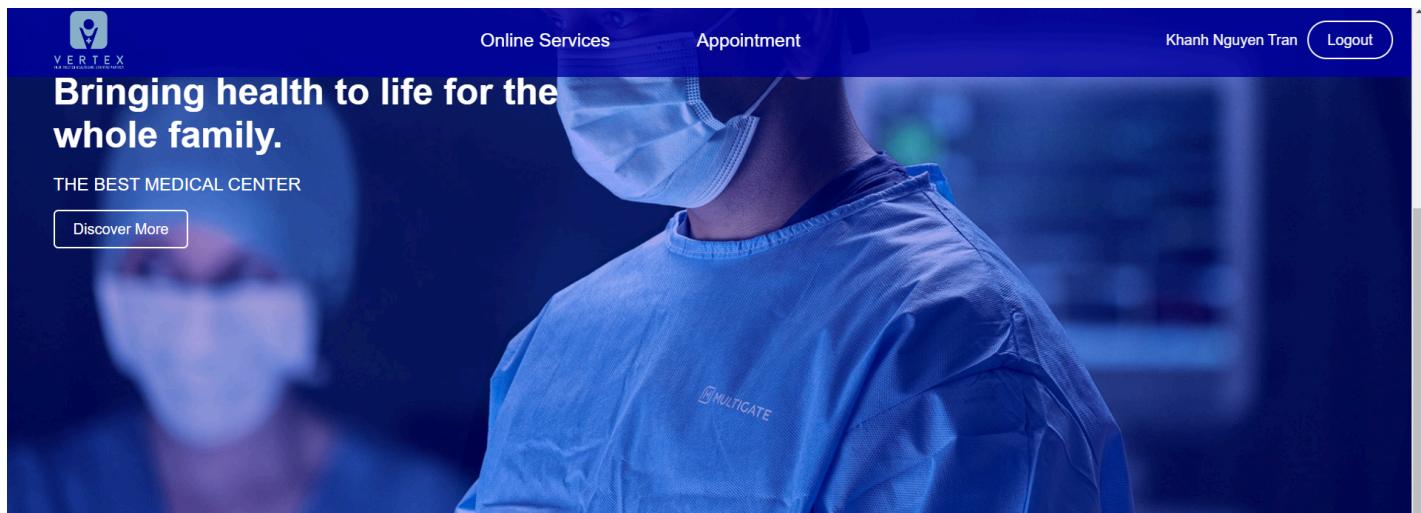
Sign-up page



Account verification page

The screenshot shows a web application's login interface. At the top, there is a dark blue header bar. On the left side of the header is a logo consisting of a white icon of a person with a graduation cap inside a square frame, followed by the word "VERTEX" in white capital letters, with a smaller line of text below it that is too small to read. On the right side of the header are two white-outlined circular buttons: one labeled "Signup" and another labeled "Login". Below the header is a light gray background area containing a white rectangular login form. The form has a title "Login" at the top center. Below the title are two input fields: the first is labeled "Email:" and the second is labeled "Password:", both preceded by a small black placeholder text. At the bottom of the form is a large blue rectangular button with the word "Login" in white.

Login page



The screenshot shows a medical center's user dashboard. At the top, there is a dark blue header with the logo 'VERTEX' on the left, followed by 'Online Services' and 'Appointment' buttons. On the right, it shows the user's name 'Khanh Nguyen Tran' and a 'Logout' button. Below the header is a large banner with a blue-tinted image of a medical professional wearing a mask and scrubs. The text 'Bringing health to life for the whole family.' is displayed prominently over the image. Underneath the banner, the text 'THE BEST MEDICAL CENTER' is visible, along with a 'Discover More' button. The main content area is titled 'Upcoming Appointments' and contains a table with three rows of data. The table has columns for 'Doctor Name', 'Date', and 'Time'. The data is as follows:

Doctor Name	Date	Time
Stephen Curry	2024-05-10	12:40
Alvin Hayes	2024-05-16	03:29
Lebron James	2024-05-25	08:18

User dashboard

 VERTEX
WEBSITE DESIGN & DEVELOPMENT

Online Services Appointment [Logout](#)



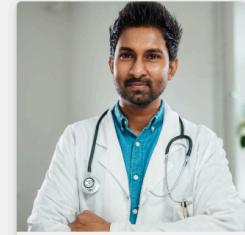
Alvin Hayes
Specialty: Ophthalmology
[Choose Service](#)



Stephen Curry
Specialty: Gastroenterology
[Choose Service](#)



Lebron James
Specialty: Cardiology
[Choose Service](#)



John Doe
Specialty: Infectious disease
[Choose Service](#)

Appointment page



VERTEX
WEBSITE FOR DOCTORS

Online Services

Appointment

Logout

Telemedicine Consultation

Cost: \$500

[Choose Service](#)

Mental Health Counseling

Cost: \$6000

[Choose Service](#)

Diet Counseling

Cost: \$2000

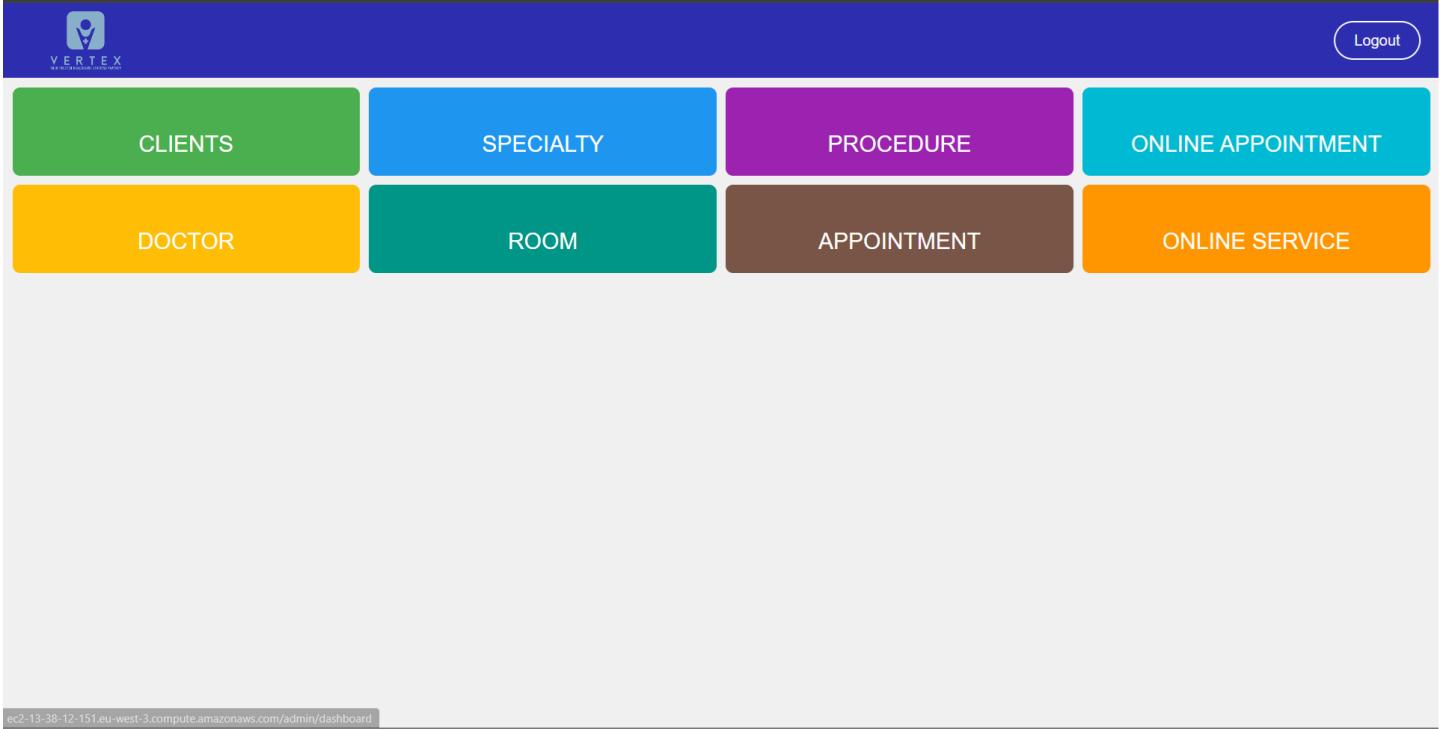
[Choose Service](#)

Remote Rehabilitation

Cost: \$20000

[Choose Service](#)

Online service page



Admin dashboard

References and Resources:

- [1] Amazon Web Services. (n.d.). AWS Documentation. Retrieved from <https://docs.aws.amazon.com/>
- [2] HL7 International. (n.d.). Health Level Seven International. Retrieved from <https://www.hl7.org/>
- [3] HIPAA Journal. (n.d.). Health Insurance Portability and Accountability Act. Retrieved from <https://www.hipaajournal.com/>
- [4] Telemedicine Magazine. (n.d.). Telemedicine Regulations and Guidelines. Retrieved from <https://www.telemedmag.com/>
- [5] Gartner. (n.d.). Gartner Research and Insights. Retrieved from <https://www.gartner.com/en>
- [6] The Open Group. (n.d.). Healthcare Interoperability Standards. Retrieved from <https://www.opengroup.org/>
- [7] Healthcare Information and Management Systems Society (HIMSS). (n.d.). Healthcare IT News and Insights. Retrieved from <https://www.himss.org/>
- [8] Healthcare Information and Management Systems Society (HIMSS). (n.d.). Telehealth and Telemedicine Resources. Retrieved from <https://www.himss.org/>
- [9] American Medical Association (AMA). (n.d.). Digital Health and Technology Resources. Retrieved from <https://www.ama-assn.org/>
- [10] National Institute of Standards and Technology (NIST). (n.d.). NIST Special Publications. Retrieved from <https://www.nist.gov/>
- [11] Healthcare Information and Management Systems Society (HIMSS). (n.d.). Healthcare Data Security and Privacy Guidelines. Retrieved from <https://www.himss.org/>
- [12] AWS Architecture Center. (n.d.). AWS Architecture Best Practices. Retrieved from <https://aws.amazon.com/architecture/>
- [13] Healthcare Information and Management Systems Society (HIMSS). (n.d.). Healthcare Industry Trends and Insights. Retrieved from <https://www.himss.org/>

[14] Journal of Medical Internet Research (JMIR). (n.d.). Healthcare IT Research and Publications. Retrieved from <https://www.jmir.org/>

[15] Healthcare Information and Management Systems Society (HIMSS). (n.d.). Telemedicine Regulations and Guidelines. Retrieved from <https://www.himss.org/>